Dingyi Pei Moti Yung Dongdai Lin
Chuankun Wu (Eds.)

Information Security and Cryptology

Third SKLOIS Conference, Inscrypt 2007
Xining, China, August 31–September 5, 2007
Revised Selected Papers

Springer

Volume Editors

Dingyi Pei
Guangzhou University
Institute of Information Security
Guangzhou, China
E-mail: gztcdpei@scut.edu.cn

Moti Yung
Google Inc., and Columbia University
Department of Computer Science
New York, NY, USA
E-mail: moti@cs.columbia.edu

Dongdai Lin
SKLOIS, Institute of Software
Chinese Academy of Sciences
Beijing, China
E-mail: ddlin@is.iscas.ac.cn

Chuankun Wu
SKLOIS, Institute of Software
Chinese Academy of Sciences
Beijing, China
E-mail: ckwu@is.iscas.ac.cn

Library of Congress Control Number: Applied for

CR Subject Classification (1998): E.3, D.4.6, F.2.1, C.2, J.1, C.3, K.4.4, K.6.5

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-79498-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-79498-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12263486 06/3180 5 4 3 2 1 0

Preface

The Third SKLOIS Conference on Information Security and Cryptology (Inscrypt 2007) was organized by the State Key Laboratory of Information Security of the Chinese Academy of Sciences in cooperation with Qinhai University for Nationalities. This international conference was held in Xining, Qinhai Province of China, and was sponsored by the Institute of Software, the Chinese Academy of Sciences, the Graduate University of the Chinese Academy of Sciences and the National Natural Science Foundations of China.

By now, Inscrypt (the International SKLOIS Conference on Information Security and Cryptology) has become a tradition, and it is, in fact, a leading event in this area, which takes place annually in China. We are pleased with the continuous support by authors, committee members, reviewers, sponsors and organizers. Indeed, the research areas covered by Inscrypt are important, since modern computing (including communication infrastructures and applications) requires increased security, trust, safety and reliability. This need has motivated the research community worldwide to produce important fundamental, experimental and applied work in the wide areas of cryptography and information security research in recent years. Accordingly, the program of Inscrypt 2007 covered numerous fields of research within these general areas.

The international Program Committee of the conference received a total of 167 submissions from 21 countries and regions, from which only 43 submissions were selected for presentation, 33 of which in the regular papers track and 10 submissions in the short papers track. All anonymous submissions were reviewed by experts in the relevant areas, and based on their ranking, technical remarks and strict selection criteria the papers were chosen for the various tracks. We note also that reviews of submissions by committee members were hidden from their authors throughout the entire review process. We also note that due to the conference format, many good papers were regrettably not accepted.

Many people and organizations helped in making the conference a reality. We would like to take this opportunity to thank the Program Committee members and the external experts for their invaluable help in producing the conference program. We thank the conference Organizing Committee, the various sponsors and the conference attendees. Last but not least, we also express our thanks to all the authors who submitted papers to the conference, the invited speakers and the session Chairs.

August/September 2007

Dingyi Pei
Moti Yung

Inscrypt 2007

Third SKLOIS Conference
on Information Security and Cryptology

Xining, Qinghai, China
August 31 - September 2, 2007

Sponsored and organized by

State Key Laboratory of Information Security
(Chinese Academy of Sciences)
and
Qinhai University for Nationalities

General Co-chairs

Dengguo Feng	SKLOIS, Chinese Academy of Sciences, China
Youyi Zhang	Qinhai University for Nationalities, China

Program Co-chairs

Dingyi Pei	Guangzhou University, China
Moti Yung	Google inc and Columbia University, USA

Program Committee

Feng Bao	I2R, Singapore
Rana Barua	Indian Statistical Institute, India
Lejla Betina	U.K. Leuven, Belgium
Emmanuel Bresson	DCSSI Crypto Lab, France
Bogdan Carbunar	Motorola Labs, USA
Robert H. Deng	SMU, Singapore
Cunsheng Ding	HKUST, Hong Kong, China
Marc Girault	France Telecom, France
Bok Min Goi	Multimedia University, Malaysia
Dieter Gollmann	TU Harburg, Germany
Goichiro Hanaoka	AIST, Japan
Tor Helleseth	Bergen University, Norway
Lei Hu	SKLOIS, Chinese Academy of Sciences, China
Stamatiou Iwannis	University of Patras, Greece
Hongxia Jin	IBM Almaden Research Center, USA
Vladimir Kolesnikov	Bell-Labs, USA
Xuejie Lai	Shanghai Jiaotong University, China

DongHoon Lee	Korea University, Korea
Benoit Libert	U.C. Louvain, Belgium
Dongdai Lin	SKLOIS, Chinese Academy of Sciences, China
Michael Locasto	Columbia University, USA
Masahiro MAMBO	Tsukuba University, Japan
Wenbo Mao	HP, Beijing, China
David Naccache	ENS, France
Rei Safavi-Naini	Calgary, Canada
Mridul Nandi	Indian Statistical Institute, India
Juan Gonzalez Nieto	QUT, Australia
Giuseppe Persiano	University of Salerno, Italy
Junji Shikata	University of Yokohama, Japan
Vitaly Shmatikov	University of Texas at Austin, USA
Francesco Sica	Mount Allison University, Canada
Rainer Steinwandt	Florida Atlantic University, USA
Willy Susilo	UWO, Australia
Bogdan Warinschi	University of Bristol, UK
DongHo Won	Sungkyunkwan University, Korea
Chuankun Wu	SKLOIS, Chinese Academy of Sciences, China
Shouhuai Xu	University of Texas at San Antonio, USA
Yongjin Yeom	NSRI, Korea
Yuefei Zhu	Information Engineering University, China

Organizing Committee Co-chairs

Dongdai LIN	SKLOIS, Chinese Academy of Sciences, China
Yanhui Niu	Qinhai University for Nationalities, China
Chuankun Wu	SKLOIS, Chinese Academy of Sciences, China

Organizing Committee

Jiwu Jing	SKLOIS, Chinese Academy of Sciences, China
Fuming Qi	Qinhai University for Nationalities, China
Shengfu Zhang	Qinhai University for Nationalities, China
Yuqing Zhang	SKLOIS, Chinese Academy of Sciences, China
Zhenfeng Zhang	SKLOIS, Chinese Academy of Sciences, China

Secretary and Treasurer

Yi Qin	SKLOIS, Chinese Academy of Sciences, China

WEB/Registration

Guangsheng Miao	SKLOIS, Chinese Academy of Sciences, China

Table of Contents

Cryptanalysis of the SFLASH Signature Scheme[*]
(Extended Abstract)

Vivien Dubois[1], Pierre-Alain Fouque[1],
Adi Shamir[1,2], and Jacques Stern[1]

[1] École normale supérieure,
45 rue d'Ulm, 75005 Paris, France
{vivien.dubois,pierre-alain.fouque,jacques.stern}@ens.fr
[2] Weizmann Institute of Science, Israel
adi.shamir@weizmann.ac.il

Abstract. SFLASH is a signature scheme proposed by Patarin, Goubin and Courtois in 2001 [9,7] following a design they had introduced in 1998 [8]. SFLASH is reputed for being very fast and has been recommended by the NESSIE European Consortium since 2003 as the best known solution for implementation on low cost smart cards [5]. In this abstract, we present new attacks on the general design proposed by Patarin *et al.* [8] which allows to forge signatures in a few minutes for practical instantiations including the SFLASH scheme recommended by NESSIE [5].

Keywords: multivariate cryptography, signature, SFLASH, differential cryptanalysis.

Multivariate Cryptography is an area of research which attempts to build asymmetric primitives, based on hard computational problems related to multivariate quadratic polynomials over a finite field. Multivariate schemes have recently received much attention, for several reasons. First, the hard problems of reference are not known to be polynomial in the quantum model, unlike integer factorization and the discrete logarithm problems. More importantly, Multivariate Cryptography offers a large collection of primitives and problems of a new flavor. In general, multivariate schemes require modest computational resources and can be implemented on low cost smart cards. Moreover, these schemes benefit from several nice properties such as providing very short or very fast signatures. Also, they are quite versatile: a number of generic non-exclusive variations can be derived from a few basic schemes. Even when the original schemes are weak, variations are often considered to avoid structural attacks.

[*] This abstract compiles the results of two papers appearing in the proceedings of *EUROCRYPT 2007* and *CRYPTO 2007*. Part of this work is supported by the Commission of the European Communities through the IST program under contract IST-2002-507932 ECRYPT.

One of the more elaborate outcomes of Multivariate Cryptography is probably the SFLASH signature scheme. Designed by Patarin, Goubin and Courtois [9,7], the scheme is among the fastest signatures schemes known, with NTRUSign and TTS [3,10]. Although initial tweaks in the first version of SFLASH were shown inappropriate [2], the current version of SFLASH is considered secure, as testified from the recent acceptance of this primitive by the NESSIE European Consortium [5].

The structure of SFLASH is among the simplest in Multivariate Cryptography. Roughly speaking, SFLASH is a truncated C^* scheme. The C^* scheme was invented by Matsumoto and Imai in 1988 [4], and was shown to be insecure by Patarin in 1995 [6]. Later, Patarin, Goubin and Courtois considered the simple variation of C^* consisting in deleting from the public key a large number of coordinates [8]. Schemes derived from C^* by this principle are called C^{*-} schemes; they are well suited for signature. As soon as the number of deleted coordinates is large enough, C^{*-} schemes are considered secure. SFLASH belongs to the C^{*-} family and has been chosen as a candidate for the NESSIE selection, and finally accepted.

Our Results. Despite this success, it might be argued that the security of C^{*-} schemes remains insufficiently understood. In particular, one may rightfully question the reasons for the particular choice of parameters opted for in SFLASH. Might other parameters yield the same security?

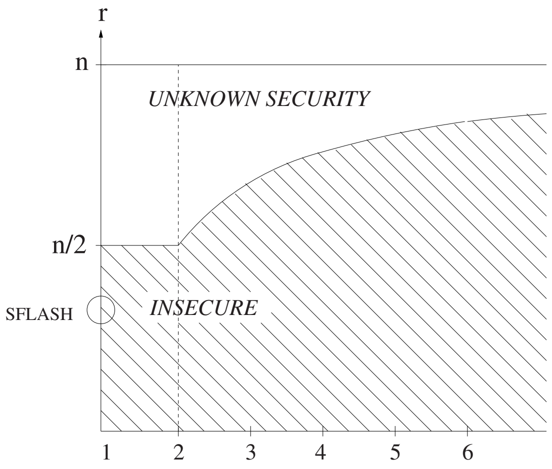
In this abstract, we report on new insights in the security of C^{*-} schemes whose main result is that most practical instantiations of C^{*-} schemes, including the SFLASH scheme recommended by NESSIE, are insecure. We indeed show that when the number of coordinates that have been deleted from the original C^* public key is less than one half, as appearing in all proposed realizations of C^{*-} schemes, the remaining coordinates can be used to recover a full C^* public key in polynomial time; this C^* public key can then be inverted using Patarin's attack [6] to forge a signature for an arbitrary message. For SFLASH, recovering a full C^* public key takes only a few minutes, and then a signature can be forged for an arbitrary message within a few seconds.

Our method to recover additional coordinates of the public key is through the composition by specific linear maps, which are related to some *commutation property* with the internal function. The definition of these maps depends on the secret key but we show that they can be identified by an attacker as they satisfy specific properties with respect to the *differential* of the public key. The differential, as introduced in [1], is a bilinear symmetric function defined from a quadratic function. Considering the differential is natural in hope of capturing properties of quadratic functions under concepts from linear algebra, which can be easily computed.

A first attack on C^{*-} schemes comes from considering *skew-symmetric* linear maps with respect to the differential of the public key. We show that the linear maps which are skew-symmetric with respect to the differential of a C^* public key do satisfy the commutation property that makes the attack possible. Moreover, the characteristic equation satisfied by these maps is linear and massively overdefined, which makes the vector space formed by these maps easy to recover through linear algebra even when the number of available public coordinates is very small (asymptotically a constant). The dimension of the space of skew-symmetric maps exactly corresponds to the gcd, denoted d, of the internal C^* parameter and the number of variables. However, when $d = 1$, this space only contains some trivial elements which are useless for the attack. In all other cases, when $d > 1$, the skew-symmetric maps can be used to recompose the C^{*-} public key into a full C^* public key, up to as few coordinates as a proportion $1/d$ of the original ones. The SFLASH scheme escapes this attack since its parameters were chosen prime numbers and therefore satisfy the complementary condition $d = 1$.

A second attack on C^{*-} schemes is obtained by relaxing the condition of skew-symmetry. We indeed consider the skew-symmetric action on the differential of a wider class of commuting maps and show a specific property of these maps which is to keep the differential into a small dimensional subspace. Alternatively, the considered maps are skew-symmetric with respect to the differential *modulo* this small subspace. This condition, as a refinement of the global skew-symmetry condition, is of course satisfied by the skew-symmetric maps but it also admits many more solutions which can also be used for the purpose of the attack and whose existence is independent of the parameter d. We then show that such maps can be recovered from a C^{*-} public key provided it features more than the half of the original coordinates and when $d = 1$. The SFLASH scheme, as well as other proposed instantiations, falls under this attack. On the other side, for C^{*-} schemes with $d > 1$, this attack is not interesting since the first attack achieves a better bound in this case.

The figure below summarizes the scope of the two attacks, separated by a dotted line, where r is the number of deleted coordinates and n is their initial number.



References

1. Fouque, P.-A., Granboulan, L., Stern, J.: Differential Cryptanalysis for Multivariate Schemes. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 341–353. Springer, Heidelberg (2005)
2. Gilbert, H., Minier, M.: Cryptanalysis of SFLASH. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 288–298. Springer, Heidelberg (2002)
3. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: NTRUSIGN: Digital Signatures Using the NTRU Lattice. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 122–140. Springer, Heidelberg (2003)
4. Matsumoto, T., Imai, H.: Public Quadratic Polynominal-Tuples for Efficient Signature-Verification and Message-Encryption. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988)
5. NESSIE: New European Schemes for Signatures Integrity and Encryption. Portfolio of recommended cryptographic primitives, `http://www.nessie.eu.org/index.html`
6. Patarin, J.: Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt 1988. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)
7. Patarin, J., Courtois, N., Goubin, L.: FLASH, a Fast Multivariate Signature Algorithm. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 298–307. Springer, Heidelberg (2001)
8. Patarin, J., Goubin, L., Courtois, N.: $C^*{}_{-+}$ and HM: Variations Around Two Schemes of T. Matsumoto and H. Imai. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 35–49. Springer, Heidelberg (1998)
9. Specifications of SFLASH. Final Report NESSIE, pp. 669–677 (2004)
10. Yang, B.-Y., Chen, J.-M.: Building Secure Tame-like Multivariate Public-Key Cryptosystems: The New TTS. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 518–531. Springer, Heidelberg (2005)

On the Evolution of User Authentication: Non-bilateral Factors

Moti Yung[1,2]

[1] Google Inc.
[2] Columbia University

Abstract. In this position note we review the notion of "User Authentication," i.e., the idea of a person facing the computer and in need of authenticating herself to the system in order to get access to her account. We analyze the state of the art of user authentication and that of "Authentication Factors," i.e., the techniques and instruments the user can present to the computer. We compare the state of the art of these factors with the acceptable textbook view.

1 Introduction

Authentication factors are the basic instruments available to a human user to authenticate herself in order, e.g., to convince a computing system of her true "identity" as is known or registered in the system. These factors, presented by a human claiming to be a specific user (and get access to that specific user's account), are passed to elements of the system (e.g., a software program in a server) which, in turn, make a decision whether the human is the claimed specific user. Determination is based on registered information that the elements hold about the actual specific user. The accepted classification of authentication factors, as expressed in numerous textbooks, distinguishes three basic types of factors:

1. "Something You Know" (e.g., a password);
2. "Something You Have" (e.g., a device generating one-time passwords); and
3. "Something You Are" (e.g., biometrics information).

Given the current state of the art and the fact that nowadays, Internet Computing is based on human interaction with the computing infrastructure, this works re-examines the notion of authentication factors. Based on the evolution of computing systems and modern computing in general, the work reviews the parallel evolution of authentication factors and explains it and its impact on user security.

2 The Traditional Characteristics Vs. Non-bilateral Factors

The above three characteristics of the nature of authentication factors is very natural, since a user facing a system indeed can present to it evidence of the three basic characteristics. However, these characteristics were made in the setting of a basic model of a user facing a machine or an operating system , which was a good view of a traditional time-sharing stand-alone system. what we note is that modern computing environment is much different, and we claim that in modern environments it is rarely the case that a user faces "alone" a single system. In fact, nowadays there are various system elements that are present in modern infrastructure. Elements can be part of the computing components, networking infrastructure components, or other aspects of the entire system. Today, when a user presents the factors, they may be directly given to the system or there may be a sequence of transfers. Even if virtually, the user believes he has interacted with "the system," the infrastructure behind this interaction is complex.

A first demonstrative example consider the case where the user first presents a password to a smart card which activates the smartcard to produce an authenticator string; the card then authenticates on behalf of the user to a local system by presenting the string, that, in turn, validates the authenticator, and presents a credential that authenticates itself with an assertion about the authenticity of this user in this connection. The assertion may be a digitally signed statement accompanied by a certificate. The credential and assertion, in turn, are presented to a remote server or a member of a federation of server that mediates the authentication; this results in the connection to the user being validated as coming from the actual user and may allow this connecting user access the resources associated with this actual user. In fact, in this example we see how the user's initial authentication is "carried through" the system via a chain of trust among system elements. The involvement of human players and system elements in the process led to calling this protocol that involves users, factors and machines, "a ceremony" [2].

The fact that modern environments include more than merely the user and a single system element gives the opportunity to extend the possible characteristics of user authentication factors. The participants do not even have to be a chain of system elements along a linear flow of a message. This basic issue can be summarized as following:

– The traditional characteristics assume a *Bilateral Relation* between the user and the system; modern environments include additional elements and components.

The above observation implies that we can now consider factors that involve additional parties, channels, computing elements and pieces of information available to the system. Thus, side parties and components can take part in the authentication ceremony, and we can use factors such as:

1. The mobile phone line of the user or of an e-mail account belonging to the user which generate an independent channel to the user; this is an independent validation channel ("Something Registered About the User");
2. The IP address or the laptop computer mac address of the user's computer, ("Something Embedded in the User's Computing Environment");

3. A remote server associated with the user that the user already logged into and expressed the intention to further authenticate with another system element ("Some Support Tools the Computing Environment Has");.

The above are examples of factors based on the user's computing environment: knowledge about it, its physical characteristics and available elements and channels in it. They can be used and are, indeed, used in identifying the likelihood of the claimed user being the specific authentic one. In some abstract general sense the user environment may be thought to be part of the user and these factors can all be thought of as "something the user has." But, in reality these are not something the user possesses, but rather something in the environment associated with the user. They may be associated with the user at some time (when the user is at home) but not at other times (when the user is at a cyber cafe, say), and may change as the user re-configures parts of her environment (e.g., buys a new computer).

The independence of the factors that belong to the infrastructure is important in order for the system to use them as factors supporting the user's identity. For example, the mobile phone platform and the computer platform may be the same, and a phone call over the wire and over wireless LAN may get mixed as the infrastructure gets more complicated. The relation and independence of infrastructure components has to be analyzed and the system has to be aware of the source of the various factors.

We note that the more the factor known by the infrastructure is actually a factor that is essential part of the infrastructure the stronger it is. If a factor gives some physical meaning to the computing, e.g., an IP address is hard to spoof as a factor if used for routing the communication of the interaction itself. The more the factor is part of the correctly completion of the overall interaction in this physical sense, the better it seems to be.

Beyond Technical Components: Factors do not have to be part of the computing and communication elements. It was pointed out [1] that human components have to be considered and can, in fact, help in the authentication process. Indeed, social relationships to which the user belongs and organizational structure in which the user takes place can serve as additional factors. Relying on already authenticated users in some formal and careful way amounts to *"Someone You Know"* factor.

Keep New Technologies in Mind: It is important to remember that as technologies change one may attempt to employ them in the area of authentication factors. One such technology is "Internet Search Technology". It can be used to find facts that the claimed user needs to know and use for extension of "something you know" (or perhaps "something you (virtually) are") to *"something the infrastructure as a whole knows about you."* The idea here is that the specialized search engine knows where to look for details about a user fast, while generic search is still slow when personal information of specific nature is involved. This example is an interesting twist on privacy on the web, where relative loss of privacy is exploited to secure the authentication process! With lack of privacy the available details on the web may replace some of the registration process where users give personal details to be used as “something you know” factor.

3 How Are Factors Employed Nowadays?

Enterprise User vs. Consumer User: There are a number of different settings where authentication factors are used, which determines the possibility of usage and effectiveness of the factors. The enterprise setting where a user is an employee is much different from the e-commerce user which typifies the consumer case. It may be easier to provide devices to employees than to end consumers (who may be casual), which has been the case in reality. The participation of non-bilateral factors and mechanisms may depend on the setting as well, for example, vouching by a third party requires a well established social structure, typical inside enterprises.

Absolute Reliance vs. statistical Reliance: The level of reliance on factors is another issue. Authentication factors traditionally have been used to ensure the system with close to absolute certainty of the identity of the user. In modern systems the factors may be treated as providing some probability of assurance and by gathering enough factors (possibly incrementally), the probability of an identity can be calculated. In fact, at the end of a possibly lengthy session with many checks, the system can concludes that the user “has passed” the authentication protocol, based on statistics (i.e., some confidence level). The involvement of factors can be done adaptively based on assessment of the success so far in the process. It seems that absolute reliance (a factor that gives a yes/no answer) was the traditional method, where as the Internet evolves and the level of threat increases it makes sense to rely on factors statistically, and to collect multi-factor input. In this mode, not any single source of authentication needs to be perfect. (We note that engineering wise, it is often easier to build a system that is made out of imperfect components, and use multiplicity of such components to create a more reliable system. The “statistical confidence based factors” are mechanisms that possess this nature of relatively and possibly imperfect components).

Changing the Task of Authentication: As can be seen over the years, indeed, the use and purpose of authentication factors can be changed as systems evolve. One important issue in the context of Internet access and services is the fact that the server (e.g., of a bank or a merchant) has to prove its authenticity to the user as well (i.e., authentication in the reverse direction). Since the factors presented to the user have to be interpreted by a user looking at her browser, the security of user interfaces, human factors in user interfaces and defining end-to-end two way authentication to a browser user are important. The notions of how to prevent the user from getting into a wrong web page is especially important. User friendliness of interfaces and usability issues in general play now an important part in achieving the tasks of authentication. (Note that the extended nature and requirement of user authentication in various web-based scenarios is beyond the scope of the current note; here it serves only to demonstrate the current issues with authentication factors).

4 Ask: Who Activates the Factors?

As noted above, lack of robust two-way authentication factors has opened the door for attacks that attempt social engineering *en masse*, namely trying to direct users to a fake web cite where the user inserts her authentication credentials. This type of attacks called phishing attacks are common place nowadays. Besides social engineering via faked sites and tempting email, the attack of directing users to wrong sites that are faked can be done by contaminating the resolve infrastructure or by malware on the user computer.

In phishing attacks users give their factors such as answers to questions and passwords to the site, which may use them elsewhere. Thus an important issue is how limited in time the credentials are and whether they are limited and bound to the current site and cannot be used by attackers who may attempt to contact the real site and, say, extract money from the user account employing the stolen credentials. Credentials that are bound and are non-reusable are preferable, as they limit the opportunity of the attackers.

The user may give away credentials like passwords, account number, account name, answers to questions in the knowledge base, etc. These credentials can be used by the attackers. However, new factors like mac address of a computer, a cookie inserted in the user's browser, or a call to a mobile phone line owned by the user, do not change ownership by the user even if she becomes a victim of a social engineering attack and the attacker cannot "demand" to get them. Thus, the last type of credentials have been known as passive factors, whereas the first type of factors are activated by the user.

In the view of the multi-lateral nature of modern factors, to our belief, we can analyze the difference between the various credential types. We believe that the basic difference is the underlying fact of *who activates the factor*. If the user herself activates them and they are reusable within some time and space limitations, they can be re-used by attackers (in off-line or on-line attacks, depending on the "validity window" for the credential). If, on the other hand, the credentials are factors that are activated by third parties: the communication infrastructure, independent elements of the computing base and other parties, then the user is in no position to give it away. In fact, the advantage of some of the modern factors is due to the non-bilateral nature of these factor and the fact that the user does not possess them, even though the factors are employed to authenticate the user. Thus, one has to pay attention to analyze separately and carefully "User-Activated" vs. "Third-Party Activated" factors.

In order to attack the third party activation of factors (as opposed to the user self-activation), different methods are needed then merely attacking the user. One needs to pay attention to potential attacks on these new factors and to classify the parties that are involved: i.e., a cookie may be stolen more easily than a mobile phone owned by a user that is used to call the user back during authentication. Factors related to the functioning of the interaction (IP address, say) are harder to change as was already mentioned. In any event, the involvement of multi-party in the new authentication game changes its nature as it is not bi-lateral anymore. Involving these new multi-lateral factors enhances the notion of user authentication and allows prevention of attacks that isolate the user. Granted, to be used successfully the new game may require further analysis of all parties involved, since an extended game with extra parties also means extra security analysis is required.

Future Factors: The computing infrastructure is evolving rapidly and information technology and physical aspects of the wold are merging and computers are embedded in many new places. Given this trend , the above issues should be kept in mind. We expect to view ubiquitous computing as a new wave of opportunities for future user authentication by enabling new factors and new parties to take place in the process (e.g., physical location based factors, gadgets oriented factors, involvement of consumer electronic devices, ability to probe user familiarities with certain knowledge bases based on issues such as current location, background and recent interactions, and so on). We expect that the mixing of physical security, procedural security and information security is likely to increase the variability of authentication factors.

5 Conclusions

One important lesson of the above analysis and the position this note takes, is the fact that the notion of computer user authentication is deeply connected to the infrastructure of computing. As the latter evolves so does the former. Furthermore, technological changes should be considered and off the book traditional assumptions regarding security measures be updated appropriately (in the security area "following the book" and "ignoring the book" are equally important!).

Acknowledgments. Discussions over the years with many individuals have contributed to the position expressed here (which does not discuss any specific technology and does not express any preference among various methods). Given the dynamically changing nature of user authentication, these discussions have been and still are very helpful. Among others, many thanks are due to Ross Anderson, Louie Gasparini, Seth Gettic, Brian Gladstein, Lior Golan, Ari Juels, Burt Kaliski, Aaron Kechley Amit Klein, John Lin, Nasir Memon, Magnus Nystrom, Ron Rivest, and Mike Szydlo.

References

1. Brainard, J., Juels, A., Rivest, R.L., Szydlo, M., Yung, M.: Fourth factor authentication: somebody you know. In: ACM Conference on Computer and Communication Security (CCS), pp. 168–178 (2006)
2. Ellison, C.: UPnP security ceremonies design document (October 2003)

ECDSA-Verifiable Signcryption Scheme with Signature Verification on the Signcrypted Message

Raylin Tso, Takeshi Okamoto, and Eiji Okamoto

Department of Risk Engineering
Graduate School of Systems and Information Engineering
University of Tsukuba,
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan
{raylin,ken,okamoto}@risk.tsukuba.ac.jp

Abstract. In this paper, we propose a signcryption scheme which provides all the following properties at the same time. (1) forward security: the private key of a sender does not help any attack to break the confidentiality of any signcrypted message generated by the sender. (2) signature verification on the signcrypted message: this is done without revealing the original message. (3) a standardized signature mechanism: in our scheme, both the non-repudiation of the message and the signcrypted message can be done using the standardized digital signature algorithm; ECDSA. The efficiency and features of our scheme are compared with other schemes and the security is proved in the random oracle model.

Keywords: ECDL problem, ECDSA, GDH problem, signcryption.

1 Introduction

Confidentiality and *authenticity* are two of the most important goals in setting a cryptographic system. Until recently, these two goals are considered separately. In the case of public-key cryptography, confidentiality is provided by encryption schemes and authenticity is provided by signature schemes. To achieve both confidentiality and authenticity in one logical step, a traditional way is to perform the sign-then-encrypt ($St\mathcal{E}$) approach. The cost of $St\mathcal{E}$ is equal to the total cost of signature and encryption.

In 1997, Zheng [17] introduced the first cryptographic primitive that combines encrypting and signing in one step at a lower communication and computational cost, which is called *signcryption.* Zhang's idea opened a new research era and, following his pioneering work, a lot of new signcryption schemes or improvements have been proposed (e.g., [1,2,12,16]).

Contributed by a large quantity of investigation, nowadays, except the properties of confidentiality and authenticity, proper signcryption schemes should also provide (at least) the following properties:

- **Non-repudiation of the message:** The receiver of a message should have the ability to prove to a third party that the sender indeed sent the message.

This ability ensures that the sender of a message cannot later deny having sent this message.

- **Public verifiability:** A signcryption scheme is publicly verifiable if, without the knowledge of the receiver's private key, a third party can efficiently verify that a signcrypted message ξ is a valid signature on m signed by a sender.
- **Forward security:** A signcryption scheme provides forward-secure encryption if knowing the private key of a sender cannot reveal the plaintext from any signcrypted message generated by the sender.

On the other hand, to achieve simple and secure non-repudiation using a standard signature scheme, several research has been done [12,16]. In Yum and Lee's scheme [16], the non-repudiation of the message can be verified using the standardized Korea Certificate-Based Digital Signature Algorithm, KCDSA [14], and in Shin et. al. 's scheme [12], the non-repudiation of the message can be done using one of the most widely used standardized signature schemes, Digital Signature Algorithm (DSA) [8]. However, scheme in [16] does not secure even in the sense of semantic security and schemes in [12] use a modified DSA instead of using DSA directly. Furthermore, these schemes [12,16] as well as other discrete-logarithm (DL) based signcrytion schemes up to now are not forward secure.

We notice that most of the recent signcryption schemes (eg., [3,6]) providing forward security are based on the property of *bilinear pairing* [4,5]. In this case, no standardized signature scheme can be used for the verification phase. In addition, pairing based signcryption schemes are usually less efficient than those non-pairing-based schemes.

1.1 Our Contributions

The main contribution of this paper is to propose signcryption scheme which not only provides the above mentioned properties but also the following two additional properties:

- **Signature verification on the signcrypted message:** In a signcryption scheme, a receiver should have the ability to prove the identity of the sender who generated the signcrypted message while to keep the confidentiality of the original plaintext to any third party.
- **Standardized verification mechanism:** It is a desirable property if, in a signcryption scheme, the non-repudiation of a sender can be verified using some standardized verification algorithms such as DSA or ECDSA. This property makes the non-repudiation more simple since the existing infrastructure for these standardized schemes can be used without modification.

Signature verification on the signcrypted message is required in some situations when the identity of a sender has to be verified but the content of the message must be kept secret to the verifier. It is obvious that $St\mathcal{E}$ does not provide signature verification on the signcrypted message (without revealing the content of the message). On the other hand, this property has been investigated in some literatures such as [6,7]. In [6,7], they allow the origin of a signcrypted message to be verified publicly without any assistance from the receiver. This kind of constructions may be useful against the firewall systems, but, their constructions make problems when a receiver who wants to convince a third party that the plaintext actually comes from the sender. This is because that in their schemes, there is no guarantee on the unique pair of a signcrypted message and the corresponding plaintext.

In our scheme, the non-repudiation of the message and the signature verification on the signcrypted message are done by the widely used standard signature algorithm, the Elliptic Curve Digital Signature Algorithm (ECDSA) [15] which is the elliptic curve analogue of the DSA and is the only signature standard based on the elliptic curve. It has been accepted as an ISO standard (ISO 14888-3), an ANSI standard (ANSI X9.62), an IEEE standard (IEEE P1363) and a FIPS standard (FIPS 186-2).

We emphasize that this is the first work which provides all the above mentioned properties in one signcryption scheme at the same time. Moreover, this is the first work on signcryption schemes where both the signature on ciphertext and on plaintext can be verified and the correctness between the ciphertext and the corresponding plaintext can be assured.

2 Preliminaries

This section gives some cryptographic primitives and definitions required for our construction.

2.1 Security Assumptions

Definition 1. Discrete Logarithm (DL) Problem: Let G be a cyclic group of prime order p and g be a generator of G. The DL problem to the base g means the following problem:

$$\text{Given } \ g, h \in G, \ \text{ find an integer } x \text{ such that } g^x = h.$$

Definition 2. Elliptic Curve Discrete Logarithm (ECDL) Problem: The DL problem in the elliptic curve setting is defined as follows: Let $E(F_q)$ denote an elliptic curve E defined over a finite field F_q. Given a point $P \in E(F_q)$ of order n, and a point $Q = lP$ where $0 \leq l \leq n-1$, determine l.

The DL problem as well as ECDL problem are believed to be difficult and also to be the hard direction of a one-way function.

Definition 3. Computational Diffie-Hellman (CDH) Assumption: Let G be a cyclic group of prime order p and g be a generator of G, the CDH assumption states that given (g, g^a, g^b) for randomly picked $a, b \in Z_p^*$, there exists no polynomial time algorithm which can find an element $C \in G$ such that $C = g^{ab} \bmod p$ with non-negligible probability.

Definition 4. Decisional Diffie-Hellman (DDH) Assumption: Let G be a cyclic group of prime order p and g be a generator of G. The DDH problem is to distinguish ab from c by given (g, g^a, g^b, g^c), where a, b, c are random elements of Z_p^*.

The CDH assumption and the GDH assumption in the EC setting can also be defined accordingly.

Definition 5. Gap Diffie-Hellman (GDH) Assumption: The assumption that in a group where the CDH problem is hard but the DDH problem is easy is called the GDH assumption.

2.2 Notations

Throughout this paper, we will assume that *Alice* A is a sender and *Bob* B is a receiver. The following notations will be used in this paper.

- $a||b$: a concatenation of two strings a and b.
- E/D: a symmetric key cryptosystem based encryption/decryption algorithm secure against *chosen plaintext attacks*. By $E_k(m)$, we mean a message m is encrypted using a key k and the encryption algorithm E. By $D_k(c)$, we mean a ciphertext c is decrypted using a key k and the decryption algorithm D.
- H : a cryptographic one-way hash function.
- T: a secure hash function.
- $bind_{A,B}$: the concatenation of two identities of A and B.
- $PointComp()$[1]: point compress function.
- $PointDecomp()$: point decompress function.

3 Proposed Scheme

In some applications, it is desired that the sender's identity can be identified while the content of the message can be kept secret to a third party. In this section, we propose an ECDSA-verifiable signcyrption scheme with signature verification on the signcrypted message. The signature verification on the signcrypted message and the non-repudiation of the original message both require some information opened by the recipient. But, this scheme assures to the third party that c is really the ciphertext of the unique message m.

System Setting: Let q be a large prime greater than 2^{160}, $E(F_q)$ be an elliptic curve defined over the finite field F_q, G be a point on $E(F_q)$ such that the ECDL problem in the subgroup $< G >$ generated by G is infeasible and n be the prime order of G. The system parameters are (q, G, n, E, D, H, T).

[1] See Section 6.5.4 of [13]. The communication cost required for points on elliptic curves can be reduced by (almost) 50%.

Key Generation: For each entity i, the private/public key-pair for i is (d_i, Q_i), where $d_i \leftarrow_R \{1, \cdots, n-1\}$, and $Q_i \leftarrow d_i G$.

Signcrypting: Suppose *Alice* wants to signcrypt a message $m \in \{0,1\}^*$ to *Bob*, she does the following steps:

1. Pick $k \leftarrow_R \{1, \cdots, n-1\}$ and compute $R \leftarrow (x_1, y_1) = kG$.
2. Compute $(x_1, \alpha \in \{0,1\}) \leftarrow PointComp(E(F_q), R)$.
3. Compute $r \leftarrow x_1 \bmod n$ and go to step 1 if $r = 0$.
4. Compute $K \leftarrow (x_2, y_2) = kQ_B$ and $\mathcal{K} \leftarrow H(x_2)$.
5. Compute $(\mathcal{K}_{Enc}, u) \leftarrow T(\mathcal{K})$, where $u \in \{1, \cdots, n-1\}$.
6. Compute $U \leftarrow uR$, $\hat{c} \leftarrow E_{\mathcal{K}_{Enc}}(m)$, and $c \leftarrow \hat{c}||\alpha$.
7. Compute $h \leftarrow H(c||bind_{A,B}||x_1||U)$ and $s \leftarrow (ku)^{-1}(h + d_A r) \bmod n$.

The signcrypted message is $\xi \leftarrow (c, x_1, s)$ and *Alice* sends ξ to *Bob*.

Unsigncrypting: To Unsigncrypt ξ, *Bob* does the following steps:

1. Recover $R = (x_1, y_1) \leftarrow PointDecomp(E(F_q), x_1, \alpha)$.
2. Compute $K \leftarrow (x_2, y_2) = d_B R$ and $\mathcal{K} \leftarrow H(x_2)$.
3. Compute $(\mathcal{K}_{Enc}, u) \leftarrow T(\mathcal{K})$.
4. Compute $U \leftarrow uR$, $h \leftarrow H(c||bind_{A,B}||x_1||U)$, and $r \leftarrow x_1 \bmod n$.
5. Compute $e_1' \leftarrow h/s \bmod n$ and $e_1 \leftarrow u^{-1}e_1' \bmod n$.
6. Compute $e_2' \leftarrow r/s \bmod n$ and $e_2 \leftarrow u^{-1}e_2' \bmod n$.
7. Compute $R' = (x_1', y_1') \leftarrow e_1 G + e_2 Q_A$.
8. Accept ξ as a valid signcrypted message if and only if $x_1' \equiv x_1$ (ie., $x_1' \bmod n = r$), otherwise, reject ξ and stop.
9. Recover the message as $m \leftarrow D_{\mathcal{K}_{Enc}}(\hat{c})$.

Signature verification on the signcrypted message: The receiver, *Bob*, opens h with the original signcrypted message $\xi = (c, x_1, s)$ to a third party. The third party then do the following steps:

- Compute $r \leftarrow x_1 \bmod n$, $e_1' \leftarrow h/s \bmod n$ and $e_2' \leftarrow r/s \bmod n$.
- Compute $U' \leftarrow e_1' G + e_2' Q_A$.

He then accepts ξ as a valid signcrypted message if and only if $h = H(c||bind_{A,B}||x_1||U')$.

Non-repudiation of the message: By opening $\mathcal{K}$ together with the signcrypted message ξ, anyone can check the origin of ξ and recover the original message m according to the unsigncrypt algorithm.

Consistency of the unique pair of (m, c): If U' obtained from the signature verification on the ciphertext phase is equal to uR computed in the non-repudiation of the message phase, then a verifier can conclude that $\mathcal{K}_{Enc}$ (computed from $T(\mathcal{K})$ with $\mathcal{K}$ obtained in the non-repudiation of the message phase) is really the encryption key so m is really the plaintext with regard to the ciphertext c.

4 Discussion

In the above scheme, signature verification on the signcrypted message is done using the technique similar to ECDSA but checking the validity of h instead of r. In this case, (x_1, h, s) is the signature on the message $c||bind_{A,B}||x_1||U'$. Therefore, when U' opened, any one can verify the sender's identity.

To see that the forward security is preserved even when U' opened, we see that r, s are public and h can be computed by $H(c||bind_{A,B}||x_1||U')$, therefore, when a sender *Alice*'s private key d_A is revealed, one can compute $ku \leftarrow s^{-1}(h + d_A r) \bmod n$. But, k is masked by u so cannot be computed. On the other hand, to compute u from $U = uR$ or from $e'_1 G + e'_2 Q_A$ is intractable because of the ECDL problem. The attempt to find k directly from R is also impossible because of the ECDL problem. Therefore, even when d_A revealed, k is still kept secret and $K = kQ_B$ is kept secret to any third party whenever the receiver's private key is not disclosed.

Since the non-repudiation of the message is based on ECDSA, the unforgeability of the scheme is based on the unforgeability of ECDSA. On the other hand, the confidentiality of the scheme is depended on the the secrecy of the session key K and the security of the symmetric encryption/decryption algorithm.

5 Efficiency and Features Comparison

In Table 1, the efficiency and security features of our scheme is compared with other DL-problem based signcryption schemes and the Sign-then-Encrypt ($St\mathcal{E}$) scheme. In the DL-problem based signcryption schemes, since exponentiation is the most costly computation comparing to other operations, we only consider the total number of exponentiations (denoted by Exp.) required by a sender and a receiver. Our scheme is also DL-problem based but in an elliptic curve setting, so the most costly computation is the elliptic curve multiplicative computation (denoted by EC.).

Table 1. Efficiency and Features Comparison

Scheme	Efficiency		Features				
	Commun. Cost	Compu. Cost	Semantic Security	Forward Security	Public Veri.	Cipher Veri.	Standard Sign.
Zheng[17]	$2\|q\|$	3Exp	Yes	\	\	\	\
Bao&Deng[2]	$2\|q\|$	5Exp	\	\	Yes	\	\
Yum&Lee[16]	$2\|q\|$	5Exp	\	\	Yes	\	KCDSA
SC-DSA+[12]	$2\|q\|$	5Exp	Yes	\	Yes	\	DSA
$St\mathcal{E}^{\dagger}$	$\|p\| + 2\|q\|$	6Exp	Yes	Yes	Yes	\	DSA
Ours	$\|q\| + \|n\|$	7EC	Yes	Yes	Yes	Yes	ECDSA

$|p| \approx 2^{1024}$, $|q| \approx 2^{160}$, $q \approx n$.
† DSA + ElGamal public key cryptosystem.

In DL-problem based signcryption schemes without using the bilinear pairing (see [4,5] for details), our scheme is the only scheme providing both non-repudiation of the signcrypted message and forward security.

To compare with $St\mathcal{E}$, we know that in $St\mathcal{E}$, to provide the IND-CCA2 secure, the encryption scheme must be IND-CCA2 secure, while in our scheme, a IND-CPA secure symmetric scheme is enough to provide the same security (see Theorem 1 in the next section). In addition, $St\mathcal{E}$ is more secure than other DL-based signcryption schemes but our scheme provides more security features than $St\mathcal{E}$ since $St\mathcal{E}$ does not provide any method for a third party to verify the origin of the signcrypted message. Furthermore, our scheme is much more efficient than $St\mathcal{E}$ in communication cost since $|n| \approx |q|$ but $|p| >> |q|$.

6 Security Results

This section discusses the security of our schemes in the random oracle model. We first discuss the EUF-ACMA security of our schemes.

Definition 6. A signcryption scheme is said to be EUF-ACMA secure if, for any polynomial-time adversary $\mathcal{F}$, the advantage defined by

$$Adv_{\mathcal{F}}^{\substack{euf-\\acma}} \triangleq Pr\left[\begin{array}{l|l} & para \leftarrow Setup(1^\lambda)\\ & sk_i \leftarrow Ex(pk_i, para)\\ Veri(m^*,\xi^*,pk_i^*,sk_j^*)=1 & \xi_i \leftarrow SC(m,pk_i,pk_j,para)\\ (pk_i^*,pk_j^*,m^*,\xi^*)\notin SC_{list} & m \leftarrow USC(\xi_i,pk_i,pk_j,para)\\ (pk_i^*,pk_j^*,m^*,\xi^*)\notin USC_{list} & (m^*,\xi^*) \Lsh \\ & \quad \mathcal{F}^{\substack{SC,USC,\\T,H}}(pk_i^*,pk_j^*,para)\end{array}\right]$$

is negligible, where

- SC is the signcryption oracle which takes any message m, a sender's public key pk_i and a receiver's public key pk_j as input, the output is a valid signcrypted message ξ_i.
- USC is the unsigncryption oracle which unsigncrypt any ξ_i and outputs the plaintext m from any input $(\xi_i, y_i, y_j, para)$.

In addition, SC_{list} and USC_{list} are the query/answer lists coming from Signcryption oracle SC, and Unsigncryption oracle USC respectively during the attack. In the random oracle model, the attacker can also access to the random oracles T and H. The probability is taken over the coin tosses of the algorithms, of the oracles, and of the forger.

In our schemes, the unforgeable property has to be considered in two different cases: the unforgeability of the signature on the signcrypted message (ie., ciphertext) and the unforgeability of the signature on the original message (ie., plaintext).

Unforgeability of the signature on the original message: In this case, an attacker tries to forge a new signcrypted message (c, x_1, s) which can pass the unsigncrypting algorithm.

In our scheme, if we ignore the encryption phase, then, it is easy to see that the signature generation phase and the signature verification phase are both following the procedure of ECDSA. Therefore, it can be concluded that the unforgeability of the scheme is based on the unforgeability of ECDSA. To explain more precisely, if there exists an adversary who can break the EUF-ACMA of our scheme by a forgery (c, x_1, s), then, (r', s') with $r' = x_1 \bmod n$ and $s' = su \bmod n$ is a valid ECDSA signature on the message $c||bind_{i,j}||x_1||U$. Here i, j represents the sender and the receiver 's identities and u is computed by the receiver according to the unsigncryption protocol without using a signer's private key. Consequently, breaking the unforgeability of our scheme means breaking the unforgeability of ECDSA.

Although the security of ECDSA has not been proven yet, it is widely used in practise and is believed to be EUF-ACMA secure. Up to now, no significant security flaws on ECDSA are know. □

Unforgeability of the signature on the signcrypted message: In this case, an attacker tries to forge a new signature (x_1, h, s) on c (ie., ciphertext) which can pass the verification algorithm at the signature verification on the signcrypted message phase.

In our scheme, the unforgeability of the signature (ie. (x_1, h, s)) on c is similar but different to a ECDSA signature. In this case, to prove the unforgeability, we first prove the unforgeability of the following signature scheme. In the following scheme, the system setting and key generation are the same as the proposed scheme.

Signing: To sign a message $m \in \{0,1\}^*$, *Alice* with key-pair (d_A, Q_A) does the following steps:

1. Pick $k \leftarrow_R \{1, \cdots, n-1\}$ and compute $R \leftarrow (x_1, y_1) = kG$.
2. Pick $u \leftarrow_R \{1, \cdots, n-1\}$ and compute $U \leftarrow uR$.
3. Compute $r \leftarrow x_1 \bmod n$ and go to step 1 if $r = 0$.
4. Compute $h \leftarrow H(m||x_1||U)$ and $s \leftarrow (ku)^{-1}(h + d_A r) \bmod n$.

The signature on m is $\sigma \leftarrow (x_1, h, s)$.

Verification:

1. Compute $r \leftarrow x_1 \bmod n$, $e_1 \leftarrow h/s \bmod n$ and $e_2 \leftarrow r/s \bmod n$.
2. Compute $U \leftarrow e_1 G + e_2 Q_A$.
3. Accept σ if and only if $h = H(m||x_1||U)$.

It is easy to see that this is the underlying signature scheme used in our scheme for the signature verification on the signcrypted message. Specifically, they are identical when we set m of the signature scheme to be equal to $c||bind_{A,B}$ of the proposed signcryption scheme. Consequently, the unforgeability of the signature scheme implies the unforgeability of the signature on the signcrypted message of the proposed signcryption scheme The security proof of the signature scheme requires the following lemma.

Lemma 1. (The Forking Lemma).[10] Let $(\mathcal{K}, \mathcal{S}, \mathcal{V})$ be a digital signature scheme with security parameter k, with a signature of the form $(m, \sigma_1, h, \sigma_2)$, where $h = \mathcal{H}(m, \sigma_1)$ and σ_2 depends on σ_1 and h only. Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data and which can ask q_h queries to the random oracle, with $q_h > 0$. Assume that, within the time bound T, $\mathcal{A}$ produces, with probability $\varepsilon \geq 7q_h/2^k$, a valid signature $(m, \sigma_1, h, \sigma_2)$. Then there is another machine which has control over $\mathcal{A}$ and produces two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1, h', \sigma_2')$ such that $h \neq h'$, in expected time $T' \leq 84480Tq_h/\varepsilon$.

Let $\sigma_1 = (x_1, U)$, $h = H(m||x_1||U)$ and $\sigma_2 = s$. In the above signature scheme, if an attacker $\mathcal{A}$ who can produce a valid signature $(m, \sigma_1, h, \sigma_2)$ without *Alice*'s private key d_A, then, according to the Forking Lemma, we can use $\mathcal{A}$ to obtain two signatures $(\sigma_1 = (x_1, U), h, \sigma_2 = s)$ and $(\sigma_1' = (x_1', U'), h', \sigma_2' = s')$ of an identical message m such that $\sigma_1 = \sigma_1'$, but $h \neq h'$. Note that $\sigma_1 = \sigma_1'$ implies that $k = k', u = u'$, and $r = r'$ which are used in the signing phase of both signatures. Thereafter, we can easily extract d_A which is the solution to the discrete logarithm problem of Q_A to the basis G:

$$\left.\begin{array}{l} s = (ku)^{-1}(h + d_A r) \\ s' = (ku)^{-1}(h' + d_A r) \end{array}\right\} \Rightarrow sd_A r - s'd_A r = s'h - sh',$$

which leads to $d_A = (r(s - s'))^{-1}(s'h - sh') \bmod n$. Since the discrete logarithm problem is believed to be hard, we found a contradiction. This ends the proof for the unforgeability of the signature on the signcrypted message of our scheme. □

To prove the confidentiality of the proposed scheme, we first consider the security in the outsider security model. That is, we first assume that the attacker cannot ask the private keys of the sender *Alice* and the receiver *Bob*. The confidentiality of the proposed signcryption scheme is IND-CCA2 in Flexible Un-signcryption Orale model (FUO-IND-CCA2) defined in [1].

Definition 7. FUO-IND-CCA2[1]: Let $\mathcal{SC} = (KG, SC, USC)$ be a signcryption scheme where KG is a public/private key generation algorithm, SC is a signcryption algorithm, and USC is a unsigncryption algorithm. Let $\mathcal{A}$ be an adversary that conduct adaptive chosen ciphertext attack. Then $\mathcal{SC}$ is FUO-IND-CCA2 secure if for all polynomial time adversary $\mathcal{A}$

$$Pr\left[\mathcal{A}^{\overset{SCO,}{USCO}}(guess, c, z) = b \;\middle|\; \begin{array}{l} (pk_A, sk_A) \leftarrow_R KG(1^k) \\ (pk_B, sk_B) \leftarrow_R KG(1^k) \\ (m_0, m_1, z) \leftarrow \mathcal{A}^{\overset{SCO,}{USCO}}(find, pk_A, pk_B) \\ b \leftarrow_R \{0,1\} \\ c \leftarrow SC_{sk_A, pk_B}(m_b) \end{array}\right] \leq \frac{1}{2} + neg(k).$$

where SCO is a signcryption oracle that signcrypt a message with access to sk_A and pk_B, $USCO$ is an unsigncryption oracle that unsigncrypts a ciphertext with access to sk_B. $neg()$ means a negligible function. In the random oracle model, $\mathcal{A}$ can also access to the random hash function H.

Baek et al. [1] showed that the modified Zheng's scheme (MZ Scheme) is FUO-IND-CCA2 secure if the GDH problem is hard and the symmetric encryption scheme in Zheng's scheme is IND-CPA secure. Based on this idea, Shin et al [12] proved that their SC-DSA+ provides the same security. Similar to the idea in [12], we reduce the security of our scheme to the security of SC-DSA+ in the elliptic curve setting and show that our scheme is FUO-IND-CCA2 secure.

Setting:
system-parameters: $params = \{q, G, n, E, D, H, hash\}$
private/public key-pair ($Alice$): (d_A, Q_A), $d_A \leftarrow_R \{1, \cdots, n-1\}$, $Q_A \leftarrow d_A G$
private/public key-pair (Bob) :(d_B, Q_B), $d_B \leftarrow_R \{1, \cdots, n-1\}$, $Q_B \leftarrow d_B G$

Signcrypting: to signcrypt a message $m \in \{0,1\}^*$:
(1) Pick $k \leftarrow_R \{1, \cdots, n-1\}$, and compute $K \leftarrow kQ_B$
(2) Compute $(K_{enc}, K_{mac}) \leftarrow hash(K)$, and $c \leftarrow E_{K_{enc}}(m)$
(3) Compute $R \leftarrow kG = (x_1, y_1) \in Z_q \times Z_q$
(4) Compute $r \leftarrow x_1 \bmod n$, $h \leftarrow H(m||bind_{A,B}||K_{mac})$, $s = k^{-1}(h + d_A r) \bmod n$
(5) Compute $e_1 \leftarrow h/s \bmod n$ and $e_2 \leftarrow r/s \bmod n$
(6) Out the signcrypted message $\xi \leftarrow (c, e_1, e_2)$

Unsigncrypting: Bob does the following steps:
(1) $R \leftarrow e_1 G + e_2 Q_A = (x_1, y_1)$
(2) Compute $K \leftarrow d_B R$, and $(K_{enc}, K_{mac}) \leftarrow hash(K)$
(3) Recover $m \leftarrow D_{K_{enc}}(c)$
(4) Compute $r \leftarrow x_1 \bmod n$, $s \leftarrow r/e_2 \bmod n$, $h \leftarrow H(m||bind_{A,B}||K_{mac})$
(5) Accept m if and only if $e_1 s \equiv h \bmod n$

Fig. 1. SC-DSA+ (Modified in Elliptic Curve Setting)

Theorem 1. *If the GDH problem is hard and the ECDSA is EUF-ACMA secure, moreover, if the symmetric encryption scheme provides indistinguishability against chosen plaintext attack (IND-CPA secure), then the proposed signcryption scheme is FUO-IND-CCA2 secure in the random oracle model.*

Proof: Shin et. al.'s scheme (SC-DSA+) (see Figure 1) is proved to be FUO-IND-CCA2 secure in the outsider security model. Since our scheme can be regarded as a modification of SC-DSA+, we reduce the security of our scheme to the security of SC-DSA+.

In our scheme, since $U = uR$ is used for the purpose of verifying the signature on the ciphertext and u is used for the forward security purpose while these properties do not exist in Shin et. al.'s SC-DSA+ scheme, we first modify our scheme, say Scheme II, into Scheme II' so as to omit the effect of U and u. Scheme II' is essentially the same as Scheme II but, for any message m, the signcrypted message is $\xi = (c, x_1, s, u)$ (cf. $\xi = (c, x_1, s)$ in Scheme II). The unsigncrypting phase is the same as Scheme II and there is no signature verification on the signcrypted message phase.

It is obvious that Scheme II' is a weak version of Scheme II. Therefore, any attacker who can break the FUO-IND-CCA2 security of Scheme II can also break the UFO-IND-CCA2 security of Scheme II'. Now, assume that there exists an attacker $\mathcal{F}$ who can break the FUO-IND-CCA2 security of Scheme II', then we show that the attacker $\mathcal{F}$ can also break the FUO-IND-CCA2 of SC-DSA+ Scheme (in the elliptic curve setting) in the same way. At first, notice that in SC-DSA+ scheme, a signcrypted message ξ on a message m is of the form $\xi = (c, e_1, e_2)$ while from (e_1, e_2), R and s can be computed publicly via the following computations:

$$R = e_1G + e_2Q_A = (x, y),$$
$$r = x \bmod n,$$
$$s = r/e_2 \bmod n.$$

Therefore, for a message m, no security flaw should occur concerning to the confidentiality (not concerning to the unforgeability in this proof) if we use (c, R, s) as the signcrypted message instead of using (c, e_1, e_2). Also note that, according to the unsigncrypting protocol, this change makes no difference form the receiver *Bob*'s viewpoint. In the remains of the proof, we use (c, R, s) as the signcrypted message on a message m in SC-DSA+ Scheme instead of using (c, e_1, e_2).

To prove the security of Scheme II', it is sufficient to show that the output of Scheme II' is (publicly) convertable to the output of SC-DSA+ Scheme and vice versa. Let $\xi = (c', x_1, s', u)$ be the output (ie. signcrypted message) of Scheme II', then we can derive a new output (c, R, s) using conversion algorithm $\mathcal{C}$:

```
$Algorithm\ \mathcal{C}(q, G, n, Q_A, Q_B, c', x_1, s', u)$
    $c||\alpha \overset{Parsing}{\longleftarrow} c'$
    $R \leftarrow PointDecom(E(F_q), x_1, \alpha)$
    $s \leftarrow us' \bmod n$
    $return\ (c, R, s)$
```

On the other hand, there exists an algorithm $\mathcal{C}^{-1}$ and $(c', x_1, s', u) = \mathcal{C}^{-1}(q, G, n, Q_A, Q_B, c, R, s)$.

```
$Algorithm\ \mathcal{C}^{-1}(q, G, n, Q_A, Q_B, c, R, s)$
    $(x_1, \alpha) \leftarrow PointComp(E(F_q), R)$
    $c' \leftarrow c||\alpha$
    $u \leftarrow_R \{1, \cdots, n-1\}$
    $s' \leftarrow su^{-1}$
    $return\ (c', x_1, s', u)$
```

Note that the conversion is done using publicly known information only. There remains one problem. That is, the message hash's input of Scheme II' is different from the message hash's input of SC-DSA+. In order to solve the difference of message hash's input, we first give the modified hash called *hash′* defined in [12] for SC-DSA+ which is defined as follows:

```
$hash'(m, bind_{A,B}, K, R)$
    $(K_{enc}, K_{mac}) \leftarrow hash(K)$
    $h \leftarrow hash(m||bind_{A,B}||K_{mac})$
    $return\ h$
```

hash′ is another representation of original *hash* used in SC-DSA+ since each operation of *hash′* already exists in SC-DSA+ signcrypt and unsigncrypt algorithm. Therefore, according to the proof of Theorem 3 in [12], it can be considered that *hash′* is used in SC-DSA+ instead of *hash*. The input R has no meaning here but it is required in order to make the input of *hash′* to be consistent with the input of our scheme.

Using the same idea, now we define a new message hash H' for our Scheme II' as follows:

```
$H'(m, bind_{A,B}, K, R)$
    $(x_1, \alpha) \leftarrow PointComp(E(F_q), R)$
    $(x_2, y_2) \overset{Parsing}{\longleftarrow} K$
    $\mathcal{K} \leftarrow H(x_2)$
    $(\mathcal{K}_{Enc}, u) \leftarrow T(\mathcal{K})$
    $\hat{c} \leftarrow E_{\mathcal{K}_{Enc}}(m)$
    $c \leftarrow \hat{c}||\alpha$
    $U \leftarrow uR$
    $h \leftarrow H(c||bind_{A,B}||x_1||U)$
    $return\ h$
```

Since each operation of H' is already existed in Scheme II' signcrypt and unsigncrypt algorithm, the same explanation says that H' is another representation of H and we can also use H' for computing h instead of using H in our scheme.

In Scheme II' and SC-DSA+, the output form one scheme is convertable to the other and vice versa. Moreover, the problem of having different message hash's input can be solved by using the modified hash functions *hash′* and H', respectively. This implies that both scheme have the same security and, by properly simulate the random oracles, breaking one scheme means breaking the other.

From Scheme II, we can construct Scheme II' which is a weak version of Scheme II. On the other hand, Scheme II' has the same security with SC-DSA+ [2]. Therefore, if there exists an adversary $\mathcal{A}$ who can break the FUO-IND-CCA2 security of our Scheme II, then, by properly simulates our schemes using SC-DSA+ scheme, we can utilize $\mathcal{A}$ to break the FUO-IND-CCA2 security of SC-DSA+ scheme. This ends the proof. □

Forward security: Theorem 1 is proved in the outsider security model. To show the insider security, that is, to show the forward security of the proposed schemes,

[2] In case if the signcrypted message is of the form (c, R, s) since we have shown that no security flaw concerning to the confidentiality will occur by this modification.

we can see that in Scheme II, the encryption key is $kQ_B = d_B R$ while k is a random number and $kQ_B = d_B R$ is irrelevant to the sender's private key. Since d_B is the receiver's private key which is unknown to the third party [3], the only way to find the encryption key is to compute k from $s = (ku)^{-1}(h + d_A r) \bmod n$ in Scheme II. In our schem (Scheme II), even if h can be computed when U opened, u is still unknown so k cannot be computed from $s = (ku)^{-1}(h + d_A r) \bmod n$ even when d_A is known (ie., k cannot be uniquely determined from the equation with two unknowns). Therefore, a sender's private key has no concern to the confidentiality of our signcryption so we conclude that our scheme is forward secure.

References

1. Baek, J., Steinfeld, R., Zheng, Y.: Formal proofs for the security of signcryption. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 81–98. Springer, Heidelberg (2002)
2. Bao, F., Deng, R.H.: A signcryption scheme with signature directly verifiable by public key. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 55–59. Springer, Heidelberg (1998)
3. Barreto, P.S.L.M., Libert, B., McCullagh, N., Quisquater, J.: Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 515–532. Springer, Heidelberg (2005)
4. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
5. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Advances in cryptology –CRYPTO 2001. Lecture Notes in Comput. Sci., vol. 2248, pp. 514–532 (2001)
6. Chow, S.S.M., Yiu, S.M., Hui, C.K., Chow, K.P.: Efficient forward and provably secure ID-based signcryption scheme with public verifiability and public ciphertext authenticity. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 352–369. Springer, Heidelberg (2004)
7. Gamage, C., Leiwo, J., Zheng, Y.: Encrypted message authentication by firewalls. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 69–81. Springer, Heidelberg (1999)
8. NIST (National Institute for Standard and Technology), Digital Signature Standard (DSS), FIPS PUB, 186 (1994)
9. Pohlig, S., Hellman, M.: An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. IEEE Transactions on Information Theory 24, 106–110 (1978)
10. Pointcheval, D., Stern, J.: Security argrments for digital signatures and blind signatures. Journal of Cryptology 13(3), 361–396 (2000)
11. Pollard, J.: Monte Carlo methods for index computation mod p. Mathematics of Computation 32, 918–924 (1978)

[3] If an attacker who can find the private key d_b with regard to Q_B, then it is easy to construct an algorithm using the attacker to break the discrete logarithm problem.

12. Shin, J., Lee, K., Shim, K.: New DSA-verifiable signcryption schemes. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 35–47. Springer, Heidelberg (2003)
13. Stinson, D.R.: Cryptography: Theory and Practice. CRC Press, Boca Raton (1995)
14. TTAS, Digital Signature Mechanism with Appendix - Part 2: Certificate-Based Digital Signature Algorithm (KCDSA), TTSA.KO-12.001/R1 (1998)
15. Vanstone, S.: Responses to NIST's proposal. Communications of the ACM 35, 50–52 (1992)
16. Yum, D.H., Lee, P.J.: New signcryption schemes based on KCDSA. In: Kim, K.-c. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 305–317. Springer, Heidelberg (2002)
17. Zheng, Y.: Digital signcryption or how to achieve cost (signature & encryption << cost (signature) + cost (encryption). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 165–179. Springer, Heidelberg (1997)

Provably Secure Identity-Based Undeniable Signatures with Selective and Universal Convertibility[⋆]

Wei Wu, Yi Mu, Willy Susilo, and Xinyi Huang

Centre for Computer and Information Security Research
School of Computer Science & Software Engineering
University of Wollongong, Australia
{wei,ymu,wsusilo,xh068}@uow.edu.au

Abstract. In this paper, we present *the first* concrete example of identity-based undeniable signature *with* selective and universal convertibility, where the signer can release a selective proof to make a single undeniable signature publicly verifiable, or publish a universal proof to convert all his undeniable signatures into ordinary digital signatures. Furthermore, we also formalized the security models of identity-based convertible undeniable signatures. The new models capture more essence of the property "convertibility" of identity-based undeniable signatures, compared with other known security models. Our scheme can be regarded as an improvement of Libert and Quisquater's identity-based undeniable signature scheme published in CT-RSA 2004. The security of our scheme is formally proven in the random oracle model under some well-known complexity assumptions. Its unforgeability relies on the hardness of the Computational Diffie-Hellman problem, while the invisibility (and anonymity) is based on the hardness of Decisional Bilinear Diffie-Hellman problem.

Keywords: Undeniable Signatures, Convertible, Identity-based, Provable Security.

1 Introduction

Undeniable signature is a notion introduced by Chaum and van Antwerpen in Cypto'89 [4] to allow the verifier to attest the validity or invalidity of a signature *only* with the help of the signer. Formally, the validity or invalidity of an undeniable signature can only be verified via the Confirmation/Disavowal protocol with the help of the signer. They are useful in the situations where the validity of a signature must not be publicly verifiable. For example, a software vendor might want to embed signatures into his products and allow only those customers who have paid to check the authenticity of these products. If the vendor actually signed the message, he must be able to convince the customer of this fact using a confirmation protocol. If he did not sign it, he

⋆ Supported by ARC Discovery Grant DP0557493 and DP0663306.

must also be able to convince the customer that he is not the signer via a disavowal protocol. These proofs have to be *non-transferable*: once a verifier is convinced that the vendor did or did not sign a message, he should be unable to transmit the proof to another third party. Since the introduction of undeniable signatures, there have been several proposed schemes in the literature [1,7,8,9,10,11,15,16,18,22].

The concept of *convertible* undeniable signatures was introduced by Boyar, Chaum, Damgård and Pedersen [1] in Crypto'90. This concept offers more flexibility than its original version in the sense that the signer has the ability to convert one or more undeniable signatures into publicly verifiable. "Convert" in the undeniable signatures has two types: **Selectively Convert** and **Universally Convert**. **Selectively Convert** refers to the case that the signer can make one of his/her undeniable signatures publicly verifiable by releasing a selective proof. Then, one can verify the validity of a converted undeniable signature using the corresponding selective proof and signer's public key. However, the validity of other undeniable signatures that have not been converted still remains unknown and can only be verified via the confirmation/disavowal protocol with the help of the signer. The signer can also run the **Universally Convert** algorithm to generate a universal proof which can convert all his undeniable signatures into publicly verifiable ones. Thus, one can check the validity of any undeniable signature without the help of the signer. A concrete application of convertible undeniable signatures is the problem of keeping digital records of political decisions [7]. Those decisions are sensitive when they are generated, but usually become publicly accessible after some years. This can be solved by convertible undeniable signatures: the signer could release the selective/universal proof whenever required, or give them initially to a trusted third party who will release it later.

Most of the known undeniable signature schemes are designed in the traditional public key system where a public key infrastructure (PKI) is requires to certify public keys for users. PKI has a number of drawbacks since it requires a trusted third party. Identity-based (or ID-based) cryptography, as proposed by Shamir in [21], was introduced to provide an alternative to public key systems. In the new setting, a user's public key is the identity of the corresponding user. In the real world, an identity can be email address, IP address or telephone number, etc. Therefore, the need of certification can be eliminated. There is a full-trusted third party, Private Key Generator (PKG), in the ID-based system which generates all users' private keys. The PKG first publishes a "master" public key, and keeps the corresponding master secret key for himself. To obtain a private key, one should contact PKG, which uses the master secret key to generate the corresponding private key and sends it to the user by a secure channel. Since its introduction in [21], many identity based schemes have been proposed and the most notable one is the identity-based encryption scheme proposed by Boneh and Franklin in [2] that takes advantage of the properties of suitable bilinear maps (the Weil or Tate pairing) over supersingular elliptic curves.

1.1 Previous Works

In [14], Han, Yeung and Wang proposed the first ID-based undeniable signature scheme[1]. However, Zhang, Safavi-Naini and Susilo showed that this scheme is insecure against the denial attack and the forge attack [22]. Chow introduced a new scheme [5] to fix the flaw in Han-Yeung-Wang's scheme but unfortunately, there is no a formal proof provided, and therefore the security of the scheme is questionable. A new construction of the ID-based undeniable signature was given by Libert and Quisquater in CT-RSA 2004 [15]. Its security is proven in the random oracle model under the assumption that both Bilinear Computational Diffie-Hellman problem and Decisional Bilinear Diffie-Hellman problem are hard. Libert and Quisquater also mentioned in [15] that their scheme has the property of selectively convertibility. In addition to those concrete constructions, Galindo, Herranz and Kiltz [13] showed a generic way to obtain an ID-based undeniable signature scheme from a PKI-based undeniable signature scheme. The following table summarizes the known ID-based undeniable signature schemes in the literature.

Table 1. ID-based Undeniable Signature Scheme in the Literature

Scheme	**S-Convert**	**U-Convert**	**Security**
Han-Yeung-Wang's [14]			broken[22]
Chow's [5]	✓		No Proof Provided
Libert and Quisquater's [15]	✓		Pairing Related Assumptions
Galindo-Herranz-Kiltz's [13]	✓	✓	Generic Construction

1.2 Our Motivations

In CT-RSA 2004, the security models of ID-based undeniable signatures were formalized by Libert and Quisquater [15]. However, the security of convertible ID-based undeniable signature has not been formally defined yet. In an undeniable signature with convertibility, the universal proof or some selective proofs could be published by the signer, which can help others to check the validity of the undeniable signatures by themselves. On the other hand, those proofs also "leak" some additional information to the adversaries and might help the later to break the schemes. Taking Boyar-Chaum-Damgård-Pedersen's scheme [1] as an example, their scheme is secure if the signer does not publish the universal proof. As long as the universal proof is published, a key-only adversary can forge a valid signature on any message [18]. Thus, the security models in the ID-based undeniable signature cannot precisely define the security of its convertible version. It is worthwhile to formally define the security of ID-based undeniable signatures with selectively and universally convertibility.

[1] The authors claimed that the scheme is a confirmer signature scheme, but it is actually an undeniable signature scheme which has been pointed out by Zhang, Safavi-Naini and Susilo in [22].

Additionally, we are also motivated to construct an ID-based universally-convertible undeniable signature scheme whose security can be formally proved with some well-studied security problems. According to the summary given in Table 1, the only way to obtain an ID-based undeniable signature scheme with universal convertibility is the generic construction given in [13]. To date, there exists no concrete scheme which satisfies this property. Galindo-Herranz-Kiltz's construction uses the PKI-based undeniable signature scheme as the building block. That is, if the underlying PKI-based scheme is universally convertible, then its ID-based version will have this property as well. Therefore, the security of the resulting ID-based scheme also relies on its PKI-based version. The following table shows the known PKI-based undeniable signature schemes with universal convertibility:

Table 2. PKI-based Undeniable Signature Schemes with Universal Convertibility

Scheme	**Security**
Boyar-Chaum-Damgård-Pedersen's [1]	broken[18]
Damgård-Pedersen's [7]	DL related assumptions
Michels-Petersen-Horster's [18]	no proof provided
Michels-Stadler's [19]	sketchy proof provided
Gennaro-Rabin-Krawczyk's [11]	RSA related assumptions
Miyazaki's [17]	sketchy proof provided
Laguillaumie-Vergnaud's [16]	pairing related assumptions

As we can see in Table 2, Boyar-Chaum-Damgård-Pedersen's scheme has been broken, and can not be used to construct the ID-based scheme. Other schemes [17,18,19] only have sketchy proofs, and therefore the security of their schemes are only heuristic. There are three schemes [7,11,16] which have formal proofs. However, the proofs in the first two schemes [7,11] only analyze the security of the basic undeniable signature schemes. Here, the "basic" refers to the scheme where the property convertibility is not considered in the security analysis and the adversary is not allowed to obtain any selective or universal proof. It is only mentioned in [7,11] that their basic schemes could be extended to have the property convertibility, without proving whether the schemes in [7,11] are still secure when the selective and universal proofs are published. The only known universally convertible undeniable signature scheme with formally proof was proposed by Laguillaumie and Vergnaud [16]. The unforgeability of their scheme is based on the hardness of a well known problem: Computational Diffie-Hellman problem. However, the invisibility of their scheme is based on $(\ell, 1)$-xyz-**DCAA** assumption. As pointed out by the authors in [16], this assumption is a non-standard decisional assumption. Therefore, there is no suitable PKI-based universally convertible undeniable signature scheme which can serve as the building block in Galindo-Herranz-Kiltz's generic construction [13] and obtain a provably secure ID-based universally-convertible undeniable signature scheme under well-known security assumptions.

1.3 Contributions of This Paper

In this paper, we first formalize the security models of the identity-based convertible undeniable signatures. We clearly define the properties that an identity-based convertible undeniable signature scheme should satisfy. These properties are defined by the game between the oracles and the adversaries defined in our paper. Compared with the other model in [15], our new model reflects more essence of the identity-based undeniable signature with convertibility.

We then give the *first* concrete example of both selectively and universally-convertible undeniable signature in the identity-based system. In the new scheme, the signatures can only be verified with the help of the signer when they are generated. The signer himself can decide when and how to convert his signatures into universally verifiable signatures. He can release a selective proof to make a single undeniable signature publicly verifiable, or publish a universal proof to convert all his undeniable signatures into ordinary signatures. The security of the new scheme is formally analyzed in the random oracle model. Its unforgeability relies on the hardness of the Computational Diffie-Hellman problem, while the invisibility (and anonymity) is based on the hardness of the Decisional Bilinear Diffie-Hellman problem. All the security assumptions have been widely accepted and used in the cryptology research field.

1.4 Organization

This paper is organized as follows. In the next section, we will describe the definition and security models of ID-based convertible undeniable signature. Our concrete ID-based convertible undeniable signature scheme is presented in Section 3. In the same section, we also give the formal security analysis of our scheme. Finally, Section 4 concludes this paper.

2 Definitions and Security Models of Identity-Based Convertible Undeniable Signature

2.1 Outline of Identity-Based Convertible Undeniable Signatures

Our ID-based convertible undeniable signature scheme consists of the following algorithms:

Common Parameter Generation: A probabilistic algorithm that on input a security parameter k, outputs the system's master key s and a string cp which denotes the common scheme parameters including the message space $\mathcal{M}$ and the signature space $\mathcal{S}$. cp is shared among all the users in the system.

Key Extraction: A deterministic algorithm that on input system's parameter cp, the master secret key s and an identity ID_i of the user, outputs the secret key SK_{ID_i} of the user.

Undeniable Sign: A probabilistic (or deterministic) algorithm that on input the common parameter cp, the signer S's secret key SK_{ID_s} and the message m to be signed, outputs S's convertible undeniable signature σ.

Undeniable Verify: A deterministic algorithm that on input the common parameter *cp*, S's secret key SK_{ID_s} and a message-signature pair (m, σ), outputs 1 if it is a valid message-signature pair. Otherwise, outputs 0.

Confirmation Protocol: An interactive (or non-interactive) algorithm that on input the common parameter *cp*, S's secret key SK_{ID_s}, (possibly) a Verifier V's identity ID_v and a message-signature pair (m, σ), outputs a *non-transferable* transcript $Trans$ which can convince V about the validity of σ.

Disavowal Protocol: An interactive (non-interactive) algorithm that on input the common parameter *cp*, S's secret key SK_{ID_s}, (possibly) V's identity ID_v and a message-signature pair (m, σ), outputs a *non-transferable* transcript $Trans$ that shows the invalidity of σ to V.

Selectively Convert: A probabilistic (or deterministic) algorithm that on input the common parameter *cp*, S's secret key SK_{ID_s} and the message-signature pair (m, σ), outputs a selective proof $\Pi^{ID_s}_{(m,\sigma)}$.

Selectively Verify: A deterministic algorithm that on input the common parameter *cp*, S's identity ID_s, message-signature pair (m, σ) and its selective proof $\Pi^{ID_s}_{(m,\sigma)}$, outputs the verification decision $d \in \{Acc, Rej\}$.

Universally Convert: A deterministic algorithm that on input the common parameter *cp* and S's secret key SK_{ID_s}, outputs the universal proof Π_{ID_s}.

Universally Verify: A deterministic algorithm that on input the common parameter *cp*, S's identity ID_s, *any* message-signature pair (m, σ) and the universal proof Π_{ID_s}, outputs the verification decision $d \in \{Acc, Rej\}$.

We allow the adversary to access the following oracles and submit their queries adaptively:

– $\mathcal{O}^{Sign}$: On an undeniable sign query (m, ID_s), this oracle runs the **Undeniable Sign** algorithm to generate the undeniable signature σ and returns it to the adversary.

– $\mathcal{O}^{Ver}$: On a verify query (m, σ, ID_s) (and possibly ID_v), this oracle first runs the **Undeniable Verify** algorithm to decide whether (m, σ) is a valid message-signature pair under the public key ID_s and outputs the decision result $d \in \{0, 1\}$. According to the decision result d, the oracle responds based on whether a passive attack or an active/concurrent attack is mounted.
  1. Active/Concurrent attack: The oracle $\mathcal{O}^{Ver}$ executes the confirmation/disavowal protocol with adversary (acting as a cheating verifier) depending on the verification result $d \in \{0, 1\}$.
  2. Passive attack: The oracle $\mathcal{O}^{Ver}$ returns a transcript of confirmation protocol if $d = 1$. Otherwise, a transcript of disavowal protocol is returned.

– $\mathcal{O}^{SCon}$: On a selectively convert query (m, σ, ID_s), this oracle runs the **Selectively Convert** algorithm to generate the selective proof $\Pi^{ID_s}_{(m,\sigma)}$ and returns it to the adversary.

– $\mathcal{O}^{UCon}$: On a universally convert query ID_s, this oracle runs **Universally Convert** algorithm to generate the universal proof Π_{ID_s} and returns it to the adversary.

– $\mathcal{O}^{Corrupt}$: On a corruption query ID, this oracle returns the corresponding secret key to the adversary.

The security of our ID-based convertible undeniable signature will be defined using the game between these oracles and the adversary. We will focus on the case when the Confirmation/Disavowal protocol is non-interactive for the rest of the paper[2]. The security when confirmation/disavowal protocol is interactive can be defined analogously.

Notation: In the definition of the security models, we will use the notation:

$$\{Q_1, Q_2, \cdots, Q_n\} \nrightarrow \{\mathcal{O}^1, \mathcal{O}^2, \cdots, \mathcal{O}^n\}$$

which denotes that "No query $Q \in \{Q_1, Q_2, \cdots, Q_n\}$ is allowed to submit to any oracle $\mathcal{O} \in \{\mathcal{O}^1, \mathcal{O}^2, \cdots, \mathcal{O}^n\}$. Using this notation, the security models can be described by just pointing out its aim while hiding all details.

2.2 Unforgeability

As the security model defined in [15], the forger $\mathcal{F}$ has access to the oracle $\mathcal{O}^{Sign}$ and $\mathcal{O}^{Ver}$. In addition, we also allow the adversary to submit queries to $\mathcal{O}^{SCon}$ and $\mathcal{O}^{UCon}$ adaptively. This is to ensure that the knowledge of the selective or universal proof does not help the adversary to forge a new valid message signature pair. The unforgeability of our ID-based convertible undeniable signature is formally defined as:

Queries: $\mathcal{F}$ can submit queries to all the oracles defined in Section 2.1
Output: (m^*, σ^*, ID^*).
Restrictions: (1). $(m^*, ID^*) \nrightarrow \mathcal{O}^{Sign}$ and $ID^* \nrightarrow \mathcal{O}^{Corrupt}$ (2). (m^*, σ^*) is valid under the identity ID^*.

The success probability of an adaptively chosen message and chosen identity forger $\mathcal{F}$ wins the above game is defined as $\mathsf{Succ}\ \mathcal{F}^{CMA,\ CIDA}_{EUF,\ IDCUS}$.

Definition 1. *We say an identity based convertible undeniable signature scheme is unforgeable against a $(t, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$ forger $\mathcal{F}^{CMA,\ CIDA}_{EUF,\ IDCUS}$, if $\mathcal{F}^{CMA,\ CIDA}_{EUF,\ IDCUS}$ runs in time at most t, makes at most q_{US} queries to $\mathcal{O}^{Sign}$, q_V queries to $\mathcal{O}^{Ver}$, q_{SC} queries to $\mathcal{O}^{SCon}$, q_{UC} queries to the $\mathcal{O}^{UCon}$, q_C queries to $\mathcal{O}^{Corrupt}$ and $\mathsf{Succ}\ \mathcal{F}^{CMA,\ CIDA}_{EUF,\ IDCUS}$ is negligible.*

2.3 Invisibility

Given a message-signature pair (m, σ) and the identity ID_s of the signer S, the invisibility property requires that it is difficult to decide whether it is a valid message-signature pair without the help of the signer, the knowledge of selective

[2] If the Confirmation/Disavowal protocol is non-interactive, there is no need to consider the active/concurrent attack [20].

proof $\Pi_{(m,\sigma)}^{ID_s}$ or universal proof Π_{ID_s} . It is defined using the game between the oracles in Section 2.1 and an adaptively chosen message and chosen identity distinguisher $\mathcal{D}_{INV,\ IDCUS}^{CMA,\ CIDA}$. The whole game is divided into two phases.

Phase 1: The distinguisher $\mathcal{D}$ can adaptively access all the Oracles.
Output: (m^*, ID_s^*)
Restrictions: (1). $ID_s^* \nrightarrow \{\mathcal{O}^{UCon}, \mathcal{O}^{Corrupt}\}$ (2). $(m^*, ID_s^*) \nrightarrow \mathcal{O}^{Sign}$.

As a response, $\mathcal{O}^{Sign}$ chooses a random bit $\gamma \in \{0,1\}$. If $\gamma = 1$, this oracle will run **Undeniable Sign** algorithm to generate the undeniable signature σ and sets $\sigma^* = \sigma$. Otherwise, this oracle chooses a random element σ^* in the signature space $\mathcal{S}$. Then, it returns the challenging signature σ^* to $\mathcal{D}$.

Phase 2: The distinguisher $\mathcal{D}$ can still access all the oracles adaptively.
Restrictions: (1). $ID_s^* \nrightarrow \{\mathcal{O}^{UCon}, \mathcal{O}^{Corrupt}\}$. (2). $(m^*, ID_s^*) \nrightarrow \mathcal{O}^{Sign}$ (3). $(m^*, \sigma^*, ID_s^*) \nrightarrow \{\mathcal{O}^{Ver}, \mathcal{O}^{SCon}\}$.
Output: $\gamma' \in \{0,1\}$.

The distinguisher wins the game if $\gamma = \gamma'$. The advantage of the distinguisher $\mathcal{D}_{INV,\ IDCUS}^{CMA,\ CIDA}$ has in the above game is defined as $\mathsf{Adv}\ \mathcal{D}_{INV,\ IDCUS}^{CMA,\ CIDA} = |\Pr[\gamma = \gamma'] - 1/2|$.

Definition 2. *We say an identity based convertible undeniable signature scheme is invisible against a $(t, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$-distinguisher $\mathcal{D}_{INV,\ IDCUS}^{CMA,\ CIDA}$, if $\mathcal{D}_{INV,\ IDCUS}^{CMA,\ CIDA}$ runs in time at most t, makes at most q_{US} queries to $\mathcal{O}^{Sign}$, q_V queries to $\mathcal{O}^{Ver}$, q_{SC} queries to $\mathcal{O}^{SCon}$, q_{UC} queries to the $\mathcal{O}^{UCon}$, q_C queries to $\mathcal{O}^{Corrupt}$ and $\mathsf{Adv}\ \mathcal{D}_{INV,\ IDCUS}^{CMA,\ CIDA}$ is negligible.*

2.4 Anonymity

Essentially, the anonymity property requires that given a valid message-signature pair (m, σ) and two possible signers' identities ID_0, ID_1, it is computationally impossible to decide who generated this signature. This property was introduced to the undeniable signature by Galbraith and Mao [9]. The authors suggested that anonymity is the most relevant security property for undeniable signatures. In our ID-based convertible undeniable signatures, this property is defined using the game between the oracles in Section 2.1 and an adaptively chosen message and chosen identity distinguisher $\mathcal{D}_{ANONY,\ IDCUS}^{CMA,\ CIDA}$. Similarly to the models defined in Section 2.3, the whole game is divided into two phases.

Phase 1: The distinguisher $\mathcal{D}$ can adaptively access all the Oracles.
Output: (m^*, ID_0^*, ID_1^*).
Restrictions:
(1). $\{ID_0^*, ID_1^*\} \nrightarrow \{\mathcal{O}^{UCon}, \mathcal{O}^{Corrupt}\}$. (2). $\{(m^*, ID_0^*), (m^*, ID_1^*)\} \nrightarrow \mathcal{O}^{Sign}$.

As a response, $\mathcal{O}^{Sign}$ chooses a random bit $\gamma \in \{0,1\}$. If $\gamma = 0$, this oracle will run **Undeniable Sign** algorithm to generate the undeniable signature σ under the secret key $SK_{ID_0}^*$. Otherwise, it will run **Undeniable Sign** algorithm to generate the undeniable signature σ under the secret key $SK_{ID_1}^*$. Then, it returns the challenging signature to $\mathcal{D}$.

Phase 2: The distinguisher $\mathcal{D}$ can adaptively access all the Oracles.
Restrictions:
(1). $\{ID_0^*, ID_1^*\} \nrightarrow \{\mathcal{O}^{UCon}, \mathcal{O}^{Corrupt}\}$. (2). $\{(m^*, ID_0^*), (m^*, ID_1^*)\} \nrightarrow \mathcal{O}^{Sign}$.
(3). $\{(m^*, \sigma^*, ID_0^*), (m^*, \sigma^*, ID_1^*)\} \nrightarrow \{\mathcal{O}^{SCon}, \mathcal{O}^{Ver}\}$.
Output: $\gamma' \in \{0, 1\}$.

The adversary wins the game if $\gamma = \gamma'$. The advantage of the distinguisher $\mathcal{D}_{ANONY,\ IDCUS}^{CMA,\ CIDA}$ has in the above game is defined as $\mathsf{Adv}\ \mathcal{D}_{ANONY,\ IDCUS}^{CMA,\ CIDA} = |\Pr[\gamma' = \gamma] - 1/2|$.

Definition 3. *We say an identity based convertible undeniable signature scheme is anonymous against a* $(t, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$*-distinguisher* $\mathcal{D}_{ANONY,\ IDCUS}^{CMA,\ CIDA}$*, if* $\mathcal{D}_{ANONY,\ IDCUS}^{CMA,\ CIDA}$ *runs in time at most* t*, makes at most* q_{US} *queries to* $\mathcal{O}^{Sign}$*,* q_V *queries to* $\mathcal{O}^{Ver}$*,* q_{SC} *queries to* $\mathcal{O}^{SCon}$*,* q_{UC} *queries to the* $\mathcal{O}^{UCon}$*,* q_C *queries to* $\mathcal{O}^{Corrupt}$ *and* $\mathsf{Adv}\ \mathcal{D}_{ANONY,\ IDCUS}^{CMA,\ CIDA}$ *is negligible.*

According to the analysis in [9] and [15], the property anonymity and invisibility are equivalent in the ID-based system.

2.5 Non-impersonation

The Non-Impersonation requires that only the signer has the ability to convince or disavowal an undeniable signature. It can be further divided by the following three types:

1. **Type$_1$: Impersonation of Selectively Convert Algorithm**
   In this case, the impersonator $\mathcal{I}$ can adaptively access all the Oracles. After all the queries, $\mathcal{I}$ outputs a valid selective proof $\Pi_{(m^*, \sigma^*)}^{ID_s^*}$ with the restrictions that $(m^*, \sigma^*, ID_s^*) \nrightarrow \mathcal{O}^{SCon}$ and $ID_s^* \nrightarrow \{\mathcal{O}^{UCon}, \mathcal{O}^{Corrupt}\}$.
2. **Type$_2$: Impersonation of Universally Convert algorithm**
   In this case, the impersonator $\mathcal{I}$ can adaptively access all the Oracles. After all the queries, $\mathcal{I}$ outputs a universal proof $\Pi_{ID_s^*}$ with the restriction that $ID_s^* \nrightarrow \{\mathcal{O}^{UCon}, \mathcal{O}^{Corrupt}\}$.
3. **Type$_3$: Impersonation of Confirmation/Disavowal protocol**
   In this case, the impersonator $\mathcal{I}$ can adaptively access all the Oracles. After all the queries, $\mathcal{I}$ can output $(m^*, \sigma^*, Trans^*, ID_s^*, ID_v^*)$ such that $Trans^*$ can confirm or deny the undeniable signature σ^*. The only restrictions are that $\{ID_s^*, ID_v^*\} \nrightarrow \mathcal{O}^{Corrupt}$ and $(m^*, \sigma^*, ID_s^*, ID_v^*) \nrightarrow \mathcal{O}^{Ver}$.

The success probability of an adaptively chosen message and chosen identity impersonator $\mathcal{I}$ wins the above game is defined as $\mathsf{Succ}\ \mathcal{I}_{Type_i,\ IDCUS}^{CMA,\ CIDA}, i \in \{1, 2, 3\}$.

Definition 4. *We say an identity based convertible undeniable signature scheme is non-impersonated against a* $(t, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$*-adversary* $\mathcal{I}_{IDCUS}^{CMA,\ CIDA}$*, if* $\mathcal{I}_{IDCUS}^{CMA,\ CIDA}$ *runs in time at most* t*, makes at most* q_{US} *queries to* $\mathcal{O}^{Sign}$*,* q_V *queries to* $\mathcal{O}^{Ver}$*,* q_{SC} *queries to* $\mathcal{O}^{SCon}$*,* q_{UC} *queries to the* $\mathcal{O}^{UCon}$*,* q_C *queries to* $\mathcal{O}^{Corrupt}$ *and* $\mathsf{Succ}\ \mathcal{I}_{Type_i,\ IDCUS}^{CMA,\ CIDA}(i = 1, 2, 3)$ *is negligible.*

3 Proposed Scheme

In this section, we will first review some fundamental backgrounds which are related to our schemes. Then we will describe our ID-based convertible undeniable signature with a security and efficiency analysis.

3.1 Bilinear Maps

Let $\mathbb{G}_1$ and $\mathbb{G}_T$ be two groups of prime order p and let g be a generator of $\mathbb{G}_1$. The map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is said to be an admissible bilinear map if the following three conditions hold true: (i) e is bilinear, i.e. $e(aP, bP) = e(P, P)^{ab}$ for all $a, b \in \mathbb{Z}_p$. (ii) e is non-degenerate, i.e. $e(P, P) \neq 1_{\mathbb{G}_T}$. (iii) e is efficiently computable. We say that $(\mathbb{G}_1, \mathbb{G}_T)$ are bilinear groups if there exists the bilinear maps $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ as above, and e, and the group action in $\mathbb{G}_1$ and $\mathbb{G}_T$ can be computed efficiently. See [3] for more details on the construction of such pairings.

Computational Diffie-Hellman Problem (CDH) in $\mathbb{G}_1$: Given a triple $\mathbb{G}_1$ elements (P, aP, bP), find the element $C = abP$.

Decisional Bilinear Diffie-Hellman Problem (DBDH) in $\mathbb{G}_1$: Given a 5-triple $\mathbb{G}_1$ elements (P, aP, bP, cP, H), decide whether $H = e(P, P)^{abc}$.

3.2 Concrete Scheme

In this section, we will describe our construction of ID-based convertible undeniable signature scheme. It consists of the following algorithms:

Common Parameter Generation: Let $(\mathbb{G}_1, \mathbb{G}_T)$ be the bilinear groups and e be the pairing $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. The order of $\mathbb{G}_1$ is p where $p \geq 2^k$, k is the system's security number. The generator of $\mathbb{G}_1$ is P. The Private Key Generator (PKG) chooses a random number $s \in \mathbb{Z}_p$ which is set as the master secret key. The system's master public key is set as $P_{pub} = sP$. There are four cryptographic hash functions: $H_1, H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1$, $H_3 : \mathbb{G}_1 \times \mathbb{G}_T \rightarrow \mathbb{G}_1$ and $H_4 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$.

Key Extraction: In our scheme, each user ID has two private keys SK_{ID} and VK_{ID}. SK_{ID} can be regarded as the signing key and VK_{ID} is the verifying key. At the beginning, both of them are kept as secrets but VK_{ID} can be published later when ID wants to convert all his undeniable signatures into publicly verifiable ones. Similarly to the traditional ID-based system, both two keys are generated by PKG where $SK_{ID} = sH_1(ID)$ and $VK_{ID} = sH_1(ID \| \text{Undeniable})$.

Undeniable Sign: The signer S generates a convertible undeniable signature for message m as follows:

– Uses his private key VK_{ID_s} to compute $U = e(VK_{ID_s}, H_2(m))$,

– Chooses a random number $v \in \mathbb{Z}_p$ and sets $V = vP$, $W = SK_{ID_s} + vH_3(U\|V)$.

The undeniable signature of the message m is set to be $\sigma = (U, V, W)$.

Undeniable Verify: For a message/signature pair (m, σ) where σ is (U, V, W),

– The signer S firstly checks whether $e(W, P) \overset{?}{=} e(H_1(ID_s), P_{pub})e(H_3(U\| V), V)$. If the equation does not hold, output 0. Otherwise,

– S checks whether $U = e(H_2(m), VK_{ID_s})$. If the equation holds, output 1. Otherwise, output 0.

Confirmation Protocol: Given the verifier V's identity ID_v and a valid message-signature pair (m, σ) to be confirmed, the signer S will use the designated verifier techniques [12] to prove its validity.

– S chooses $r, c_v \in_R \mathbb{Z}_p$, $C_v \in_R \mathbb{G}_1$, then computes
   1. $R_1 = e(P, P)^r$, $R_2 = e(P, H_2(m))^r$ and $R_3 = e(C_v, P)e(H_1(ID_v), P_{pub})^{c_v}$,
   2. $c = H_4(ID_s\|ID_v\|R_1\|R_2\|R_3\|m\|\sigma), c_s = c - c_v \pmod p$, $C_s = rP - c_sVK_{ID_s}$.

  S then sends (c_s, c_v, C_s, C_v) to the verifier V as the transcript of **Confirmation Protocol**.

– On receiving (c_s, c_v, C_s, C_v) and the message-signature pair (m, σ) where σ is (U, V, W), the verifier
   1. checks whether $e(W, P) \overset{?}{=} e(H_1(ID_s), P_{pub})e(H_3(U\|V), V)$. If the equality holds, then
   2. computes $R'_1 = e(C_s, P)e(H_1(ID_s\|\text{Undeniable}), P_{pub})^{c_s}$, $R'_2 = e(C_s, H_2(m))U^{c_s}$, $R'_3 = e(C_v, P)\ e(H_1(ID_v), P_{pub})^{c_v}$ and
   3. checks whether $c_s + c_v \overset{?}{=} H_4(ID_s\|ID_v\|R'_1\|R'_2\|R'_3\|m\|\sigma)$. If the equality holds as well, V will accept σ as a valid undeniable signature of message m.

Disavowal Protocol: Given the verifier V's identity ID_v and a message-signature pair (m, σ) where σ is (U, V, W) to be denied,

– signer S firstly chooses $z, d_v, \alpha, \beta \in_R \mathbb{Z}_p$, $D_v \in_R \mathbb{G}_1$ and computes:
   1. $A = [\frac{e(VK_{ID_s}, H_2(m))}{U}]^z \neq 1$, $B = \frac{[e(P, H_2(m))]^\alpha}{U^\beta}$,
      $C = \frac{e(P,P)^\alpha}{e(H_1(ID_s\|\text{Undeniable}), P_{pub})^\beta}$ and $D = e(D_v, P)e(H_1(ID_v), P_{pub})^{d_v}$,
   2. $d = H_4(ID_s\|ID_v\|A\|B\|C\|D\|m\|\sigma)$, $d_s = d - d_v \pmod p$,
   3. $D_s = \alpha P + d_s zVK_{ID_s}$, $\widehat{d_s} = \beta + d_s z \pmod p$,

  S then sends $(A, d_s, d_v, \widehat{d_s}, D_s, D_v)$ to the verifier V.

– On receiving $(A, d_s, d_v, \widehat{d_s}, D_s, D_v)$ and the message-signature pair (m, σ) where $\sigma = (U, V, W)$, the verifier V
   1. checks whether $e(W, P) \overset{?}{=} e(H_1(ID_s), P_{pub})e(H_3(U\|V), V)$. If the equality does not hold, $\mathcal{A}$ will believe that σ is not a valid undeniable signature.
   2. computes $B' = \frac{e(D_s, H_2(m))}{U^{\widehat{d_s}} \cdot A^{d_s}}$, $C' = \frac{e(D_s, P)}{e(H_1(ID_s\|\text{Undeniable}), P_{pub})^{\widehat{d_s}}}$, $D' = e(D_v, P)e(H_1(ID_v), P_{pub})^{d_v}$.

If $A \neq 1$ and $d_s + d_v = H_4(ID_s \| ID_v \| A \| B' \| C' \| D' \| m \| \sigma)$ holds, V will believe that σ is an invalid undeniable signature of message m.

Selectively Convert: Given a message/signature pair (m, σ) where $\sigma = (U, V, W)$,

1. If $e(W, P) = e(H_1(ID_s), P_{pub})e(H_3(U\|V), V)$ and $U = e(H_2(m), VK_{ID_s})$, S will generate the selective proof as follows: he firstly chooses a random number $r \in \mathbb{Z}_p$ and computes $R_1 = e(P, P)^r, R_2 = e(P, H_2(m))^r$ and computes $c = H_4(ID_s \| R_1 \| R_2 \| m \| \sigma)$, $C = rP - cVK_{ID_s}$. The selective proof is $\Pi^{ID_s}_{(m,\sigma)} = (c, C)$ which will prove that σ is a valid undeniable signature of the message m.
2. Else, if $e(W, P) = e(H_1(ID_s), P_{pub})e(H_3(U\|V), V)$ but $U \neq e(H_2(m), VK_{ID_s})$, the proof is generated as follows: The signer S firstly chooses $z, \alpha, \beta \in_R \mathbb{Z}_P$ and computes:
   (a) $A = [\frac{e(VK_{ID_s}, H_2(m))}{U}]^z \neq 1$, $B = \frac{[e(P, H_2(m))]^\alpha}{U^\beta}$,
   $C = \frac{e(P,P)^\alpha}{e(H_1(ID_s \| \text{Undeniable}), P_{pub})^\beta}$,
   (b) $d = H_4(ID_s \| A \| B \| C \| m \| \sigma)$, $D_s = \alpha P + dzVK_{ID_s}$, $\widehat{d_s} = \beta + dz \pmod{p}$,
   The selective proof is $\Pi^{ID_s}_{(m,\sigma)} = (A, d, D_s, \widehat{d_s})$ which will prove that σ is an invalid undeniable signature of the message m.
3. Otherwise, $e(W, P) \neq e(H_1(ID_s), P_{pub})e(H_3(U\|V), V)$, signer S does not need to do anything since anyone can find that σ is not valid.

Selectively Verify: Given the selective proof $\Pi^{ID_s}_{(m,\sigma)}$ of the message-signature pair (m, σ) where $\sigma = (U, V, W)$, anyone can check whether $e(W, P) \stackrel{?}{=} e(H_1(ID_s), P_{pub})e(H_3(U\|V), V)$.

- If the equality does not hold, output Rej which means the signature σ is not valid.
- Else, if $\Pi^{ID_s}_{(m,\sigma)} = (c, C)$, then check whether
  $$c \stackrel{?}{=} H_4(ID_s \| e(C, P)e(H_1(ID_s \| \text{Undeniable}), P_{pub})^c \| e(C, H_2(m))U^c \| m \| \sigma).$$
  If it holds, one will accept σ as a valid signature and output Acc.
- Otherwise, $\Pi^{ID_s}_{(m,\sigma)} = (A, d, D_s, \widehat{d_s})$, then one continues to compute $B' = \frac{e(D_s, H_2(m))}{U^{\widehat{d_s}} \cdot A^d}$ and $C' = \frac{e(D_s, P)}{e(H_1(ID_s \| \text{Undeniable}), P_{pub})^{\widehat{d_s}}}$. Then, he checks whether $d \stackrel{?}{=} H_4(ID_s \| A \| B' \| C' \| m \| \sigma)$. If $A \neq 1$ and the equation holds, one will reject σ as an invalid signature and output Rej.

Universally Convert: In order to convert all the undeniable signatures into publicly verifiable ones, the signer ID_s publishes his verify key VK_{ID_s} as the universal proof Π_{ID_s}.

Universally Verify: Given a message/signature pair (m, σ) where $\sigma = (U, V, W)$ and VK_{ID_s},

- Anyone can check whether $e(VK_{ID_s}, P) = e(H_1(ID_s \| \text{Undeniable}), P_{pub})$. If the equation does not hold, which means VK_{ID_s} is not valid and the verification halts. Otherwise,

- one continues to check whether $e(W, P) = e(H_1(ID_s), P_{pub})e(H_3(U\|V), V)$. If the equation does not hold, output *Rej*. Otherwise,
- check whether $U = e(VK_{ID_s}, H_2(m))$. If the equation holds as well, output *Acc*. Otherwise, output *Rej*.

3.3 Security Analysis

In this section, we will formally prove the security of the proposed scheme.

Analysis of the Confirmation and Disavowal Protocols
Our confirmation protocol is inspired from a designated verifier proof [12] proposed by Jakobsson, Sako and Impagliazzo that allows a prover to convince a designated verifier of the equality of two discrete logarithms. The denial protocol is an adaptation of a protocol proposed by Camenisch and Shoup [6] to prove the inequality of two discrete logarithms. The Confirmation and Disavowal protocols in our scheme have also been used in [15]. According to the analysis given in [15], both of the two protocols satisfy the completeness, soundness and non-transferability. Therefore, we will not show the detail here and please refer to [15] for the detail of the analysis.

Theorem 1. *If there exists a $(t, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$-forger $\mathcal{F}_{EUF,\ IDCUS}^{CMA,\ CIDA}$ who can additionally submit q_{H_i} queries to the random oracle H_i for $i \in \{1, 2, 3, 4\}$ and win the game defined in Section 2.2 with non-negligible success probability $\mathsf{Succ}\ \mathcal{F}_{EUF,\ IDCUS}^{CMA,\ CIDA}$, then there exists an algorithm $\mathcal{A}$ who can use $\mathcal{F}$ to solve a random instance of Computational Diffie-Hellman problem with probability $Succ_{\mathcal{A},\mathbb{G}_1}^{CDH} \geq \frac{1}{q_{H_1}}(1 - \frac{1}{q_{H_1}})^{q_C}\mathsf{Succ}\ \mathcal{F}_{EUF,IDCUS}^{CMA,CIDA}$ in polynomial time.*

Theorem 2. *If there exists a $(t, q_{US}, q_V, q_{SC}, q_{UC}, q_C)$-distinguisher $\mathcal{D}_{INV,\ IDCUS}^{CMA,\ CIDA}$ who can additionally submit q_{H_i} queries to the random oracle H_i for $i \in \{1, 2, 3, 4\}$ and win the game defined in Section 2.3 with non-negligible advantage $\mathsf{Adv}\ \mathcal{D}_{INV,\ IDCUS}^{CMA,\ CIDA}$, then there exists an algorithm $\mathcal{A}$ who can use $\mathcal{D}$ to solve a random instance of Decisional Bilinear Diffie-Hellman problem with the advantage $\mathsf{Adv}\ _{\mathcal{A},\mathbb{G}_1}^{DBDH} \geq \frac{1}{q_{H_1}q_{H_2}}(1 - \frac{1}{q_{H_1}q_{H_2}})^{q_{US}}(1 - \frac{1}{q_{H_1}})^{q_{UC}+q_C}\mathsf{Adv}\ \mathcal{D}_{INV,\ IDCUS}^{CMA,\ CIDA}$ in polynomial time.*

Theorem 3. *Our proposed scheme is secure against the impersonator defined in Section 2.5 under the assumption that the Computational Diffie-Hellman problem is hard in $\mathbb{G}_1$.*

Due to the page limitation, we omit the proofs of the above three theorems. Please refer to the full version for the details.

4 Conclusion

We presented a first concrete example of identity-based convertible undeniable signature scheme with selective and universal convertibility with provable security. The signatures of our scheme are not publicly verifiable at the time they are generated, while the signer has the choice to decide when and how to make his signatures publicly verifiable, by releasing a selective proof or universal proof. Compared with generic construction of identity-based undeniable signature, our concrete scheme is based on standard security assumptions.

Acknowledgement. The authors would like to thank the anonymous referees of the 3rd SKLOIS Conference on Information Security and Cryptology (Inscrypt 2007) for the suggestions to improve this paper.

References

1. Boyar, J., Chaum, D., Damgård, I.B., Pedersen, T.P.: Convertible Undeniable Signatures. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 189–205. Springer, Heidelberg (1991)
2. Boneh, D., Franklin, M.: Identity-based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
3. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
4. Chaum, D., van Antwerpen, H.: Undeniable Signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)
5. Chow, S.S.M.: Verifiable Pairing and Its Applications. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 170–187. Springer, Heidelberg (2005)
6. Camenisch, J., Shoup, V.: Practical Verifiable Encryption and Decryption of Discrete Logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
7. Damgård, I.B., Pedersen, T.P.: New Convertible Undeniable Signature Schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 372–386. Springer, Heidelberg (1996)
8. Galbraith, S.D., Mao, W., Paterson, K.G.: RSA-Based Undeniable Signatures for General Moduli. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 200–217. Springer, Heidelberg (2002)
9. Galbraith, S.D., Mao, W.: Invisibility and Anonymity of Undeniable and Confirmer Signatures. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 80–97. Springer, Heidelberg (2003)
10. Gennaro, R., Krawczyk, H., Rabin, T.: RSA-Based Undeniable Signatures. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 132–149. Springer, Heidelberg (1997)
11. Gennaro, R., Rabin, T., Krawczyk, H.: RSA-Based Undeniable Signatures. Journal of Cryptology 13(4), 397–416 (2000)
12. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated Verifier Proofs and Their Applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
13. Galindo, D., Herranz, J., Kiltz, E.: On the Generic Construction of Identity-Based Signatures with Additional Properties. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 178–193. Springer, Heidelberg (2006)

14. Han, S., Yeung, W.K.Y., Wang, J.: Identity-based Confirmer Signatures from Pairings over Elliptic Curves. In: Proceedings of the 4th ACM Conference on Electronic Commerce, pp. 262–263. ACM Press, New York (2003)
15. Libert, B., Quisquater, J.-J.: Identity Based Undeniable Signatures. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 112–125. Springer, Heidelberg (2004)
16. Laguillaumie, F., Vergnaud, D.: Time-Selective Convertible Undeniable Signatures. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 154–171. Springer, Heidelberg (2005)
17. Miyazaki, T.: An Improved Scheme of the Gennaro-Krawczyk-Rabin Undeniable Signature System Based on RSA. In: Won, D. (ed.) ICISC 2000. LNCS, vol. 2015, pp. 135–149. Springer, Heidelberg (2001)
18. Michels, M., Petersen, H., Horster, P.: Breaking and Repairing a Convertible Undeniable Signature Scheme. In: Third ACM Conference on Computer and Communications Security, pp. 148–152. ACM Press, New York (1996)
19. Michels, M., Stadler, M.: Efficient Convertible Undeniable Signature Schemes. In: The 4th International Workshop on Selected Areas in Cryptography (SAC 1997), pp. 231–244 (1997)
20. Ogata, W., Kurosawa, K., Heng, S.-H.: The Security of the FDH Variant of Chaum's Undeniable Signature Scheme. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 328–345. Springer, Heidelberg (2005)
21. Shamir, A.: Identity-based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
22. Zhang, F., Safavi-Naini, R., Susilo, W.: Attack on Han et al.'s ID-based Confirmer (Undeniable) Signature at ACM-EC 2003 (2003), `http://eprint.iacr.org/2003/129`

An Efficient ID-Based Proxy Signature Scheme from Pairings[⋆]

Chunxiang Gu and Yuefei Zhu

Network Engineering Department, Zhengzhou Information Science and Technology Institute
P.O. Box 1001-770, Zhengzhou, 450002, P.R. China
gcxiang5209@yahoo.com.cn

Abstract. This paper proposes a new ID-based proxy signature scheme based on the bilinear pairings. The number of paring operation involved in the verification procedure of our scheme is only one, so our scheme is more efficient comparatively. The new scheme can be proved secure with the hardness assumption of the k-Bilinear Diffie-Hellman Inverse problem, in the random oracle model.

Keywords: ID-based cryptography, proxy signatures, bilinear pairings.

1 Introduction

In 1984, Shamir [1] first proposed the idea of ID-based public key cryptography (ID-PKC) to simplify key management procedure of traditional certificate-based PKI. In ID-PKC, an entity's public key is directly derived from certain aspects of its identity, such as an IP address belonging to a network host or an e-mail address associated with a user. Private keys are generated for entities by a trusted third party called a private key generator (PKG). The direct derivation of public keys in ID-PKC eliminates the need for certificates and some of the problems associated with them. Recently, due to the contribution of Boneh and Franklin [2], a rapid development of ID-PKC has taken place. Using bilinear pairings, people proposed many new ID-based signature schemes [3,4,5]. With these ID-based signature schemes, a lot of new extensions, such as ID-based proxy signature scheme, ID-based ring signature scheme, etc.[6,7], have also been proposed.

A proxy signature scheme allows one entity, called *original signer*, to delegate her signing capability to one or more entities, called *proxy signers*. Then the proxy signer can generate *proxy signatures*, which are signatures of some messages on behalf of the original signer. Upon receiving a proxy signature, a verifier can validate its correctness by the given verification procedure, and then is convinced of the original signer's agreement on the signed message.

Since Mambo, Usuda and Okamoto [8] first introduced the proxy signature scheme, many new constructions have been proposed. Based on the delegation

[⋆] Research supported by NSFC (No.60473021), the Natural Science Fundation of Henan Province (072300410260).

type, proxy signatures can be classified as *full delegation*, *partial delegation* and *delegation by warrant*. In [9], Kim et al provided a new type of delegation called partial delegation with warrant, which can be considered as the combination of partial delegation and delegation by warrant. Depending on whether the original signer can generate the same proxy signatures as the proxy signers do, there are two kinds of proxy signature schemes: *proxy-unprotected* and *proxy-protected*. In practice, the *proxy-protected partial delegation by warrant* schemes have attracted much more investigations than others, because they clearly distinguish the rights and responsibilities between the original signer and the proxy signer. In this paper, we will also focus on this kind of schemes. In fact, for simplicity, this special kind of schemes are often called proxy signature schemes.

In [6], Zhang and Kim provided an ID-based proxy signature scheme based on pairings. The scheme is similar to Kim et al.'s scheme [9] which is based on certificate-based public key setting. There are no security proof in their original work. Later, Gu and Zhu [10] gave a formal security model for ID-based proxy signature schemes and provided a security proof for the scheme of Zhang and Kim in the random oracle model.

In this paper, we provide a more efficient ID-based proxy signature scheme from pairings. The new scheme can be proved secure in the random oracle model. The rest of this paper is organized as follows: In Section 2, we recall some preliminary works. In Section 3, we present a new ID-based proxy signature scheme with a correctness and efficiency analysis. In Section 4, we offer a formal security proof in the random orale model. Finally, we conclude in Section 5.

2 Preliminaries

2.1 Bilinear Pairings

Let $(G_1, +)$ and $(G_2, \cdot)$ be two cyclic groups of prime order q. $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be a map which satisfies the following properties.

1. Bilinear: $\forall P, Q \in G_1, \forall \alpha, \beta \in Z_q, \hat{e}(\alpha P, \beta Q) = \hat{e}(P, Q)^{\alpha\beta}$;
2. Non-degenerate: If P is a generator of G_1, then $\hat{e}(P, P)$ is a generator of G_2;
3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.

Such an bilinear map is called an *admissible bilinear pairing* [2]. The Weil pairings and the Tate pairings of elliptic curves can be used to construct efficient admissible bilinear pairings.

We review a complexity problem related to bilinear pairings: the Bilinear Diffie-Hellman Inverse (BDHI) problem [11]. Let P be a generator of G_1, and $a \in Z_q^*$.

– k**-BDHI problem**: given $(P, aP, a^2P, ...a^kP) \in (G_1^*)^{k+1}$, output $\hat{e}(P, P)^{a^{-1}}$.
  An algorithm $\mathcal{A}$ solves k-BDHI problem with the probability ε if

$$Pr[\mathcal{A}(P, aP, a^2P, ...a^kP) = \hat{e}(P, P)^{a^{-1}}] \geq \varepsilon,$$

where the probability is over the random choice of generator $P \in G_1^*$, the random choice of $a \in Z_q^*$ and random coins consumed by $\mathcal{A}$.

We assume through this paper that the k-BDHI problem is intractable, which means that there is no polynomial time algorithm to solve k-BDHI problem with non-negligible probability.

2.2 ID-Based Proxy Signatures

In this paper, unless stated otherwise, let A be the original signer with identity ID_A and private key D_A. He delegates his signing rights to a proxy signer B with identity ID_B and private key D_B. A warrant is used to delegate signing right. In [10], Gu and Zhu gave a formal security model for ID-based proxy signature schemes.

Definition 1. *[10] An ID-based proxy signature scheme is specified by eight polynomial-time algorithms with the following functionalities.*

- **Setup:** *The parameters generation algorithm, takes as input a security parameter $k \in N$ (given as 1^k), and returns a master secret key s and system parameters Ω. This algorithm is performed by PKG.*
- **Extract:** *The private key generation algorithm, takes as input an identity $ID_U \in \{0,1\}^*$, and outputs the secret key D_U corresponding to ID_U. PKG uses this algorithm to extract the users' secret keys.*
- **Delegate:** *The proxy-designation algorithm, takes as input A's secret key D_A and a warrant m_ω, and outputs the delegation $W_{A\rightarrow B}$.*
- **DVerify:** *The designation-verification algorithm, takes as input ID_A, $W_{A\rightarrow B}$ and verifies whether $W_{A\rightarrow B}$ is a valid delegation come from A.*
- **PKgen:** *The proxy key generation algorithm, takes as input $W_{A\rightarrow B}$ and some other secret information z (for example, the secret key of the executor), and outputs a signing key D_p for proxy signature.*
- **PSign:** *The proxy signing algorithm, takes as input a proxy signing key D_p and a message $m \in \{0,1\}^*$, and outputs a proxy signature (m, δ).*
- **PVerify:** *The proxy verification algorithm, takes as input ID_A, ID_B and a proxy signature (m, δ), and outputs 0 or 1. In the later case, (m, δ) is a valid proxy signature of A.*
- **ID:** *The proxy identification algorithm, takes as input a valid proxy signature (m, δ), and outputs the identity ID_B of the proxy signer.*

An ID-based proxy signature scheme should first be correct. That is, $\forall\ m, m_\omega \in \{0,1\}^*$, it should have the following properties:

1. $DVerify(Delegate(m_\omega, D_A), ID_A) = 1$
2. For $W_{A\rightarrow B} = Delegate(m_\omega, D_A)$, let $D_P \leftarrow PKgen(W_{A\rightarrow B}, D_B)$, then $PVerify(PSign(m, D_P), ID_A, ID_B) = 1$, and $ID(PSign(m, D_P)) = ID_B$.

We consider an adversary $\mathcal{A}$ which is assumed to be a probabilistic Turing machine which takes as input the global scheme parameters and a random tape.

Definition 2. *[10] For an ID-based proxy signature scheme ID_PS. We define an experiment $Exp_{\mathcal{A}}^{ID\text{-}PS}(k)$ of adversary $\mathcal{A}$ and security parameter k as follows:*

1. *A challenger $\mathcal{C}$ runs **Setup** and gives the system parameters Ω to $\mathcal{A}$.*
2. *$C_{list} \leftarrow \emptyset$, $D_{list} \leftarrow \emptyset$, $G_{list} \leftarrow \emptyset$, $S_{list} \leftarrow \emptyset$. ($\emptyset$ means NULL.)*
3. *Adversary $\mathcal{A}$ can make the following requests or queries adaptively.*
   - *$Extract(.)$: This oracle takes as input a user's ID_i, and returns the corresponding private key D_i. If $\mathcal{A}$ gets $D_i \leftarrow Extract(ID_i)$, let $C_{list} \leftarrow C_{list} \cup \{(ID_i, D_i)\}$.*
   - *$Delegate(.)$: This oracle takes as input the designator's identity ID and a warrant m_ω, and outputs a delegation W. If $\mathcal{A}$ gets $W \leftarrow Delegate(ID, m_\omega)$, let $D_{list} \leftarrow D_{list} \cup \{(ID, m_\omega, W)\}$.*
   - *$PKgen(.)$: This oracle takes as input the proxy signer's ID and a delegation W, and outputs a proxy signing key D_p. If $\mathcal{A}$ gets $D_p \leftarrow PKgen(ID, W)$, let $G_{list} \leftarrow G_{list} \cup \{(ID, W, D_p)\}$.*
   - *$PSign(.)$: This oracle takes as input the delegation W and message $m \in \{0,1\}^*$, and outputs a proxy signature created by the proxy signer. If $\mathcal{A}$ gets $(m, \tau) \leftarrow PSign(W, m)$, let $S_{list} \leftarrow S_{list} \cup \{(W, m, \tau)\}$.*
4. *$\mathcal{A}$ outputs (ID, m_ω, W) or (W, m, τ).*
5. *If $\mathcal{A}$'s output satisfies one of the following terms, $\mathcal{A}$'s attack is successful.*
   - *The output is (ID, m_ω, W), and satisfies: $DVerify(W, ID) = 1$, $(ID, .) \notin C_{list}$, $(ID, ., .) \notin G_{list}$ and $(ID, m_\omega, .) \notin D_{list}$. $Exp_{\mathcal{A}}^{ID\text{-}PS}(k)$ returns 1.*
   - *The output is (W, m, τ), and satisfies $PVerify((m, \tau), ID_i, ID_j) = 1$, $(W, m, .) \notin S_{list}$, and $(ID_j, .) \notin C_{list}$, $(ID_j, W, .) \notin G_{list}$, where ID_i and ID_j are the identities of the designator and the proxy signer defined by W, respectively. $Exp_{\mathcal{A}}^{ID\text{-}PS}(k)$ returns 2.*

   Otherwise, $Exp_{\mathcal{A}}^{ID\text{-}PS}(k)$ returns 0.

Definition 3. *[10] An ID-based proxy digital signature scheme ID_PS is said to be existential delegation and signature unforgeable under adaptive chosen message and ID attacks (**DS-EUF-ACMIA**), if for any polynomial time adversary $\mathcal{A}$, any polynomial $p(.)$ and big enough k,*

$$Pr[Exp_{\mathcal{A}}^{ID\text{-}PS}(k) = 1] < \frac{1}{p(k)} \quad and \quad Pr[Exp_{\mathcal{A}}^{ID\text{-}PS}(k) = 2] < \frac{1}{p(k)}$$

3 A New Efficient ID-Based Proxy Signature Scheme

In this section, we present a new efficient ID-based proxy signature scheme. Our scheme is based on a variation of the ID-based signature scheme proposed by Barreto et.al [5] in Asiacrypt'05. The method for obtaining private keys from identities is a simplification of a method suggested by Sakai and Kasahara [12]. This leads to a more efficient performance.

3.1 Description of the Scheme

The new scheme can be described as follows:

- **Setup:** Takes as input a security parameter k, and returns a master key s and system parameters $\Omega = (G_1, G_2, q, \hat{e}, P, P_s, P_{ss}, g, g_s, H_1, H_2)$, where $(G_1, +)$ and $(G_2, \cdot)$ are two cyclic groups of order q, $\hat{e} : G_1 \times G_1 \to G_2$ is an admissible bilinear map, $P_s = sP$, $P_{ss} = s^2 P$, $g = \hat{e}(P, P)$, $g_s = \hat{e}(P_s, P)$, $H_1 : \{0,1\}^* \to Z_q^*$ and $H_2 : \{0,1\}^* \times G_1 \to Z_q$ are hash functions.
- **Extract:** Takes as input an identity $ID_X \in \{0,1\}^*$, computes $D_X = (H_1(ID_X) + s)^{-1} P$, and lets D_X be the user's secret key.
- **Delegate:** Takes as input the secret key D_A, the proxy signer's identity ID_B and a warrant m_ω, selects a random $x \in Z_q^*$, computes $q_B = H_1(ID_B)$, $r_A = g_s^x \cdot g^{q_B x}$, $h_A = H_2(m_\omega, r_A)$, $V_A = (x + h_A) D_A$, and outputs the delegation $W_{A \to B} = (m_\omega, r_A, V_A)$.
- **DVerify:** Once B receives $W_{A \to B} = (m_\omega, r_A, V_A)$, he computes $h_A = H_2(m_\omega, r_A)$, $q_A = H_1(ID_A)$, $q_B = H_1(ID_B)$, and accepts the delegation only if
$$\hat{e}((q_A + q_B)P_s + q_A q_B P + P_{ss}, V_A) = r_A \cdot g_s^{h_A} \cdot g^{q_B h_A}.$$
- **PKgen:** If B accepts the delegation $W_{A \to B} = (m_\omega, r_A, V_A)$, he computes the proxy signing key D_P as $D_P = h_A \cdot D_B - V_A$, where $h_A = H_2(m_\omega, r_A)$.
- **PSign:** The proxy signer can pre-compute $\xi = g^{h_A(q_A - q_B)} / r_A$, where $q_A = H_1(ID_A)$, $q_B = H_1(ID_B)$ and r_A is from $W_{A \to B}$. Let D_P be the proxy signing key, for a message m, the proxy signer chooses $y \in Z_q^*$ at random and computes $r_P = \xi^y$, $h_P = H_2(m, r_P)$, $V_P = (y + h_P) D_P$, and lets $(m, \tau) = (m, r_P, V_P, m_\omega, r_A)$ be the proxy signature for m.
- **PVerify:** For a proxy signature $(m, r_P, V_P, m_\omega, r_A)$, a recipient first checks if the proxy signer and the message conform to m_ω. Then he computes $h_P = H_2(m, r_P)$, $q_A = H_1(ID_A)$, $q_B = H_1(ID_B)$ and verifies whether
$$\hat{e}((q_A + q_B)P_s + q_A q_B P + P_{ss}, V_P) = r_P \cdot g^{h_A h_P (q_A - q_B)} \cdot r_A^{-h_P}.$$
If both steps succeed, the proxy signature on behalf of A is valid.
- **ID:** The proxy signer's identity ID_B can be revealed by m_ω.

3.2 Correctness and Efficiency

Set $Q = (q_A + q_B)P_s + q_A q_B P + P_{ss}$. Consistency of the scheme is easily proved as follows: For any $m_\omega \in \{0,1\}^*$, $Delegate(m_\omega, D_A) = (m_\omega, r_A, V_A)$, $h_A = H_2(m_\omega, r_A)$. Then,

$$\begin{aligned}\hat{e}(Q, V_A) &= \hat{e}((q_A + s)(P_s + q_B P), (x + h_A)(q_A + s)^{-1} P) \\ &= \hat{e}(P_s + q_B P, (x + h_A) P) \\ &= r_A \cdot (g_s \cdot g^{q_B})^{h_A}\end{aligned}$$

That is, $DVerify(Delegate(m_\omega, D_A), ID_A) = 1$. On the other hand,

$$D_P = PKgen((m_\omega, r_A, V_A), D_B) = h_A \cdot D_B - V_A = \frac{h_A(q_A - q_B) - x(s + q_B)}{(s + q_A)(s + q_B)} P.$$

For any $m \in \{0,1\}^*$, $PSign(m, D_P) = (m, r_P, V_P, m_\omega, r_A)$, $h_P = H_2(m_\omega, r_P)$. Then,

$$\begin{aligned}
\hat{e}(Q, V_P) &= \hat{e}((q_A + s)(s + q_B)P, (y + h_P)\frac{h_A(q_A - q_B) - x(s + q_B)}{(s + q_A)(s + q_B)}P) \\
&= \hat{e}(P, (h_A(q_A - q_B) - x(s + q_B))P)^{(y+h_P)} \\
&= (g^{h_A(q_A - q_B)}/g^{x(s+q_B)})^{(y+h_P)} \\
&= (g^{h_A(q_A - q_B)}/r_A)^{(y+h_P)} \\
&= r_P \cdot (g^{h_A(q_A - q_B)}/r_A)^{h_P}
\end{aligned}$$

Hence, $PVerify(PSign(m, D_P), ID_A, ID_B) = 1$. m_ω designates the identity of the proxy signer, and it is a part of the signature. So it is easy to see that $ID(PSign(m, D_P)) = ID_B$.

Denote by M an ordinary scalar multiplication in $(G_1, +)$, by E an Exp. operation in $(G_2, .)$, and by $\hat{e}$ a computation of the pairing. The hash function maps an identity to an element in G_1 used by the scheme in [6] usually requires a "Maptopoint operation" [2]. As discussed in [2], Maptopoint operation (denoted by H) is so inefficient that we can't neglect it. Do not take other operations into account. We compare our new scheme to the ID-based proxy signature scheme of Zhang and Kim [6] in the following table.

schemes	Delgate	DVerify	PKgen	PSign	PVerify
Zhang-Kim [6]	$2M + 1E$	$2\hat{e} + 1E + 1H$	$1M$	$2M + 1E$	$2\hat{e} + 2E + 2H$
proposed	$1M + 2E$	$1\hat{e} + 2M + 2E$	$1M$	$1M + 1E$	$1\hat{e} + 2M + 2E$

Note: The hash function used in our scheme which maps an identity to an element in Z_q^* is so efficient that we usually can neglect it.

Some general performance enhancements can be applied to our schemes. For pre-selected $P \in G_1$ and $\mu \in G_2$, there are efficient algorithms [13] to compute kP and μ^l for random $k, l \in Z_q$ by pre-computing and storing. In our scheme, P, P_s and g, g_s are fixed system parameters. And for the signer and the proxy signer, their secret keys are also fixed.

4 Security Proof

In this section, we reduce the security of our scheme to the hardness assumption of k-BDHI problem in the random oracle model.

Assume there is an adversary $\mathcal{F}_0$ who can break the ID-based proxy signature scheme. We will construct a polynomial time algorithm $\mathcal{F}_1$ that, by simulating the challenger and interacting with $\mathcal{F}_0$, solves $(n_1 + 1)$-BDHI problem, where n_1 is the number of queries that $\mathcal{F}_0$ can ask to the random oracle $H_1(.)$.

Lemma 1. *Given system parameters $\Omega = (G_1, G_2, q, \hat{e}, P, P_s, P_{ss}, g, g_s, H_1, H_2)$ and identities $ID_A, ID_B \in \{0,1\}^*$, let $q_A = H_1(ID_A)$, $q_B = H_1(ID_B)$, $D_A =$*

$(s+q_A)^{-1}P$, and $T=(q_A+q_B)P_s+q_Aq_BP+P_{ss}$, the following distributions are the same.

$$\delta=\left\{(r,h,V)\;\middle|\;\begin{array}{c}x\in_R Z_q^*\\ h\in_R Z_q\\ r=g_s^x\cdot g^{xq_B}\\ V=(x+h)D_A\end{array}\right\} and\;\; \delta'=\left\{(r,h,V)\;\middle|\;\begin{array}{c}V\in_R G_1\\ h\in_R Z_q\\ r=\hat{e}(T,V)\cdot g_s^{-h}\cdot g^{-q_Bh}\\ r\neq 1\end{array}\right\}$$

Proof: First we choose a triple (α,β,γ) such that $\alpha\in G_2^*,\beta\in Z_q,\gamma\in G_1$ and satisfying $\alpha=\hat{e}(T,\gamma)\cdot g_s^{-\beta}\cdot g^{-\beta q_B}$. We then compute the probability of appearance of this triple following each distribution of probabilities:

$$Pr_\delta[(r,h,V)=(\alpha,\beta,\gamma)]=\Pr_{x\neq 0}\begin{bmatrix}\alpha=g_s^x\cdot g^{xq_B}\\ h=\beta\\ (x+h)D_A=\gamma\end{bmatrix}=\frac{1}{q(q-1)}.$$

$$Pr_{\delta'}[(r,h,V)=(\alpha,\beta,\gamma)]=\Pr_{r\neq 1}\begin{bmatrix}h=\beta\\ V=\gamma\\ \alpha=r=\hat{e}(T,V)\cdot g_s^{-h}\cdot g^{-q_Bh}\end{bmatrix}=\frac{1}{q(q-1)}.$$

Hence, we can simulate the *Delegate*(.) oracle for input (ID_A,ID_B,m_ω) without the secret key D_A indistinguishably from the real one as following:

– $\mathcal{S}_\mathcal{D}(ID_A,ID_B,m_\omega)$:
  - Pick randomly $V_A\in G_1$, $h_A\in Z_q$.
  - Compute $r_A=\hat{e}((q_A+q_B)P_s+q_Aq_BP+P_{ss},V_A)\cdot g_s^{-h_A}\cdot g^{-q_Bh_A}$, where $q_A=H_1(ID_A)$, $q_B=H_1(ID_B)$,
  - If $H_2(m_\omega,r_A)$ has been defined, then abort (a collision appears). Otherwise, set $H_2(m_\omega,r_A)=h_A$.
  - Set $W=(m_\omega,r_A,V_A)$.

Lemma 2. *Given system parameters $\Omega=(G_1,G_2,q,\hat{e},P,P_s,P_{ss},g,g_s,H_1,H_2)$, identities $ID_A,ID_B\in\{0,1\}^*$ and $W_{A\to B}=(m_\omega,r_A,V_A)$, let $q_A=H_1(ID_A)$, $q_B=H_1(ID_B)$, $h_A=H_2(m_\omega,r_A)$, $D_B=(s+q_B)^{-1}P$, $D_P=h_AD_B-V_A$, $\xi=g^{h_A(q_A-q_B)}/r_A$ and $T=(q_A+q_B)P_s+q_Aq_BP+P_{ss}$, the following distributions are the same.*

$$\delta=\left\{(r_P,h_P,V_P)\;\middle|\;\begin{array}{c}y\in_R Z_q^*\\ h_P\in_R Z_q\\ r_P=\xi^y\\ V_P=(y+h_P)D_P\end{array}\right\}$$

and

$$\delta'=\left\{(r_P,h_P,V_P)\;\middle|\;\begin{array}{c}V_P\in_R G_1\\ h_P\in_R Z_q\\ r_P=\hat{e}(T,V_P)\cdot(g^{-h_A(q_A-q_B)}\cdot r_A)^{h_P}\\ r\neq 1\end{array}\right\}$$

Proof: Readers can see that the proof is almost the same as that of Lemma 1. We omit it in this paper.

That is, we can simulate the *PSign*(.) oracle for input $(W_{A\to B}=(m_\omega,r_A,V_A),m)$ without the secret proxy key D_P indistinguishably from the real one as following:

– $\mathcal{S}_{\mathcal{PS}}(W_{A\to B}, m)$:
  - Pick randomly $V_P \in G_1$, $h_P \in Z_q$.
  - Check whether $H_2(m_\omega, r_A)$ is defined. If not, request oracle $H_2(.)$ with (m_ω, r_A). Let $H_2(m_\omega, r_A) = e$.
  - Compute $r_P = \hat{e}((q_A + q_B)P_s + q_A q_B P + P_{ss}, V_P) \cdot (g^{-e(q_A - q_B)} \cdot r_A)^{h_P}$, where $q_A = H_1(ID_A)$, $q_B = H_1(ID_B)$.
  - If $H_2(m, r_P)$ has been defined, then abort(a collision appears). Otherwise, set $H_2(m, r_P) = h_P$.
  - Let $(m, \tau) = (m, r_P, V_P, m_\omega, r_A)$ be the reply.

Theorem 1. *In the random oracle mode, let $\mathcal{F}_0$ be a polynomial-time adversary who manages an $Exp_{\mathcal{A}}^{ID\text{-}PS}(k)$ within a time bound $T(k)$, and gets return 1 or 2 by non-negligible probability $\varepsilon(k)$. We denote respectively by n_1,n_2 and n_3 the number of queries that $\mathcal{F}_0$ can ask to the random oracle $H_1(.)$, $H_2(.)$ and the proxy singing oracle $PSign(.)$. Assume that $\varepsilon(k) \geq 10(n_3+1)(n_2+n_3)n_1/q$, then there is an adversary $\mathcal{F}_1$ who can solve (n_1+1)-BDHI problem within expected time less than $120686 \cdot n_2 \cdot n_1 \cdot T(k)/\varepsilon(k)$.*

Proof: Without any loss of generality, we may assume that for any ID, $\mathcal{F}_0$ queries $H_1(.)$ with ID before ID is used as (part of) an input of any query to $Extract(.)$, $Delegate(.)$, $PKgen(.)$ and $PSign(.)$, by using a simple wrapper of $\mathcal{F}_0$.

$\mathcal{F}_1$ is given input parameters of pairing $(q, G_1, G_2, \hat{e})$ and a random instance $(P, aP, a^2P, ..., a^{n_1}P, a^{n_1+1}P)$ of the (n_1+1)-BDHI problem, where P is random in G_1^* and a is a random in Z_q^*. $\mathcal{F}_1$ simulates the challenger and interacts with $\mathcal{F}_0$ as follows:

1. $\mathcal{F}_1$ randomly chooses different $h_0, h_1, ...h_{n_1-1} \in Z_q^*$, and computes $f(x) = \prod_{i=1}^{n_1-1}(x+h_i) = \sum_{i=0}^{n_1-1} c_i x^i$.
2. $\mathcal{F}_1$ computes $Q = \sum_{i=0}^{n_1-1} c_i a^i P = f(a)P$, $aQ = \sum_{i=0}^{n_1-1} c_i a^{i+1} P$, $a^2Q = \sum_{i=0}^{n_1-1} c_i a^{i+2} P$, and $Q' = \sum_{i=1}^{n_1-1} c_i a^{i-1} P$. In the (unlikely) situation where $Q = 1_{G_1}$, there exists an $h_i = -a$, hence, $\mathcal{F}_1$ can solve the (n_1+1)-BDHI problem directly and abort.
3. $\mathcal{F}_1$ computes $f_i(x) = f(x)/(x+h_i) = \sum_{j=0}^{n_1-2} d_j x^j$. Obviously, $(a+h_i)^{-1}Q = (a+h_i)^{-1}f(a)P = f_i(a)P = \sum_{j=0}^{n_1-2} d_j a^j P$ for $1 \leq i \leq n_1$.
4. $\mathcal{F}_1$ randomly chooses an index t with $1 \leq t \leq n_1$, sets $v = 0$.
5. $\mathcal{F}_1$ computes $g = \hat{e}(Q, Q)$, $g_s = \hat{e}(aQ, Q)$, sets the system parameters $\Omega = (G_1, G_2, q, \hat{e}, Q, aQ, a^2Q, g, g_s, H_1, H_2)$, where H_1, H_2 are random oracles controlled by $\mathcal{F}_1$.
6. $\mathcal{F}_1$ sets $C_{list} = \emptyset$, $D_{list} = \emptyset$, $G_{list} = \emptyset$, $S_{list} = \emptyset$, and starts $Exp_{\mathcal{F}_0}^{ID\text{-}PS}(k)$ by giving $\mathcal{F}_0$ the system parameters Ω. During the execution, $\mathcal{F}_1$ emulates $\mathcal{F}_0$'s oracles as follows:
   – $H_1(.)$: $\mathcal{F}_1$ maintains a H_1_list, initially empty. For a query ID, if ID already appears on the H_1_list in a tuple (ID, l, D), $\mathcal{F}_1$ responds with l. Otherwise, sets $v = v+1$, $ID_v = ID$, if $v = t$, $\mathcal{F}_1$ sets $l_v = h_0$, $D_v = \perp$; otherwise, $\mathcal{F}_1$ selects a random $n_1 \geq \vartheta > 0$ which has not been chosen

and sets $l_v = h_\vartheta + h_0$, $D_v = (a + h_\vartheta)^{-1}Q$. In both case, adds the tuple (ID_v, l_v, D_v) to H_1_list and responds with l_v.

- $H_2(.)$: For a query (m, r), $\mathcal{F}_1$ checks if $H_2(m, r)$ is defined. If not, $\mathcal{F}_1$ picks a random $c \in Z_q^*$ and defines $H_2(m, r) = c$. $\mathcal{F}_1$ returns $H_2(m, r)$ to $\mathcal{F}_0$.
- *Extract*(.): For input ID_i, $\mathcal{F}_1$ searches in H_1_list for (ID_i, l_i, D_i). If $D_i = \perp$ then $\mathcal{F}_1$ aborts. Otherwise, $\mathcal{F}_1$ responds with D_i. Set $C_{list} \leftarrow C_{list} \cup \{(ID_i, D_i)\}$.
- *Delegate*(.): For input ID_i and warrant m_ω (we assume the identity of the proxy signer's is ID_j), if $i \neq t$, $\mathcal{F}_1$ computes $W = Delegate(D_i, m_\omega)$. Otherwise, $\mathcal{F}_1$ runs the simulator $\mathcal{S_D}(ID_t, ID_j, m_\omega)$ and gets the reply W. Let W be the reply to $\mathcal{F}_0$, and set $D_{list} \leftarrow D_{list} \cup \{(ID_i, m_\omega, W)\}$.
- *PKgen*(.): For input proxy signer's identity ID_j and delegation $W = (m_\omega, r_0, V_0)$, if $j = t$, then abort. Otherwise, $\mathcal{F}_1$ computes $D_P = H_2(m_\omega, r_0)D_j - V_0$ as the reply to $\mathcal{F}_0$. Let $G_{list} \leftarrow G_{list} \cup \{(W, ID_j, D_P)\}$.
- *PSign*(.): For input $W = (m_\omega, r_0, V_0)$ and message m, designator's identity be ID_i and proxy signer's identity be ID_j. If $j \neq t$, $\mathcal{F}_1$ computes the proxy signature $\tau = (r_P, V_P, m_\omega, r_0)$ on m with secret signing key $D_P = H_2(m_\omega, r_0)D_j - V_0$, and return (m, τ) as the reply. Otherwise, $\mathcal{F}_1$ simulates ID_t's proxy signature on behalf of ID_i with the simulator $\mathcal{S_{PS}}(W, m)$ and lets the output (m, τ) of $\mathcal{S_{PS}}$ as the reply. Let $S_{list} \leftarrow S_{list} \cup \{(W, m, \tau)\}$.

7. $\mathcal{F}_1$ keeps interacting with $\mathcal{F}_0$ until $\mathcal{F}_0$ halts or aborts.
   - Case 0: If $\mathcal{F}_0$'s output is (ID^*, m_ω^*, W^*), where $W^* = (m_\omega^*, r_0^*, V_0^*)$, and satisfies: $DVerify(W^*, ID^*) = 1$, $(ID^*, .) \notin C_{list}$, $(ID^*, ., .) \notin G_{list}$ and $(ID^*, m_\omega^*, .) \notin D_{list}$, and $ID^* = ID_t$, $\mathcal{F}_1$ can get a delegation forgery $(m_\omega^*, r_0^*, h_0^*, V_0^*)$ corresponding to identity ID_t (whose secret key is $a^{-1}Q$), where $h_0^* = H_2(m_\omega^*, r_0^*)$. By replays of Step 6 with the same random tape but different choices of $H_2(.)$, as done in the Forking Lemma [14], $\mathcal{F}_1$ can get another valid forgery $(m_\omega^*, r_0^*, h_1^*, V_1^*)$ such that $h_1^* \neq h_0^*$. Set $statue = 0$.
   - Case 1: If $\mathcal{F}_0$'s output is $(W^*, m^*, \tau^*) = ((m_\omega^*, r_0^*, V_0^*), m^*, (r_P^*, V_P^*, m_\omega^*, r_0^*))$ with designator's identity ID_t and proxy signer's identity ID_j, and satisfies $PVerify((m^*, \tau^*), ID_t) = 1$, $(W^*, m^*, .) \notin S_{list}$, and $(ID_j, .) \notin C_{list}$, $(ID_j, W^*, .) \notin G_{list}$, $\mathcal{F}_1$ can get a forgery $(W^*, m^*, (r_P^*, h_P^*, V_P^*, m_\omega^*, r_0^*))$ corresponding to proxy signing key $D_P = \mu a^{-1}Q - V_0^*$, where $\mu = H_2(m_\omega^*, r_0^*)$ and $h_P^* = H_2(m, r_P^*)$. Define $H_2(m_\omega^*, r_0^*) = \mu$. By replays of Step 4 with the same random tape but different choices of $H_2(.)$, as done in the Forking Lemma [14], $\mathcal{F}_1$ can get another valid forgery $(W^*, m^*, (r_P^*, h_{P1}^*, V_{P1}^*, m_\omega^*, r_0^*))$ such that $h_{P1}^* \neq h_P^*$. Set $statue = 1$.
8. $\mathcal{F}_1$ can compute $a^{-1}Q$ as follows:
   - If $statue = 0$,

$$a^{-1}Q = (h_1^* - h_0^*)^{-1}(V_1^* - V_0^*).$$

– If $statue = 1$,

$$a^{-1}Q = H_2(m_\omega^*, r_0^*)^{-1}((h_{P1}^* - h_P^*)^{-1}(V_{P1}^* - V_P^*) + V_0^*).$$

9. $\mathcal{F}_1$ computes $\hat{e}(Q, a^{-1}Q) = \hat{e}(Q, Q)^{a^{-1}}$. Then, $\mathcal{F}_1$ computes and outputs

$$\hat{e}(P, P)^{a^{-1}} = \hat{e}(Q, Q)^{a^{-1}} / \hat{e}(Q', Q + c_0 P))^{c_0^{-2}}$$

as the solution to the given instance of $(n_1 + 1)$-BDHI problem.

This completes the description of $\mathcal{F}_1$.

During $\mathcal{F}_1$'s execution, if $\mathcal{F}_0$ manages an $Exp_{\mathcal{F}_0}^{ID\text{-}PS}(k)$ and gets return 1 or 2, collisions appear with negligible probability, as mentioned in [14]. So $\mathcal{F}_1$'s simulations are indistinguishable from $\mathcal{F}_0$'s oracles. Because t is chosen randomly, $\mathcal{F}_1$ can get a forgery of $(m_\omega^*, r_0^*, h_0^*, V_0^*)$ corresponding to identity ID_t, or $(W^*, m^*, (r_P^*, h_P^*, V_P^*, m_\omega^*, r_0^*))$ corresponding to proxy signing key $D_P = \mu a^{-1}Q - V_0^*$, within expected time $T(k)$ with probability $\varepsilon(k)/n_1$.

In fact, the delegation and proxy signing are both schemes producing signatures of the form (m, r, h, V), where each of r, h, V corresponds to one of the three moves of a honest-verifier zero-knowledge protocol. By applying the forking lemma[14], $\mathcal{F}_1$ can produce two valid forgery $(m_\omega^*, r_0^*, h_0^*, V_0^*)$ and $(m_\omega^*, r_0^*, h_1^*, V_1^*)$ such that $h_0^* \neq h_1^*$ within expected time less than $120686 \cdot n_2 \cdot n_1 \cdot \frac{T(k)}{\varepsilon(k)}$. So $\mathcal{F}_1$ can output $\hat{e}(P, P)^{a^{-1}}$. Thus we prove the theorem.

5 Conclusion

This paper presents an efficient and provably secure ID-based proxy signature scheme based on the bilinear pairings. Although fruitful achievements [15,16] have been made in enhancing the computation of pairings, the computation of pairings are still a heavy burden for schemes from pairings. The number of paring operation involved in the verification procedure of our schemes is only one, so our scheme is more efficient comparetively. The scheme can be proved secure with the hardness assumption of the k-BDHI problem, in the random oracle model.

References

1. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
2. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
3. Cha, J.C., Cheon, J.H.: An identity-based signature from gap Diffie-Hellman groups. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 18–30. Springer, Heidelberg (2002)

4. Hess, F.: Efficient identity based signature schemes based on pairings. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 310–324. Springer, Heidelberg (2003)
5. Barreto, P.S.L.M., Libert, B., McCullagh, N., Quisquater, J.: Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 515–532. Springer, Heidelberg (2005)
6. Zhang, F., Kim, K.: Efficient ID-based blind signature and proxy signature from bilinear pairings. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 312–323. Springer, Heidelberg (2003)
7. Zhang, F., Kim, K.: ID-based blind signature and ring signature from pairings. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 533–547. Springer, Heidelberg (2002)
8. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: 3rd ACM Conference on Computer and Communications Security (CCS 1996), pp. 48–57. ACM Press, New York (1996)
9. Kim, S., Park, S., Won, D.: Proxy signatures, revisited. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 223–232. Springer, Heidelberg (1997)
10. Gu, C., Zhu, Y.: Provable Security of ID-based Proxy Signature Schemes. In: Lu, X., Zhao, W. (eds.) ICCNMC 2005. LNCS, vol. 3619, pp. 1277–1286. Springer, Heidelberg (2005)
11. Boneh, D., Boyen, X.: Efficient Selective ID Secure Identity Based Encryption without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
12. Sakai, R., Kasahara, M.: ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054
13. Sakai, Y., Sakurai, K.: Efficient Scalar Multiplications on Elliptic Curves without Repeated Doublings and Their Practical Performance. In: Clark, A., Boyd, C., Dawson, E.P. (eds.) ACISP 2000. LNCS, vol. 1841, pp. 59–73. Springer, Heidelberg (2000)
14. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of Cryptology 13(3), 361–369 (2000)
15. Barreto, P., Kim, H., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
16. Duursma, I., Lee, H.: Tate pairing implementation for hyperelliptic curves $y^2 = x^p + x + d$. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 111–123. Springer, Heidelberg (2003)

Improved and Multiple Linear Cryptanalysis of Reduced Round Serpent

B. Collard, F.-X. Standaert*, and J.-J. Quisquater

UCL Crypto Group, Microelectronics Laboratory, Louvain-la-Neuve, Belgium

Abstract. This paper reports on the improved and multiple linear cryptanalysis of reduced round Serpent by mean of a branch-and-bound characteristic search within the algorithm. We first present a 9-round linear characteristic with probability $\frac{1}{2} + 2^{-50}$ that involves a reduction of the estimated data complexity of the best reported attack by a factor of 16. Then, we investigate the possibility to take advantage of multiple linear approximations for improving the linear cryptanalysis of Serpent. According to the framework of Biryukov *et al.* from Crypto 2004, we provide estimations of the improved data complexity of such attacks and derive practical cryptanalysis scenarios. For computational reasons, the branch-and-bound search is not guaranteed to be optimal. However, these are the best reported complexities of a linear attack against Serpent.

Keywords: linear cryptanalysis, multiple linear cryptanalysis, Advanced Encryption Standard, Serpent, linear approximations, branch-and-bound.

1 Introduction

The linear cryptanalysis [8] is one of the most powerful attacks against modern block ciphers in which an adversary exploits a linear approximation of the type:

$$P[\chi_P] \oplus C[\chi_C] = K[\chi_K] \tag{1}$$

In this expression, P, C and K respectively denote the plaintext, ciphertext and the secret key while $A[\chi]$ stands for $A_{a_1} \oplus A_{a_2} \oplus ... \oplus A_{a_n}$,with $A_{a_1}, ..., A_{a_n}$ representing particular bits of A in positions $a_1, ..., a_n$ (χ is usually denoted as a mask). In practice, linear approximations of block ciphers can be obtained by the concatenation of one-round approximations and such concatenations (also called characteristics) are mainly interesting if they maximize the deviation (or bias) $\epsilon = p - \frac{1}{2}$ (where p is the probability of a given linear approximation).

In its original paper, Matsui described two methods for exploiting the linear approximations of a block cipher, respectively denoted as *algorithm 1* and *algorithm 2*. In the first one, given an r-round linear approximation with sufficient bias, the algorithm simply counts the number of times the left side of Equation 1 is equal to zero for N pairs (plaintext, ciphertext). If $T > N/2$, then it assumes either $K[\chi_K] = 0$ if $\epsilon > 0$ or $K[\chi_K] = 1$ if $\epsilon < 0$ so that the experimental value

* Postdoctoral researcher of the Belgian Fund for Scientific Research (FNRS).

$(T - N/2)/N$ matches the theoretical bias. If $T > N/2$, an opposite reasoning holds. For the attack to be successful, it is shown in [8] that the number of available (plaintext, ciphertext)-pairs must be proportional to $\frac{1}{\epsilon^2}$.

In the second method, an r-1-round characteristic is used and a partial decryption of the last round is performed by guessing the key bits involved in the approximation. As a consequence, all the guessed key bits can be recovered rather than the parity $K[\chi_K]$ which yields much more efficient attacks in practice.

Among the various proposals to improve the linear cryptanalysis of block ciphers, Kaliski and Robshaw proposed in 1994 an algorithm using several linear approximations [6]. However, their method imposed a strict constraint as it requires to use only approximations implying the same bits of subkeys $K[\chi_K]$. This restricted at the same time the number and the quality of the approximations available. As a consequence, an approach removing this constraint was proposed in 2004 [4] that can be explained as follows. Let us suppose that one has access to m approximations on r block cipher rounds of the form:

$$P[\chi_P^i] \oplus C[\chi_C^i] = K[\chi_K^i] \; (1 \leq i \leq m), \tag{2}$$

and wishes to determine the value of the binary vector:

$$\mathbf{Z} = (z_1, z_2, ..., z_m) = (K[\chi_K^1], K[\chi_K^2], ..., K[\chi_K^m]) \tag{3}$$

The improved algorithm associates a counter T_i with each approximation, that is incremented each time the corresponding linear approximation is verified for a particular pair (plaintext-ciphertext). As for algorithm 1, the values of $K[\chi_K^i]$ are determined from the experimental bias $(T^i - N/2)/N$ and the theoretical bias ϵ_i by means of a maximum likelihood rule. The extension of algorithm 2 to multiple approximations is similarly described in [4].

An important consequence of this work is that the theoretical data complexity of the generalized multiple linear cryptanalysis is decreased compared to the original attack. According to the authors of [4], the attack requires a number of texts inversely proportional to the capacity of the system of equations used by the adversary that is defined as: $\overline{c}^2 = 4 \cdot \sum_{i=1}^{n} \epsilon_i^2$. Therefore, by increasing this quantity by using more approximations, one can decrease the number of texts necessary to perform a successful key recovery.

In this paper, we aim to apply the previously described cryptanalytic tools to the AES candidate Serpent [1]. For this purpose, we first apply a branch-and-bound algorithm to derive an improved single linear characteristic for the cipher. It allows us to reduce the expected complexity of a linear cryptanalysis by a factor of 16. Due to the structure of the Serpent algorithm components (in particular its S-boxes and diffusion layer), the Matsui's branch-and-bound method could not be applied as such and we proposed a modified algorithm, based on the minimization of the number of active S-boxes in the linear transform. Then, in the second part of the paper, we take advantage of our modified algorithm in order to investigate multiple linear approximations. We show that a large number of linear approximations with significant biases can be derived and evaluate the resulting capacities of the obtained systems. As result of these experiments, the *theoretical* complexity against 10-rounds Serpent can be as low as 2^{80}. We mention that these conclusions have to be tempered by the possibility to perform practical attacks dealing with large number of equations and by the possibility that a significant part of these equations give rise to dependent information, as discussed at the end of this paper. Therefore, practical experiments of multiple linear cryptanalysis against actual ciphers appear to be a necessary step for the better understanding of these theoretical improvements.

2 The Serpent Algorithm

The Serpent block cipher was designed by Ross Anderson, Elie Biham and Lars Knudsen [1]. It was an Advanced Encryption Standard candidate, finally rated just behind the AES Rijndael. Serpent has a classical SPN structure with 32 rounds and a block width of 128 bits. It accepts keys of 128, 192 or 256 bits and is composed of the following operations:

- an initial permutation IP,
- 32 rounds, each of them built upon a subkey addition, a passage through 32 S-boxes and a linear transformation L (excepted the last round, where the linear transformation is not applied),
- a final permutation FP.

In each round R_i, only one S-box is used 32 times in parallel. The cipher uses 8 distinct S-boxes S_i $(0 \leq i \leq 7)$ successively along the rounds and consequently, each S-box is used in exactly four different rounds. Finally, the linear diffusion transform is entirely defined by *XOR*s ($\oplus$), *rotations* ($\lll$) and left *shifts* ($\ll$). Its main purpose is to maximize the avalanche effect within the cipher. If one indicates by X_0, X_1, X_2, X_3 the $4 \cdot 32$ bits at the input of the linear transformation, it can be defined by the following operations:

$$\begin{array}{c}
\text{input} = X_0, X_1, X_2, X_3 \\
\hline
X_0 = X_0 \lll 13 \\
X_2 = X_2 \lll 3 \\
X_1 = X_1 \oplus X_0 \oplus X_2 \\
X_3 = X_3 \oplus X_2 \oplus (X_0 \ll 3) \\
X_1 = X_1 \lll 1 \\
X_3 = X_3 \lll 7 \\
X_0 = X_0 \oplus X_1 \oplus X_3 \\
X_2 = X_2 \oplus X_3 \oplus (X_1 \ll 7) \\
X_0 = X_0 \lll 5 \\
X_2 = X_2 \lll 22 \\
\hline
\text{output} = X_0, X_1, X_2, X_3
\end{array}$$

3 Matsui's Branch-and-Bound Approximation Search

The first step in a linear cryptanalysis consists in finding linear approximations with biases as high as possible. In practice, the adversary usually starts by investigating the non-linear components in the cipher (*e.g.* the S-boxes) and tries to extrapolate partial approximations through the whole. A first problem is then to compute the probability of the concatenated linear approximations, that is usually estimated thanks to the following *piling-up lemma*. Let the bias of a linear approximation on the block cipher ith round be defined as:

$$(\chi_{I_i}, \chi_{O_i}) = \epsilon_i = \Pr\{I_i[\chi_{I_i}] \oplus O_i[\chi_{O_i}] = 0\} - 1/2 \tag{4}$$

The total bias ϵ_{tot} on r rounds is then given by:

$$\epsilon_{tot} = [\epsilon_1, \epsilon_2, ..., \epsilon_r] = 2^{r-1} \prod_{i=1}^{r} \epsilon_i, \tag{5}$$

and the best r rounds linear approximation is defined as:

$$B_r = \max_{\substack{\chi_{I_i} = \chi_{O_{i-1}} \\ (2 \leq i \leq r)}} [(\chi_{I_1}, \chi_{O_1}), (\chi_{I_2}, \chi_{O_2}), ..., (\chi_{I_r}, \chi_{O_r})] \tag{6}$$

Next to the pilling up lemma, the problem of searching the best r-round approximation is not trivial. Il consists of finding $r + 1$ binary masks (one for each output round plus one mask for the input of the cipher), which define a linear approximation of maximum bias for a particular encryption algorithm. The difficulty of the problem mainly comes from the great cardinality of the set of candidates. In 1994, Matsui proposed a branch-and-bound algorithm making possible to effectively find the best linear approximation of the DES [9]. The algorithm works by induction: knowing the value of maximum bias on the *r-1* first rounds, it manages to find the maximum bias on r rounds as well as the corresponding input and output masks. For this purpose, it constantly requires to know a lower bound $\overline{B_r}$. This bound must imperatively be lower or equal to the exact B_r and the closer it is from B_r, the faster the algorithm converges.

In practice, the set of all the possible characteristics through the cipher can be seen as a large tree. At the roots stand the input masks of the cipher approximations, at each branch stand the output masks of a round and at the leaves stand the output masks of the cipher. Each branch of the tree is divided in as many new branches as there are possible approximations for the next round. In this tree, a linear approximation on i rounds forms a path starting from a root up to the i-th level. The number of leaves of the tree growing exponentially with the number of rounds, it quickly becomes unthinkable to evaluate all the approximations by exhaustive search. The principle of the branch-and-bound therefore consists in cutting whole branches of the tree as soon as it becomes obvious that all the corresponding linear approximations will have a bias lower than the bound $\overline{B_r}$. Being given an approximation on i rounds and its bias ϵ_i, a linear approximation on r rounds generated by concatenating an approximation on $r-i$ rounds with this approximation cannot have a bias better than $[B_{r-i}, \epsilon_i]$. Consequently, if $[\epsilon_i, B_{r-i}]$ is strictly lower than $\overline{B_r}$, it is useless to prospect this part of the tree more in detail. Matsui's technique is obtained by the systematic application of this reasoning and is described in more details in [9], Appendix A.

The branch-and-bound strategy can be applied as such to many ciphers. However, its efficiency and time complexity varies, notably depending on the initial estimation $\overline{B_r}$. Small estimations increase the size of the search tree. If the estimation is too large, the algorithm may not return any characteristic at all [5]. For DES, good estimations could be found by first performing a restricted search over all characteristics which only cross a single S-box in each round. Unfortunately, the algorithm may perform poorly for other ciphers and in particular, could hardly be applied as such to Serpent, as the next section underlines.

4 Linear Approximations Search for Serpent

4.1 Observations on the S-Boxes and Single Round Approximations

The Serpent S-boxes have only four (non-trivial) different biases: $\pm 2^{-2}$ and $\pm 2^{-3}$. Consequently, by the piling-up lemma, the bias of any approximation is a power of 2. The S-boxes S_0, S_1, S_2 and S_6 have 36 biases $\epsilon = \pm 0.25$ and 96 biases $\epsilon = \pm 0.125$. The S-boxes S_3, S_4, S_5 and S_7 have 32 biases $\epsilon = \pm 0.25$ and 112 biases $\epsilon = \pm 0.125$. In Table 1, we summarized the distributions of the approximations biases for one round of Serpent and compared them with similar results obtained in [10] for the DES and FEAL.

Table 1. Number of 1-round linear approximations with bias $\geq \epsilon$ for some ciphers

ϵ	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
DES	1	13	195	3803	40035	371507	...
FEAL	16	1808	98576	$3.45 \cdot 10^6$	$7.74 \cdot 10^8$	$1.22 \cdot 10^9$	...
Serpent: $S_{0,1,2,6}$	1	1153	647041	$2.35 \cdot 10^8$	$6.25 \cdot 10^{10}$	$1.29 \cdot 10^{13}$	$2.15 \cdot 10^{15}$
Serpent: $S_{3,4,5,7}$	1	1025	512513	$1.67 \cdot 10^8$	$3.96 \cdot 10^{10}$	$7.33 \cdot 10^{12}$	$1.10 \cdot 10^{15}$

This table clearly illustrates that the number of approximations on a round of Serpent is several orders superior to those of DES and FEAL. We consequently expected practical problems in the linear approximations search due to an explosion of the number of candidates to be explored within the branch-and-bound. This was experimentally confirmed: a classical implementation of Matsui's algorithm appeared unable to determine the best approximation on as low as three rounds. One reason for this phenomenon is the good diffusion provided by the linear transform, causing a significant increase in the number of active S-boxes at each round, with an exponential increase in the number of approximation-candidates being tested. Even by arbitrarily limiting the number of candidates to be explored, the algorithm was stuck after 5 to 6 rounds.

4.2 Modified Search Algorithm

Matsui's branch-and-bound method is primarily concerned with the analysis of the S-boxes in order to obtain linear approximations with high biases. However, the previously described failure suggests that in Serpent, the search for optimal linear characteristics is severely slowed down by the avalanche effect. Therefore, limiting the number of active S-boxes in the linear approximations could help to improve the search efficiency. This observation led us to modify the branch-and-bound algorithm. In our modified method, we start by exhaustively counting all the couples (input, output) of the linear transform for which the number of active S-boxes is arbitrarily low. Such couples are called *transform candidates* and are entirely defined by the number of active input/output S-boxes, the position of these S-boxes and their corresponding mask value. The transform candidates are then stored in a database (*e.g.* a hash table) so that all the candidates having the same active output S-boxes are stored in the same list.

Once the database is created, we launch the actual approximation search: a linear approximation on r rounds can be obtained by the concatenation of r transform candidates, if the positions of the active S-boxes at the exit of a transform candidate correspond to those of the active S-boxes at the input of the next transform candidate (*i.e.* we have to fulfill boundary conditions, as in the original method). These constraints are easily checked, as the candidates are picked up in the database according to the position of their active output S-boxes. To calculate the bias of an approximation, we simply apply the piling up lemma to the S-boxes approximations generated between the transform candidates. We can then determine the best approximations by means of a traditional branch-and-bound, where the only difference is that we pile up the transform candidates instead of piling up round approximations. Note that we pile up the candidates starting with the last round, then going down gradually until the first round, in order to benefit from the knowledge of best bias in the branch-and-bound.

In order to improve the flexibility of the approximation search, the input mask in the first round and the output mask in the last round are chosen in order to maximize the biases of these rounds. Therefore, in a r rounds approximation, transform candidates are only picked up for the $r-1$ inner rounds, which speeds up the research. Additionally, the first round input mask and last round output mask can be replaced by any other mask, provided that the biases are left unchanged. Due to the properties of the Serpent S-boxes, for any approximation found by the algorithm, many more similar approximations can be generated by the modification of the outer masks.

4.3 Performance and Hardware Constraints

The number of possible transform candidates that one can pile up in each round is limited by the boundary conditions and the size of the lists. Moreover, among these candidates, a majority leads to a zero bias and is thus directly rejected. Indeed, a significant proportion of the Serpent S-boxes linear approximations is null: the proportion varies between $93/225 = 0.413$[1] for the S-boxes $S_0, S_1, S_2, S6$ and $81/225 = 0.36$ for the others. If there are q active S-boxes at the exit of the transform candidates, then only a fraction of roughly $(1 - 0.413)^q$ or $(1-0.36)^q$ of these candidates will lead to a non-zero bias. As this proportion falls when q increases, there is no exponential increase of the number of candidates anymore.

Nevertheless, the major drawback of the proposed method remains in the consequent size of the database used. Even if the proportion of useful transform candidates is weak, the number of candidates stored can easily reach several millions. In an optimized structure, a transform candidate with q active S-boxes requires $(10 + 9 * q)$ bits of memory[2]. If one considers a reasonable average of 16 active S-boxes per candidate, it yields 154 bits, *i.e.* 54400 candidates per megabyte, or approximately $55 \cdot 10^6$ candidates per gigabyte. Additionally, it is of course possible that the algorithm does not find any linear approximation beyond a certain number of rounds, either because the list of candidates checking the boundary conditions is empty, or because all these candidates lead to a zero bias. The higher the number of transform candidates, the lower the risk.

On the positive side, this database can be precomputed before the execution of the branch-and-bound. Once it is created, the execution of the algorithm is very fast since in each stage of the branch-and-bound, we only consult tables and calculate biases. In our experimentations, we generated the database as follows: we exhaustively generated all the possible transform candidates with maximum i $(1 \leq i \leq 5)$ active input S-boxes and stored only those with maximum $(15 - i)$ active output S-boxes. This required the analysis of approximately $3 \cdot 10^{11}$ linear transformations[3] among which only about $130 \cdot 10^6$ were stored/kept. With about 10^6 analyzes per second, this search took roughly 4 days on a 2GHz processor[4]. The exhaustive search of all the transform candidates with $i = 6$ would require the analysis of approximately $21 \cdot 10^{12}$ transformations, *i.e.* roughly 250 days of computation on the same processor. Better strategies could probably be investigated, taking advantage of the linearity of the diffusion layer, and are a scope for further research. Note that the database would not be the same if differential approximations were to be found [2,9] although it would be strongly correlated.

[1] As $\exists$ 132 approximations with non-zero biases among the 15*15 non-trivial ones.

[2] Namely 2*5 bits to store the number of input/output active S-boxes (1...32) and 9 bits to store the 5-bit position (1...32) and 4-bit mask (0...15) of each active S-box.

[3] That is $2 \cdot \sum_{k=1}^{5} 15^k * C_{32}^k$.

[4] Note that the task can be parallelized.

5 Practical Results

5.1 Best Approximation Found

Since the best 9-round characteristic found by Biham *et al.* involves at most 15 active S-boxes at the extremity of the linear transform [3], our previously described database ensures us to find an approximation with a bias at least as high. Table 2 summarizes the results of our approximations search, in function of the starting S-box (and therefore round). The best biases are given in the penultimate column and the values of Biham *et al.* are in the last column. As a first result of our investigations, we found an improved approximation starting with S_3. Our improved characteristic is very similar to the one in [3], excepted for the three first rounds. It involves a reduction of the linear cryptanalysis data complexity by a factor of 16. Due to a lack of room, the description of the linear approximations used in our attacks are not given in this paper. It is available at: http://www.dice.ucl.ac.be/crypto/publications/2007/inscrypt_appendix.pdf.

Table 2. Biases of the best 9-round approximations for different starting S-boxes

r	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	Max	$\overline{B_r}$
3	2^{-8}	2^{-8}	2^{-8}	2^{-8}	2^{-8}	2^{-7}	2^{-8}	2^{-7}	2^{-7}	2^{-7}
4	2^{-13}	2^{-14}	2^{-13}	2^{-16}	2^{-12}	2^{-12}	2^{-12}	2^{-12}	2^{-12}	2^{-12}
5	2^{-26}	2^{-23}	2^{-21}	2^{-23}	2^{-17}	2^{-18}	2^{-18}	2^{-18}	2^{-17}	2^{-18}
6	–	–	–	2^{-30}	2^{-23}	2^{-25}	2^{-24}	2^{-33}	2^{-23}	2^{-25}
7	–	–	–	2^{-36}	2^{-30}	2^{-32}	–	–	2^{-30}	2^{-32}
8	–	–	–	2^{-43}	2^{-37}	–	–	–	2^{-37}	2^{-39}
9	–	–	–	2^{-50}	–	–	–	–	2^{-50}	2^{-52}

Note that we also tried to find iterative approximations, *i.e.* approximations of which the input masks fulfill the boundary conditions of their own output. Such approximations are useful in practice because they can be straightforwardly extended to an arbitrary number of rounds [7]. However, our restricted database did not allow us to find any such approximation on 8 rounds. Searching over larger databases would therefore be necessary to further investigate the existence of good iterative characteristics.

5.2 Multiple Linear Approximations Found

In addition to the previously reported characteristic, we ran our search algorithm in order to generate a list of useful approximations for a multiple linear cryptanalysis of Serpent. As for the previous section, this generation is very fast due to the properties of the branch-and-bound that allows an effective limitation of the candidates to explore. Table 3 reports the bias distribution of the 150 best 9-rounds linear approximations of Serpent found with our database (starting with S_3). Additionally, since each of these approximations can generate several others by simply replacing its input and output masks (as discussed in Section 4.2), we straightforwardly obtained the distribution in table 4.

Table 3. Bias distribution of the 150 best approximations found

bias	# of approx.	bias	# of approx.
2^{-50}	3	2^{-53}	42
2^{-51}	12	2^{-54}	66
2^{-52}	27	2^{-55}	

Table 4. Bias distribution and cumulative capacities of the extended approximations

bias	# of approx.	capacity	bias	# of approx.	capacity
2^{-50}	786432	$2.482 \cdot 10^{-24}$	2^{-55}	$1.208 \cdot 10^{13}$	$5.184 \cdot 10^{-20}$
2^{-51}	$6.134 \cdot 10^{7}$	$5.087 \cdot 10^{-23}$	2^{-56}	$1.250 \cdot 10^{14}$	$1.481 \cdot 10^{-19}$
2^{-52}	$2.290 \cdot 10^{9}$	$5.024 \cdot 10^{-22}$	2^{-57}	$1.059 \cdot 10^{15}$	$3.520 \cdot 10^{-19}$
2^{-53}	$5.447 \cdot 10^{10}$	$3.188 \cdot 10^{-21}$	2^{-58}	$7.513 \cdot 10^{15}$	$7.138 \cdot 10^{-19}$
2^{-54}	$9.281 \cdot 10^{11}$	$1.463 \cdot 10^{-20}$	2^{-59}	$4.553 \cdot 10^{16}$	$1.262 \cdot 10^{-18}$

5.3 Resulting Capacity and Discussion of the Results

According to the framework in [4], the use of multiple approximations in linear cryptanalysis allows decreasing the number of plaintexts needed for a successful key recovery proportionally to the capacity of the obtained system (given in Table 4). It involves the following observations:

– The data complexity of the simple linear cryptanalysis of 10-rounds Serpent is approximately 2^{100} (using the best 9-round approximation with bias 2^{-50}).
– If the 786432 approximations with bias 2^{-50} are used, the resulting capacity equals $2.482 \cdot 10^{-24}$, which yields a theoretical data complexity of $2^{78.4}$.
– Cumulatively using all the $2.29 \cdot 10^{9}$ approximations of bias higher than 2^{-52}, we could theoretically reach a capacity of $5.024 \cdot 10^{-22}$, which would correspond to $2^{70.8}$ plaintext-ciphertext pairs.
– The more realistic use of 2048 (*resp.* $1.802 \cdot 10^{6}$) approximations with bias 2^{50} (*resp.* greater than 2^{52}) involving the same target subkey would result in a theoretical data complexity of 2^{87} (*resp.* 2^{81}).

As a matter of fact, these results do not take the size of the target subkey (and therefore the time complexity) into account but only consider the data complexity. In the next section, we propose more realistic attacks presenting a better trade-off between data and time complexities. Let us also mention that a possibly more powerful way to exploit multiple approximations would be to consider Matsui's *algorithm 1* and therefore avoid the time complexity problems related to key guesses, *e.g.* using the three 2^{-50} bias approximations and their derivatives. Since each approximation reveals up to one bit of information on the secret key and $m \gg 128$, the resulting linear system is strongly overdefined.

In general, the previous results have to be tempered by the possible influences of dependencies between the various linear approximations exploited in an attack. This is specially true in our context since our approximations are all generated from an initial set of 150 characteristics and the number of derivatives is much larger than $2 \cdot 128$. As discussed in [4], it is actually hard to determine the consequence of these dependencies on the capacity of the multiples approximations. Because $2 \cdot m \cdot \epsilon \ll 1$ (for any reasonable choice for the number m of approximations), the dependencies between the text masks (χ_P^i, χ_C^i) should have a negligible influence on the capacity. Therefore, the major question relates to the dependencies between the linear trails. As a matter of fact, the estimated data complexities in our analysis (as well as in Biryukov *et al.*'s) are fairly optimistic and an important next step in the understanding of linear cryptanalysis would be to experiment these predictions with a real life cipher.

6 Realistic Attack Scenarios Against Serpent

In this section, we present realistic attack scenarios on reduced round Serpent using (multiple) linear cryptanalysis. The reduced version is just like Serpent, excepted for its reduced number of rounds. After the last S-box, the linear transformation is omitted as it has no cryptographic significance. The linear approximations used were generated with the algorithm presented before.

Using an approximation on $r - 1$ rounds, one can recover bits of the subkey in round r. In Matsui's original method, a partial decryption of the last round is performed for every ciphertext by guessing the key bits involved in the approximation. The parity of the approximation for the plaintext and the partially decrypted ciphertext is then evaluated and a counter corresponding to each key guess is incremented if the relation holds or decremented otherwise. The key candidate with the highest counter in absolute value is then assumed to be the correct key. However, as we only consider a limited number of bits k (in the active S-boxes) during the partial decryption of the ciphertexts, the same pattern for these k bits possibly appear several times during the attack. In order to avoid doing the same partial decryption work several times, Biham *et al.* proposed in [3] an improvement which considerably reduces the time complexity of an attack:

- Initialize an array of 2^k counters (k is the size of the target subkey).
- *For each generated ciphertext:* extract the k-bit value corresponding to the active S-boxes and evaluate the parity of the plaintext subset defined by the approximation. Increment or decrement a counter corresponding to the extracted k-bit value according to the obtained parity.
- *Once all the ciphertexts have been generated:* for each k-bit ciphertext and for each k-bit subkey, partialy decrypt the k-bit ciphertext under the k-bit subkey and evaluate the parity of the output subset (as defined by the linear approximation). Keep this value in a table of size $2^k \cdot 2^k$.

– For each k-bit subkey, evaluate its experimental bias by checking, for each k-bit ciphertext, the parity of the approximation and the value of the corresponding counter. Then output the subkey with maximal bias.

This method reduces the attack time complexity from $O(N \cdot 2^k)$ to $O(2^k \cdot 2^k)$.

6.1 Attack on 7 Rounds Serpent

The attack uses a 6-round approximation starting with S-box 4 and ending with S-box 1. This approximation is derived from the best 6-round approximation found in section 5 except a small change in the last round that reduces the number of active S-boxes from 13 to 5. The bias of the approximation falls from 2^{-23} to 2^{-25}. Using only one approximation, the data complexity is approximately 2^{52} known plaintexts. The attack requires 2^{20} counters and a time complexity of approximately 2^{40} decryptions of 1-round Serpent. Several similar approximations can be obtained by changing the input mask of the relation. We found up to 8 approximations with bias 2^{-25} and up to 96 approximations with bias 2^{-26}. This would lead to a capacity of respectively 2^{-45}, 2^{-43}, reducing the data complexity to 2^{47} or 2^{45} at the cost of a slight increase of the time/memory complexities.

6.2 Attack on 8 Rounds Serpent

Similarly, we can attack 8-round Serpent using a 7-round approximation starting with S-box 4 and ending with S-box 2. The approximation is the one found in section 5. It has a bias of 2^{-30} and 7 active S-boxes. Consequently, an attack using only one approximation requires 2^{56} 1-round decryptions, 2^{28} counters and $\simeq 2^{62}$ known plaintexts. We can again reduce the data complexity by taking advantage of multiple approximations. We found 8 approximations with the same bias and the same active S-boxes giving a capacity of 2^{-55}, thus a data complexity of approximately 2^{57} plaintexts. Adding 96 approximations with bias 2^{-31}, we obtain a capacity of 2^{-53} and therefore a data complexity of 2^{55} plaintexts (see Table 6). Again, this effect can be increased at the cost of more memory and computation. For example, there are 384 approximations with bias 2^{-32} and 512 approximations with bias 2^{-33}, that gives rise to data complexities of respectively $2^{54.1}$ or 2^{54}, but requires 2^{28} counters and 2^{56} memory access for each counter.

6.3 Attack on 9 Rounds Serpent

The best approximation found on 8 rounds has a bias of 2^{-37} but it has 23 actives input S-boxes. We can slightly modify its input in order to lower this number to 11 active S-boxes. This way, the bias of the approximation is 2^{-39} instead of 2^{-37}. This approximation starts with S-box 4 and ends with S-box 3.

The complexity of an attack based on this approximation is 2^{88} 1-round decryptions, 2^{44} counters and about 2^{80} known plaintexts. Multiples approximations allow us to decrease the number of texts needed. We found 128 approximations with bias 2^{-39}, 3584 approximations with bias 2^{-40}, 43008 approximations with bias 2^{-41} and 286720 approximations with bias 2^{-42}. The corresponding capacities are 2^{-69}, 2^{-66}, $2^{-64.1}$, 2^{-63}, thus requiring 2^{71}, 2^{68}, $2^{66.1}$, 2^{65} generated plaintexts.

6.4 Attack on 10 Rounds Serpent

We finally ran our search algorithm to generate a list of 150 9-round approximations with high bias and a reasonable number of active S-boxes. Among the the huge number of candidates (see table 4), we found the three following approximations starting and finishing with S-box 3:

- Approximation 1 with bias 2^{-55} and 11 active S-boxes,
- Approximation 2 with bias 2^{-58} and 10 active S-boxes,
- Approximation 3 with bias 2^{-59} and 8 active S-boxes.

Using the first approximation, we obtain an attack requiring 2^{112} texts, 2^{44} counters and 2^{88} decryptions. Using the second approximation, we obtain an attack requiring 2^{118} texts, 2^{40} counters and 2^{80} decryptions. Using the third approximation, we obtain an attack requiring 2^{120} texts, 2^{32} counters and 2^{64} decryptions. Multiple linear cryptanalysis based on the first approximation leads to capacities equal to 2^{-97}, $2^{-93.42}$ or $2^{-90.93}$ according to bias of the approximations. Using the second approximation, the capacities decrease to 2^{-103}, $2^{-99.42}$ or $2^{-96.93}$. With the third approximation the capacities then become 2^{-105}, $2^{-101.42}$ or $2^{-98.93}$. All the presented attack results are summarized in Table 6 and Table 5 remembers the previously known attacks against Serpent.

6.5 Attack on 11 Rounds Serpent

We can attack 11 round Serpent with a 9-rounds linear approximation. Such an attack requires a partial encryption before the first round of the approximation and a partial decryption after the last round. In this context, it becomes essential to minimize the total number of active S-boxes, both in input and in output.

Our algorithm provided a 9-rounds approximation with a bias of 2^{-58} and only 27 active S-boxes (15 in input and 12 in output). Using the trick proposed in [3], section6, we obtain a time complexity of $2^{88} + 2^{60} \cdot (2^{118} + 2^{88}) = 2^{178}$ and a memory complexity of 2^{88}. The data complexity is left unchanged, that is 2^{118} known plaintext.

In this case, it is practically not possible to use multiples approximations, as they should have at least the same active S-boxes, both in their input and output.

6.6 Summary

Table 5. Summary of previous attacks on Reduced-rounds Serpent (see [15])

Rounds	Type of attack	complexity		
		data	time	memory
6	differential [13]	2^{83}CP	2^{90}	2^{44}
	differential [13]	2^{71}CP	2^{103}	2^{79}
	differential [13]	2^{41}CP	2^{163}	2^{49}
7	differential [11]	2^{84}CP	$2^{78.9}$	2^{56}
8	Amp.Boomerang [13]	2^{128}CP	2^{163}	2^{137}
	Amp.Boomerang [13]	2^{110}CP	2^{175}	2^{119}
	differential [11]	2^{84}CP	$2^{206.7}$	2^{89}
9	Amp.Boomerang [13]	2^{110}CP	2^{252}	2^{212}
10	Rectangle [14]	$2^{126.3}$CP	2^{165}	$2^{131.8}$
	Boomerang [14]	$2^{126.3}$ACPC	2^{165}	2^{89}
	Lin.Cryptanalysis [3]	2^{116}KP	2^{92}	2^{45}
	Diff.Lin.Cryptanalysis [15]	$2^{105.2}$CP	$2^{123.2}$	2^{40}
11	Lin.Cryptanalysis [3]	2^{118}CP	$2^{205.7}$	2^{183}
	Diff.Lin.Cryptanalysis [15]	$2^{125.3}$CP	$2^{172.4}$	2^{30}
	Diff.Lin.Cryptanalysis [15]	$2^{125.3}$CP	$2^{139.2}$	2^{60}
Complexity is measured in encryption units. Memory is mesured in Bytes. CP - Chosen Plaintexts, KP - Known Plaintexts, ACPC - Adaptive Chosen Plaintexts and Ciphertext.				

Table 6. Summary of attacks on Serpent presented in this paper

Rounds	Type of attack	complexity		
		data	time	memory
7	Lin.cryptanalysis	2^{52}KP	2^{40}	2^{20}
	Mult.Lin.Cryptanalysis (8 appr.)	2^{47}KP	2^{43}	2^{23}
8	Lin.cryptanalysis	2^{62}KP	2^{56}	2^{28}
	Mult.Lin.Cryptanalysis (8 appr.)	2^{57}KP	2^{59}	2^{31}
	Mult.Lin.Cryptanalysis (104 appr.)	2^{55}KP	$2^{62.7}$	$2^{34.7}$
9	Lin.cryptanalysis	2^{80}KP	2^{88}	2^{44}
	Mult.Lin.Cryptanalysis (128 appr.)	2^{71}KP	2^{95}	2^{51}
	Mult.Lin.Cryptanalysis (3712 appr.)	2^{68}KP	$2^{99.9}$	$2^{55.9}$
10	Lin.cryptanalysis ($\epsilon = 2^{-55}$)	2^{112}KP	2^{88}	2^{44}
	Mult.Lin.Cryptanalysis (2048 appr.)	2^{99}KP	2^{99}	2^{55}
	Lin.cryptanalysis ($\epsilon = 2^{-59}$)	2^{120}KP	2^{64}	2^{32}
	Mult.Lin.Cryptanalysis (2048 appr.)	2^{107}KP	2^{75}	2^{43}
11	Lin.cryptanalysis ($\epsilon = 2^{-58}$)	2^{118}KP	2^{178}	2^{88}

7 Conclusion and Further Works

This paper first presents a modification of Matsui's branch-and-bound algorithm for the linear approximation search in a block cipher. It enabled us to find the best reported 9-round approximation for the AES candidate Serpent. The algorithm allows speeding up the search of linear approximations at the cost of larger memory requirements. It is generic and especially well suited for ciphers where the linear transformation involves an important avalanche effect (as the AES candidates and most recent ciphers). Moreover, it could be straightforwardly adapted for the research of differential characteristics.

In a second part of the paper, we take advantage of this modified branch-and-bound algorithm in order to investigate the possible use of multiple linear approximations against Serpent. According to the framework of Biryukov *et al.* [4], we provided estimations of the improved data complexity of such attacks against 10-round Serpent that can be down to approximately 2^{80}. Since these results are mainly theoretical (due to an unrealistic time complexity), we also presented several attacks against 7- to 10-round Serpent using reasonable attack parameters that outperform previously known results.

As an important scope for further research, these results should be experimented against real-life ciphers of tractable size in order to determine the actual influence of dependencies between the different approximations used in an attack. That is, to figure out the extend to which the information provided by multiple linear approximations can lead to efficient attack strategies.

Acknowledgements

The authors thank Orr Dunkelman for his helpful answers and advices. We also thank anonymous reviewers from FSE and Inscrypt for their useful comments.

References

1. Anderson, R., Biham, E., Knudsen, L.: Serpent: A Proposal for the Advanced Encryption Standard. In: The proceedings of the First Advanced Encryption Standard (AES) Conference, Ventura, CA (1998)
2. Biham, E.: On Matsui's Linear Cryptanalysis. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 341–355. Springer, Heidelberg (1995)
3. Biham, E., Dunkelman, O., Keller, N.: Linear Cryptanalysis of Reduced Round Serpent. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 16–27. Springer, Heidelberg (2002)
4. Biryukov, A., De Cannière, C., Quisquater, M.: On Multiple Linear Approximations. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 1–22. Springer, Heidelberg (2004)
5. Biryukov, A.: Linear Cryptanalysis. In: The Encyclopedia of Cryptography and Security, Kluwer Academic Publishers, Dordrecht (2005)
6. Kaliski, B.S., Robshaw, M.J.B.: Linear Cryptanalysis using Multiple Approximations. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 26–39. Springer, Heidelberg (1994)

7. Knudsen, L.R.: Iterative characteristics of DES and s^2-DES. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 497–511. Springer, Heidelberg (1993)
8. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
9. Matsui, M.: On Correlation Between the Order of S-boxes and the Strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366–375. Springer, Heidelberg (1995)
10. Ohta, K., Moriai, S., Aoki, K.: Improving the Search Algorithm for the Best Linear Expression. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 157–170. Springer, Heidelberg (1995)
11. Biham, E., Dunkelman, O., Keller, N.: The Rectangle Attack - Rectangling the Serpent. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 340–357. Springer, Heidelberg (2001)
12. Kohno, T., Kelsey, J., Schneier, B.: Preliminary Cryptanalysis of Reduced-Round Serpent. In: AES Candidate Conference, pp. 195–211 (2000)
13. Kelsey, J., Kohno, T., Schneier, B.: Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In: Proceedings of Fast Software Encryption 7, LNCS, vol. 1978, pp. 75–93. Springer, Heidelberg (1999)
14. Biham, E., Dunkelman, O., Keller, N.: New Results on Boomerang and Rectangle Attacks. In: The Proceedings of Fast Software Encryption 9. LNCS, vol. 2501, pp. 254–266. Springer, Heidelberg (2002)
15. Biham, E., Dunkelman, O., Keller, N.: Differential-linear Cryptanalysis of Serpent. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 9–21. Springer, Heidelberg (2003)

Linear Slide Attacks on the KeeLoq Block Cipher[⋆]

Andrey Bogdanov

Chair for Communication Security
Ruhr-University Bochum, Germany
abogdanov@crypto.rub.de
www.crypto.rub.de

Abstract. KeeLoq is a block cipher used in numerous widespread passive entry and remote keyless entry systems as well as in various component identification applications. The KeeLoq algorithm has a 64-bit key and operates on 32-bit blocks. It is based on an NLFSR with a nonlinear feedback function of 5 variables.

In this paper new key recovery attacks on KeeLoq are proposed. The first one has a complexity of about $2^{50.6}$ KeeLoq encryptions. The second attack finds the key in 2^{37} encryptions and works for the whole key space. In our attacks we use the techniques of guess-and-determine, slide, and linear attacks as well as cycle structure analysis. Both attacks need 2^{32} known plaintext-ciphertext pairs.

We also analyze the KeeLoq key management and authentication protocols applied in rolling-code and IFF access systems widely used in real-world applications. We demonstrate several practical vulnerabilities.

Keywords: KeeLoq, cryptanalysis, slide attacks, linear cryptanalysis, hopping codes, rolling codes, authentication protocols, identify friend-or-foe, key generation.

1 Introduction

KeeLoq is a block cipher based on an NLFSR with a nonlinear boolean feedback function of 5 variables. The algorithm uses a 64-bit key and operates on 32-bit blocks. Its architecture consists of two registers (a 32-bit text register and a 64-bit key register), which are rotated in each of 528 encryption cycles, and of a nonlinear function (NLF) providing nonlinear feedback. One bit of the key is added to the output of the NLF modulo 2 in each cycle.

The light-weight architecture of the KeeLoq cipher allows for an extremely low-cost and efficient hardware implementation (about 700 GE and 528 clock cycles per block). This contributed to the popularity of the KeeLoq cipher among designers of remote keyless entry systems, automotive and burglar alarm systems, automotive immobilizers, gate and garage door openers, identity tokens,

[⋆] This is a short version of the full work [1] on the analysis of KeeLoq systems presented at the 3rd Conference on RFID Security (RFIDSec'07) in Malaga, Spain.

component identification systems. For instance, the KeeLoq block cipher is used by such automotive OEMs as Chrysler, Daewoo, Fiat, GM, Honda, Toyota, Volvo, VW, Jaguar [2] and in the HomeLink wireless control systems to secure communication with garage door openers [3]. The KeeLoq technology supplied by Microchip Technology Inc. includes the KeeLoq cipher and a number of authentication protocols as well as key management schemes. Our description of KeeLoq is based on the newly published article [2], [4] and a number of the manufacturer's documents [5], [6], [7], [8].

Our contribution. The contribution of the paper is many-fold. First, a new technique to perform recovery attacks on KeeLoq is proposed. Our direct attack recovers the key in $2^{50.6}$. Second, the techniques allow us to propose an extended attack of complexity 2^{37} working for the whole key space. Then severe vulnerabilities of the KeeLoq protocols and key management systems are demonstrated.

Our cryptanalysis of the KeeLoq algorithm is based on the following weaknesses of the KeeLoq structure: The key schedule is self-similar, which allows us to mount a slide attack [9], [10], [11]. It is supported by the existence of an efficient linear approximation of the NLF used to recover a part of the key. Then the remainder of the key bits is obtained using other linear relations within KeeLoq.

The key recovery complexity of our first attack is $2^{50.6}$. The attack requires 2^{32} plaintext-ciphertext pairs and a memory of 2^{32} 32-bit words. Several computing devices can share the memory during the attack. All computations are perfectly parallelizable. The property inherited from the slide attacks [9], [10] is that the complexity of our attack is independent of the number of encryption cycles, which is as a rule not the case for linear or differential cryptanalysis, where the complexity often grows exponentially with the number of iterations.

The second attack is an extension of our first attack by using the cycle structure analysis introduced in [12]. Our attack finds the key in 2^{37} steps. It also requires 2^{32} known plaintext-ciphertext pairs and a memory to hold 2^{32} 32-bit words. Additionally, 2^{32} bits of memory are needed for exploring the cycle structure of the KeeLoq permutation. Our techniques work for all keys, unlike those in [12] which are applicable to 26% of all keys only. Our second attack is the best known attack on the KeeLoq block cipher working for the whole key space.

We also show how the cryptanalytic attacks on the KeeLoq block cipher apply to the KeeLoq hopping codes and IFF (Identify Friend or Foe) systems supplied by Microchip which are based on this algorithm. It is demonstrated that the attacks pose a real threat for a number of applied key management schemes: After attacking one instance of KeeLoq using attacks presented in this paper, one can reduce the effective key length of all other KeeLoq systems of the same series to 32, 48, or 60 bits depending on the key management scheme applied. In some attack models, the attacker is even able to retrieve the individual encryption key instantly.

The paper is organized as follows. Section 2 describes the KeeLoq algorithm. In Section 3 our basic linear slide key recovery attack on KeeLoq and its extension with a reduced complexity are presented. In Section 4 we discuss the impact of attacks on the KeeLoq algorithm with respect to the standard KeeLoq real-word applications supplied by Microchip. We conclude in Section 5.

2 Description of the KeeLoq Algorithm

KeeLoq is a block cipher with a 64-bit key which operates on 32-bit words [2], [4]. Its design is based on a nonlinear feedback shift register (NLFSR) of length 32 bits with a nonlinear feedback function of 5 variables. The feedback depends linearly on two other register bits and on the next key bit taken from the rotated key register of length 64 bits.

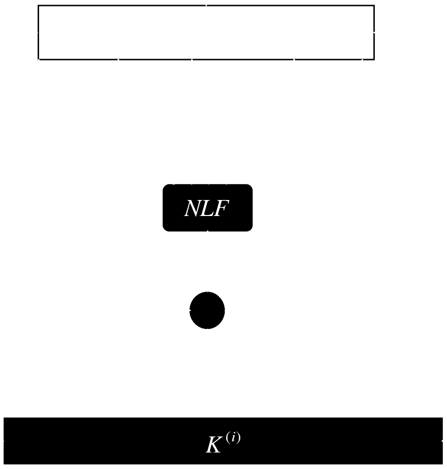


Fig. 1. The i-th KeeLoq encryption cycle

Let $V_n = \mathrm{GF}(2)^n$ be the set of all n-bit words and $Y^{(i)} = (y_{31}^{(i)}, \ldots, y_0^{(i)}) \in V_{32}$, $y_j^{(i)} \in \mathrm{GF}(2)$, describe the state of the text register in cycle i for $j = 0, \ldots, 31$ and $i = 0, 1, \ldots$ Let also $K^{(i)} = (k_{63}^{(i)}, \ldots, k_0^{(i)}) \in V_{64}$, $k_j^{(i)} \in \mathrm{GF}(2)$, denote the state of the key register in cycle i for $j = 0, \ldots, 63$ and $i = 0, 1, \ldots$ Then each cycle of encryption can be described using the following algorithm (see Figure 1):

Compute the feedback bit: $\varphi = NLF(y_{31}^{(i)}, y_{26}^{(i)}, y_{20}^{(i)}, y_9^{(i)}, y_1^{(i)}) \oplus y_{16}^{(i)} \oplus y_0^{(i)} \oplus k_0^{(i)}$
Rotate text and insert feedback: $R^{(i+1)} = (\varphi, y_{31}^{(i)}, \ldots, y_1^{(i)})$
Rotate key: $K^{(i+1)} = (k_0^{(i)}, k_{63}^{(i)}, \ldots, k_1^{(i)})$.

For encryption the key register is filled with the 64 key bits $K = (k_{63}, \ldots k_0) \in V_{64}$, $k_j \in \mathrm{GF}(2)$, $j = 0, \ldots, 63$, in the straightforward way: $K^{(0)} = K$. If $X =$

$(x_{31}, \ldots, x_0) \in V_{32}$, $x_j \in \mathrm{GF}(2)$, $j = 0, \ldots, 31$, is a block of plaintext, the initial state of the text register is $Y^{(0)} = (x_{31}, \ldots, x_0)$. The output of the algorithm is the ciphertext $Z = (z_{31}, \ldots, z_0) = Y^{(528)} \in V_{32}$, $z_j \in \mathrm{GF}(2)$, $j = 0, \ldots, 31$.

For decryption the key register is filled in the same way: $K^{(0)} = K = (k_{63}, \ldots k_0) \in V_{64}$. But the decryption procedure complements the encryption. One decryption cycle can be defined by the following sequence of operations:

Compute the feedback bit: $\varphi = NLF(y_{30}^{(i)}, y_{25}^{(i)}, y_{19}^{(i)}, y_8^{(i)}, y_0^{(i)}) \oplus y_{15}^{(i)} \oplus y_{31}^{(i)} \oplus k_{15}^{(i)}$
Rotate text and insert feedback: $R^{(i+1)} = (y_{30}^{(i)}, \ldots, y_0^{(i)}, \varphi)$
Rotate key: $K^{(i+1)} = (k_{62}^{(i)}, \ldots, k_0^{(i)}, k_{63}^{(i)})$.

The ciphertext and plaintext are input/output in a similar way: The ciphertext is input into the text register before decryption, $Y^{(0)} = Z$, and the plaintext can be read out after 528 decryption cycles, $Y^{(528)} = X$.

The NLF is a boolean function of 5 variables and is of degree 3. In the specification [4] the NLF is assigned using a table. This corresponds to the following ANF:

$$\begin{aligned} NLF(x_4, x_3, x_2, x_1, x_0) = x_0 \oplus x_1 \oplus \\ x_0x_1 \oplus x_1x_2 \oplus x_2x_3 \oplus x_0x_4 \oplus x_0x_3 \oplus x_2x_4 \oplus \\ x_0x_1x_4 \oplus x_0x_2x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4. \end{aligned} \tag{1}$$

The NLF is balanced and its correlation immunity order is 1, $\mathrm{cor}(NLF) = 1$ [13], [14]. This means that the NLF is 1-resilient [15], which is the maximum for a function of 5 variables with $\deg(NLF) = 3$ due to Siegenthaler's inequality [13]:

$$\deg(NLF) + \mathrm{cor}(NLF) \leq 4.$$

$Y^{(0)}$
$F(K)$
$Y^{(512)}$
$Y^{(528)}$

Fig. 2. Round structure of KeeLoq encryption

The KeeLoq algorithm has the following round structure. We define a KeeLoq round as the permutation $F(K) : V_{32} \to V_{32}$ depending on the key $K \in V_{64}$. A KeeLoq quarter round is defined as the permutation $F'(K') : V_{32} \to V_{32}$ depending on the subkey $K' = (k_{15}, \ldots, k_0) \in V_{16}$. Then the whole KeeLoq encryption mapping consists of successively computing 8 full round permutations $F(K)$ and consequently applying the last quarter round permutation $F'(K')$, see Figure 2. Note that the first 8 full rounds are identical. The decryption can be represented in a similar way using inverse permutations $F'(K')^{-1}$ and $F(K)^{-1}$.

The algorithm allows for an extremely simple hardware implementation comprised of a 32-bit shift register with taps on fixed positions, a 64-bit shift resister with a single tap and a 32-bit (5×1) look-up table (LUT) for the NLF. The LUT can be replaced with the corresponding logical elements according to (1).

In this case the hardware implementation of KeeLoq requires about 700 GE and 528 clock cycles per block, which is to be compared to about 1500-2000 GE needed for modern light-weight hardware-oriented stream ciphers such as Grain or Trivium.

3 Attacks on the KeeLoq Algorithm

3.1 Basic Linear Slide Attack on KeeLoq

Our basic attack is based on the following weaknesses of the algorithm: self-similar key schedule scheme, relatively short blocks of 32 bits, and existence of an efficient linear approximation of the NLF.

The attack can be outlined in the following way. For each subkey $K' = (k_{15}, \ldots, k_0)$ and for a random 32-bit input $I_0 \in V_{32}$ guess the corresponding output $O_0 \in V_{32}$ after the 64 clock cycles which depends on the other 48 key bits $(k_{63}, \ldots, k_{16})$. Using the periodic structure of the KeeLoq key schedule generate several other pairs $(I_i, O_i) \in (V_{32})^2$, $i = 1, \ldots, N-1$ (*sliding step*). For a successful attack N has to be about 2^8. For each number of such pairs we mount a distinguishing attack to obtain linear relations on some unknown key bits with a high probability due to the fact that the KeeLoq NLF is not 2-resilient (*linear correlation step*). In this way it is possible to determine $(k_{47}, \ldots, k_{16})$ bit by bit. After this an input/output pair for 16 encryption cycles can be represented as a triangular system of linear equations with the remaining bits $(k_{63}, \ldots, k_{48})$ of K as variables. It can be solved using 16 simple computational operations (*linear step*).

Sliding step. Using a single input/output pair (I_0, O_0) for the full round of 64 cycles and knowing the first 16 key bits $K' = (k_{15}, \ldots, k_0)$ one can produce an arbitrary number of other input/output pairs for this round. This is possible due to the fact that (almost) all rounds in KeeLoq are identical permutations which is the property on which the slide attacks by Biryukov and Wagner are substantially based [9], [10]. Once a pair (I_0, O_0) is known, the next input/output pair is produced by encrypting I_0 and O_0 with the key to be recovered (it is a chosen plaintext attack) and obtaining (I'_1, O'_1) as ciphertext. Then I'_1 and O'_1 are decrypted using the guessed partial key $K' = (k_{15}, \ldots, k_0)$. The resulting plaintexts form the needed pair (I_1, O_1), see Figure 3. From one pair (I_i, O_i), $i = 0, 1, 2, \ldots$, an arbitrary number of pairs (I_j, O_j), $j > i$ can be derived for a certain value of K' by iteratively encrypting I_i, O_i using KeeLoq and decrypting them with K' (thus, obtaining (I_{j+1}, O_{j+1})). We call the set of pairs (I_i, O_i) needed for determining the key a *pseudo-slide group.*

As the sliding has to be performed for each guess of $K' = (k_{15}, \ldots, k_0)$ and each round output (2^{47} times on average) in our basic attack, its complexity is crucial for the efficiency of the whole attack.

I_i
$F(K)$
$F(K)$
$F(K)$
I_{i+1}
$F'(K')$
$F(K)$
$F(K)$

Fig. 3. Generating input/output pairs using sliding techniques

Correlation step. Once the pseudo-slide group was generated in the sliding step, the following weakness of the KeeLoq NLF with respect to correlation attacks is used due to the fact that the NLF is 1-resilient, but not 2-resilient.

Lemma 1. *For uniformly distributed $x_4, x_3, x_2 \in GF(2)$ the following holds:*

– $\Pr\{NLF(x_4,x_3,x_2,x_1,x_0) = 0 \mid x_0 \oplus x_1 = 0\} = \frac{5}{8}$,
– $\Pr\{NLF(x_4,x_3,x_2,x_1,x_0) = 1 \mid x_0 \oplus x_1 = 1\} = \frac{5}{8}$.

This means that the NLF can be efficiently approximated by $x_0 \oplus x_1$. So, if x_0, x_1 are known and x_4, x_3, x_2 are random and unknown, we can determine $f(K)$ by statistically filtering out the contribution of $NLF(x_4,x_3,x_2,x_1,x_0)$ to the equation

$$NLF(x_4,x_3,x_2,x_1,x_0) \oplus f(K) = 0$$

using a very limited number of such samples. $f(K)$ is a key-dependent boolean function remaining constant for all samples.

Here we show how to obtain k_{16} and k_{32} from I_i and O_i. The remaining key bits $(k_{47},\ldots,k_{33})$ and $(k_{31},\ldots,k_{17})$ can be obtain in the same way by using k_{32}, k_{16} and shifting input and output bits.

We denote $I_i = Y^{(0)}$ and $O_i = Y^{(64)}$ for each i. The idea is to make use of the correlation weakness of the dependency between the output bits $y_0^{(64)}$, $y_1^{(64)}$ and the input bits $Y^{(0)}$. One can compute $Y^{(16)}$ from $Y^{(0)}$, since $K' = (k_{15},\ldots,k_0)$ is known. For the next bit $y_{31}^{(17)}$, which is the first key-dependent bit, one has the following equation:

$$\begin{aligned} y_{16}^{(32)} = y_{31}^{(17)} &= NLF(y_{31}^{(16)}, y_{26}^{(16)}, y_{20}^{(16)}, y_9^{(16)}, y_1^{(16)}) \oplus y_0^{(16)} \oplus y_{16}^{(16)} \oplus k_{16} = \\ &= c_0 \oplus k_{16}, \end{aligned} \tag{2}$$

where $c_0 \in \mathrm{GF}(2)$ denotes the key-independent part of (2).

After 32 encryption cycles the following holds:

$$(y_{15}^{(32)}, y_{14}^{(32)}, \ldots, y_0^{(32)}) = (y_{31}^{(16)}, y_{30}^{(16)}, \ldots, y_{16}^{(16)}) \in V_{16}.$$

Thus, the least significant half of $Y^{(32)}$ is known. Then $y_0^{(64)}$ can be represented as:

$$\begin{aligned} y_0^{(64)} &= NLF(y_{31}^{(32)}, y_{26}^{(32)}, y_{20}^{(32)}, y_9^{(32)}, y_1^{(32)}) \oplus y_0^{(32)} \oplus y_{16}^{(32)} \oplus k_{32} = \\ &= NLF(y_{31}^{(32)}, y_{26}^{(32)}, y_{20}^{(32)}, y_9^{(32)}, y_1^{(32)}) \oplus y_0^{(32)} \oplus (c_0 \oplus k_{16}) \oplus k_{32}, \end{aligned} \tag{3}$$

where $y_0^{(64)}$, $y_0^{(32)}$, $y_1^{(32)}$, $y_9^{(32)}$, c_0 are known and $y_{31}^{(32)}$, $y_{26}^{(32)}$, $y_{20}^{(32)}$, k_{32}, k_{16} are unknown. As the first two inputs of the NLF are known, its contribution to (3) can be replaced with the random variate ε using Lemma 1:

$$NLF(y_{31}^{(32)}, y_{26}^{(32)}, y_{20}^{(32)}, y_9^{(32)}, y_1^{(32)}) \oplus y_9^{(32)} \oplus y_1^{(32)} = \varepsilon \tag{4}$$

with

$$\Pr\{\varepsilon = 0\} = \frac{5}{8}. \tag{5}$$

Then the following holds:

$$y_0^{(64)} \oplus y_0^{(32)} \oplus c_0 \oplus y_9^{(32)} \oplus y_1^{(32)} = \varepsilon \oplus k_{16} \oplus k_{32}. \tag{6}$$

In order to determine $k_{16} \oplus k_{32}$ one has to distinguish between the following two cases: $k_{16} \oplus k_{32} = 0$ and $k_{16} \oplus k_{32} = 1$. In the first case:

$$\Pr\{y_0^{(64)} \oplus y_0^{(32)} \oplus c_0 \oplus y_9^{(32)} \oplus y_1^{(32)} = 0\} = \frac{5}{8}.$$

Otherwise, this probability is 3/8.

Thus, the bias δ of the first random variable with respect to the second one is $\delta = \frac{1}{4}$. Our experiments show that about 2^7 equations (6) for different pairs (I_i, O_i), $i = 0, \ldots, 2^7 - 1$, are needed to recover $\alpha = k_{16} \oplus k_{32}$ with an acceptable error probability (for all 32 key-dependent linear combinations to be determined in this way), which agrees[1] with Theorem 6 of [16].

Next we consider $y_1^{(64)}$ and its dependencies from the input and key bits. Similar to (2) one has:

$$\begin{aligned} y_{16}^{(33)} &= NLF(y_{31}^{(17)}, y_{27}^{(16)}, y_{21}^{(16)}, y_{10}^{(16)}, y_2^{(16)}) \oplus y_1^{(16)} \oplus y_{17}^{(16)} \oplus k_{17} = \\ &= NLF(c_0 \oplus k_{16}, y_{27}^{(16)}, y_{21}^{(16)}, y_{10}^{(16)}, y_2^{(16)}) \oplus y_1^{(16)} \oplus y_{17}^{(16)} \oplus k_{17} = \\ &= c_1' \oplus c_2 k_{16} \oplus y_1^{(16)} \oplus y_{17}^{(16)} \oplus k_{17} = c_1 \oplus c_2 k_{16} \oplus k_{17}, \end{aligned} \tag{7}$$

where $c_1' \in \mathrm{GF}(2)$ is the free term of NLF, $c_2 \in \mathrm{GF}(2)$ is its linear term with respect to k_{16}, and $c_1 = c_1' \oplus y_1^{(16)} \oplus y_{17}^{(16)} \in \mathrm{GF}(2)$. Here c_1 and c_2 are known and depend on $Y^{(0)}$. Then the second output bit $y_1^{(64)}$ is represented as follows:

$$\begin{aligned} y_1^{(64)} &= NLF(y_{31}^{(33)}, y_{26}^{(33)}, y_{20}^{(33)}, y_9^{(33)}, y_1^{(33)}) \oplus y_0^{(33)} \oplus y_{16}^{(33)} \oplus k_{33} = \\ &= (\varepsilon \oplus y_9^{(33)} \oplus y_1^{(33)}) \oplus y_0^{(33)} \oplus (c_1 \oplus c_2 k_{16} \oplus k_{17}) \oplus k_{33}, \end{aligned} \tag{8}$$

where the random variate ε is assigned in a way similar to (4) and c_0, c_1, c_2, $y_0^{(33)}$, $y_9^{(33)}$, $y_1^{(33)}$ are known. To determine $k_{17} \oplus k_{33}$ pairs (I_i, O_i) with $c_2 = 0$ are selected[2]. Then ε in (8) is filtered out statistically, which recovers $\beta = k_{17} \oplus k_{33}$.

[1] Strictly speaking, the mentioned Theorem 6 cannot be applied here since Assumption 4 of [16] does not hold due to the fact that the mutual bias is relatively large in our case. But this suggests that our experimental estimations are correct.

[2] Note that for random inputs I_i the probability of $c_2 = 0$ is 0.5. Therefore about $N/2$ out of N known pairs (I_i, O_i) will lead to $c_2 = 0$. This raises the required number of plaintext/ciphertext pairs to about 2^8.

After this the remaining pairs (I_i, O_i) (with $c_2 = 1$) are used to obtain $\gamma = k_{16} \oplus k_{17} \oplus k_{33}$ in the same way. Thus, $k_{16} = \beta \oplus \gamma$ and $k_{32} = \alpha \oplus k_{16}$.

Now k_{16}, k_{32} and $k_{17} \oplus k_{33}$ are known. In the next step we determine $k_{18} \oplus k_{34}$ and $k_{17} \oplus k_{18} \oplus k_{34}$ using the same plaintext/ciphertext pairs (I_i, O_i) and the same statistical recovery method. In this way all 32 key bits $(k_{47}, \dots, k_{16})$ are obtained in only 16 rather simple computational steps.

Linear step and key verification. The remaining key bits $(k_{63}, \dots, k_{48}) \in V_{32}$ can be recovered as follows. As $(k_{47}, \dots, k_0)$ are known, $Y^{(48)}$ can be computed for each pair (I_i, O_i). $y_{16}^{(64)}$ can be expressed as:

$$y_{16}^{(64)} = NLF(y_{31}^{(48)}, y_{26}^{(48)}, y_{20}^{(48)}, y_9^{(48)}, y_1^{(48)}) \oplus y_{16}^{(48)} \oplus y_0^{(48)} \oplus k_{48}, \tag{9}$$

which reveals k_{48} since the entire state $Y^{(48)}$ is known. Now $Y^{(49)}$ can be completely calculated which leads to the value of k_{49} using $y_{17}^{(64)}$, and so on. In this way the rest of the key is recovered.

At the end of the key recovery procedure we expect to obtain a number of key candidates. The upper bound for their average quantity is $2^{64-32} = 2^{32}$ due to the known plaintext unicity distance [17], since the block length is 32 bit and the key length is 64 bit. Thus, we need to verify each key candidate against max. $\lceil \frac{64+4}{32} \rceil = 3$ plaintext-ciphertext pairs for all 528 encryption cycles.

Attack complexity and experiments. The attack consists of the following stages:

- Compute all plaintext-ciphertext pairs for the whole cipher;
- Guess the partial key K' and the output O_0 after one round for some input I_0;
- For each pair of guesses (K', O_0) do the following:
  - Obtain $2^8 - 1$ other pairs (I_i, O_i); thus, the cardinality of the pseudo-slide group is 2^8 for this attack;
  - Determine $k_{16} \oplus k_{32}$ by evaluating c_0 for the first 2^7 pairs of the pseudo-slide group;
  - Determine $(k_{47}, \dots, k_{16})$ by evaluating c_1 and c_2 2^8 times;
  - Determine $(k_{63}, \dots, k_{48})$ by evaluating 2^4 nonlinear boolean functions;
- Verify max. 2^{32} candidate keys using at most 3 plaintext-ciphertext pairs for the whole cipher and 3 full encryption operations.

If one step is equivalent to a single full KeeLoq encryption (528 encryption cycles), 2^{32} steps are needed for generating 2^{32} plaintext-ciphertext pairs. Each element has to be stored in a memory of 2^{32} 32-bit words.

For each guess of (I_0, O_0) and K' operations of the following complexity have to be be performed:

- $2^9 - 2$ memory accesses for obtaining (I_i, O_i), $i = 1, \dots, 2^8 - 1$. We assume a single memory access equivalent to 4 encryption cycles. This leads to approximately 2^2 steps required to perform the memory accesses.

– 2^7 evaluations of c_0 and $k_{16} \oplus k_{32}$, each evaluation being equivalent to one encryption cycle. The complexity of this stage is then $2^7/528 \approx 2^{-2}$ steps.
– $16 \cdot 2^8 = 2^{12}$ evaluations of c_1 and c_2 for determining $(k_{47}, \ldots, k_{16})$. Each evaluation is computationally equivalent to one encryption cycle. This stage requires about $2^{12}/528 \approx 2^3$ steps.
– 2^4 evaluations of a boolean function to determine $(k_{63}, \ldots, k_{48})$. Each evaluation is equivalent to one encryption cycle which leads to a complexity of about $2^4 \cdot 2^{-9} = 2^{-5}$ steps.

Max. 2^{32} candidate keys have to be verified using at most 3 full encryptions which requires max. 2^{34} steps. Thus, the overall computational complexity of the attack is

$$2^{32} + \frac{2^{32} \cdot 2^{16}}{2} \cdot (2^2 + 2^{-2} + 2^3 + 2^{-5}) + 2^{34} \approx 2^{50.6} \text{ steps.}$$

The memory complexity is quite reasonable and is 2^{32} 32-bit words (16 GByte). This enables an attacker to place all plaintext-ciphertext values into RAM which substantially accelerates the implementation of the attack. Most computations in our attack are perfectly parallelizable.

3.2 Advanced Attack on the KeeLoq Algorithm

In this subsection, we first outline some parallel work on the cryptanalysis of KeeLoq including algebraic attacks and cycle structure analysis. Then we combine our basic linear slide attack with the cycle structure analysis and obtain the best known attack on KeeLoq working for the whole key space, which requires 2^{37} KeeLoq encryptions.

Algebraic attack. Courtois and Bard [12] used the idea of sliding KeeLoq, but employed algebraic techniques to obtain the key from a slid pair. The attack requires only one slid pair. This eliminates the necessity to guess the partial key K'.

The attack works as follows. By exploring 2^{16} random known plaintext-ciphertext pairs, the attacker expects to have at least one slid pair (I_i, I'_{i+1}), (O_i, O'_{i+1}) with $O_i = F(I_i)$. This forms the first 32 equations of the non-linear system, which are not sufficient for uniquely determining the key. To obtain more equations, the attacker can use the fact that the ciphertexts I'_{i+1} and O'_{i+1} are related by $F'[F[F'^{-1}[I'_{i+1}]]] = O'_{i+1}$ (see Figure 3), which gives the other 32 binary equations. That is, the ciphertexts in the slid pair are one round (64 KeeLoq cycles) apart in the same KeeLoq cipher with K rotated by 16 bits - $(k_{16}, k_{17}, \ldots, k_{63}, k_0, \ldots, k_{15})$.

After this, the 64 equations are solved using the SAT solver MiniSat. The procedure has to be performed 2^{31} times on average (for each combination of the available plaintext-ciphertext pairs). The total complexity of the attack is about 2^{53} KeeLoq encryptions and it requires only 2^{16} known plaintexts. However, our basic attack is faster than this algebraic attack, though requiring more known plaintexts.

Cycle structure attack. The second attack from [12] makes use of the specific cycle structure imposed by the fact that the first 8 KeeLoq rounds (64 clock cycles each) are exactly the same.

For a random permutation on n-bit words, there are on average about $\ln 2^n$ cycles. That is, for $n = 32$ the expected number of cycles is about 22, approximately half of them being even. If the same permutation is applied twice, the cycles of even size split into two classes - even and odd cycles. The odd cycles of the original permutation persist.

In KeeLoq, the same permutation F, which we consider as random, is applied 8 times (8 full rounds of KeeLoq) followed by a quarter round F'. That is, the permutation $F^8(\cdot)$ has about $11/2^{\log 8} \approx 1.4$ cycles of even size[3]. At the same time, a random permutation would have about 11 cycles of even size.

Thus, one can determine the correct value of K' by decrypting all ciphertexts one quarter round and counting the number of cycles of even size. If there are more than 6 even cycles, this can be seen as a random permutation and the guess of K' is wrong. Otherwise, the guessed K' is correct. This test allows one to determine the correct value of K' with a high probability. The complexity of this step is about 2^{37} KeeLoq encryptions.

Our extended attack. Now we can determine the quarter key $K' = (k_{15}, \ldots, k_0)$ using the cycle structure attack described above. This requires 2^{32} known plaintext-ciphertext pairs and about 2^{37} KeeLoq encryptions. Then we just apply our basic attack from Section 3.1.

Actually, we do not have to perform the whole attack, as the first 16 key bits are already known. The attacker chooses two plaintext-ciphertext pairs and builds the corresponding pseudo-slide group of size 2^8. Then the correlation and linear steps of our basic attack are directly applied to determine the remaining 48 bits of the key. This operation has to be performed for approximately 2^{31} random plaintext-ciphertext pairs to achieve a high success probability. That is, the complexity of these steps is about 2^{33} KeeLoq encryptions.

Thus, we built an attack of complexity 2^{37} operations working for the whole key space and requiring 2^{32} known plaintexts. This is the fastest known attack on the KeeLoq block cipher applicable to the whole key space.

Though it might seem that the data requirements make the attack unpractical, we show in the next section that our attacks can have practical relevance due to some severe weaknesses of the key management schemes used in real-world KeeLoq systems.

4 Attacks on KeeLoq-Based Systems in Practice

4.1 KeeLoq Protocols

The typical applications of KeeLoq are the car anti-theft systems. Here the car ignition key authenticates itself to the car. For instance, the KeeLoq block cipher is

[3] Note that we believe that [12] reports an incorrect expected number of even cycles in this case.

used by such automotive OEMs as Chrysler, Daewoo, Fiat, GM, Honda, Toyota, Volvo, VW, Jaguar [2] and in the HomeLink wireless control systems to secure communication with garage door openers [3]. Other automotive KeeLoq applications include component authentication, vehicle-to-garage authentication, etc. KeeLoq is also used in various access control and property identification systems. Below three major types of security protocols are outlined in which the KeeLoq block cipher is involved.

KeeLoq hopping codes. These are also known as rolling codes and provide authentication of an encoder to the decoder (the main system) by sending an encrypted counter value (unilateral communication) [5]. The encoder and decoder share a 64-bit symmetric key and a 16-bit synchronized counter value. To authenticate itself the encoder encrypts the next counter value and sends it to the decoder which decrypts the message and verifies whether the received counter value is within the open window of length 16. A resynchronization mechanism exists to repair communication in case the counter value received exceeds the bounds of the open window. See also [6], [18].

KeeLoq IFF. The IFF (Identify Friend or Foe) systems provide authentication of a transponder to the main systems (decoder) using a simple challenge-response protocol (bilateral communication), see [7]. The transponder and decoder share a 64-bit symmetric key K. The encoder sends its 28-bit identifier to the main system. To require authentication the decoder sends a 32-bit random challenge to the transponder that replies with the corresponding KeeLoq-encrypted challenge using K. The decoder encrypts the genuine challenge using K corresponding to the identifier and compares the message received as a reply with this value. If they coincide, the authentication is accepted. See also [19].

4.2 KeeLoq Key Management Schemes

The KeeLoq systems use a number of different key management mechanisms depending on the concrete model of encoder/decoder. In all these schemes, an individual encoder key is derived from a manufacturer's key MK and some encoder-specific information in some way during the learning phase. The individual (KeeLoq encryption) key K is stored in the EEPROM of the encoder. The manufacturer's key MK is stored in the ROM and is fixed for large series of encoders. This enables each manufacturer to produce encoders that cannot be cloned by competitors. For example, MK can remain the same for all immobilizers installed in a certain car model within one production year. This fact makes the manufacturer's key a valuable attack target. It turns out that one can deduce some information about the manufacturer key from an individual encoder key which can be found using the cryptanalytic attacks on the KeeLoq block cipher described above. The concrete amount of gained information depends on the specific key derivation function.

There are two classes of key derivation functions used in KeeLoq systems: normal key generation and secure key generation. In this paper, we concentrate

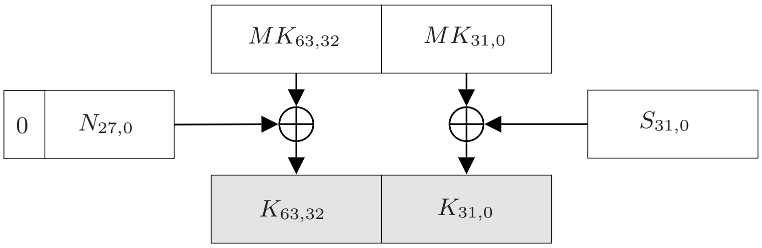


Fig. 4. XOR-based secure key generation with a 32-bit seed

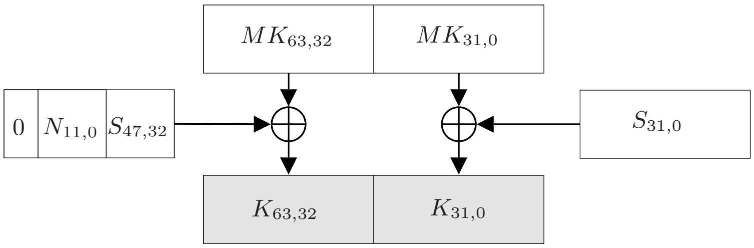


Fig. 5. XOR-based secure key generation with a 48-bit seed

on the XOR-based secure key generation and do not treat the normal key generation. The secure key derivation procedure has a variety of modes. Their general feature is that the individual encoder key depends on a randomly looking but fixed seed stored in the encoder and sent to the main system at the learning stage. The modes differ with respect to the length of the seed used.

To derive an individual key according to XOR-based secure key generation method, the encoder uses a seed S, its 28-bit encoder identifier $N_{27,0}$ and the 64-bit manufacturer's key $MK = MK_{63,0} = MK_{63,32}|MK_{31,0}$. Here $MK_{63,32}$ and $MK_{31,0}$ are the most significant and least significant 32-bit parts of MK, respectively.

There are three modes of the KeeLoq XOR-based secure key generation:

– *32-bit seed*: The 64-bit individual encoder key $K = K_{63,0} = K_{63,32}|K_{31,0}$ is obtained by XORing $MK_{31,0}$ with the seed $S = S_{31,0}$ and $MK_{63,32}$ with the zero-padded $N_{27,0}$ (Figure 4).
– *48-bit seed*: The seed $S_{47,0}$ is split into two parts - $S_{47,32}$ and $S_{31,0}$. $K_{31,0}$ is obtained by XORing $MK_{31,0}$ with $S = S_{31,0}$. $K_{63,32}$ is the XOR of the zero-padded $N_{11,0}|S_{47,32}$ with $MK_{31,0}$ (Figure 5).
– *60-bit seed*: In this case, the individual encoder key K does not depend on the encoder identifier SN. K is obtained by XORing MK with the zero-padded $S = S_{59,0}$ (Figure 6).

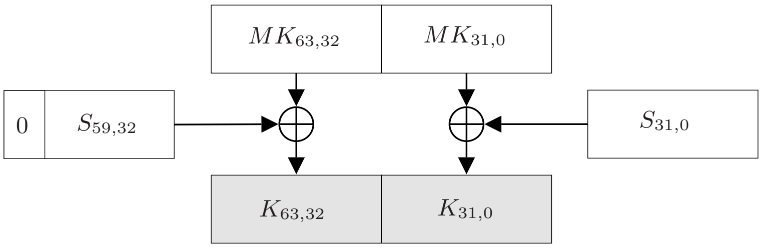


Fig. 6. XOR-based secure key generation with a 60-bit seed

4.3 Attacking KeeLoq Hopping Codes and IFF Systems

The protocols and key derivation schemes described above allow one to deduce some information about the manufacturer's key MK from a single individual encoder key.

Attack on KeeLoq IFF. In the case of KeeLoq IFF, the attacker can freely collect plaintext-ciphertext pairs choosing challenges and recording the responses. The PIC16 controllers run at a frequency of 4 MHz. As the KeeLoq cipher is implemented in hardware, every of its 528 clocks takes about 2^{-18} s. That is, it can be theoretically possible to encrypt all 2^{32} plaintexts within about 100 days on a PIC16 controller. Other controllers using KeeLoq can be clocked faster, thus reducing the time needed to collect the plaintext-ciphertext pairs. For instance, if a KeeLoq implementation is clocked with a moderate frequency of 64 MHz, the attacker can theoretically collect all needed pairs within about 6 days. Note that the encoder identifier is known as it is transmitted in plaintext before each protocol run.

Thus, one can apply the advanced linear slide attack with complexity 2^{37} working for the whole key space (Section 3.2) to the collected 2^{32} pairs and recover the individual encoder key K. As the identifier is known, the attacker obtains 32, 16, or 4 bits of MK by XORing the found K with $N_{27,0}$ or simply taking the 4 most significant bits, depending on the used key derivation function. This reduces the security of all other KeeLoq systems using the same manufacturer's key MK to 32, 48, or 60 bits, respectively.

Moreover, the attacker can have access to the seed used. For instance, he can intercept the transmission of the seed from the encoder to the decoder during the learn procedure, as the seed is sent in plaintext. Note that the attacker can also put an encoder into the learn mode by activating some of the PIC16 pins. In this case, the attacker pulls off the seed from the individual encoder key by XORing these two numbers and obtains the full manufacturer's key MK. This all other KeeLoq systems using the same manufacturer's key totally insecure, if the corresponding seeds are known.

Attack on KeeLoq hopping codes. There are at most 2^{16} input-output texts available, the synchronization counter running through the set $\{0, \ldots, 2^{16} - 1\}$.

The ciphertexts can be obtained directly from the communication channel. The corresponding plaintexts can be deduced, if the discrimination value is known. Note that it takes the encoder at most one hour to generate 2^{16} ciphertexts (even including the wireless data transmission, each session of which requires at most 50 ms). Moreover, the encoder identifier is known to the attacker, since it is transmitted in plaintext with every hopping code.

Having collected all needed plaintext-ciphertext pairs, the attacker can launch the algebraic attack mentioned above in Section 3.2 that is very well parallelizable and finds K in 2^{53} KeeLoq operations.

The rest of the attack works as described for KeeLoq IFF above, resulting in the reduction of the security to 32, 48, 60, or 0 bits, depending on the used key generation method and assumed security model.

5 Conclusion

In this paper we proposed practical key-recovery attacks on the KeeLoq block cipher used in numerous automotive applications as well as in various property identification systems.

Our cryptanalysis uses techniques of guess-and-determine, sliding, linear and cycle structure attacks. Our basic attack works with complexity of $2^{50.6}$ KeeLoq encryptions (while KeeLoq uses a 64-bit key). It requires all 2^{32} plaintexts and a memory of 2^{32} 32-bit words. Our second attack uses the cycle structure techniques to determine the first 16 bits of the key and then applies our first attack. It has a complexity of 2^{37} encryptions and requires 2^{32} known plaintexts, while being applicable to the whole key space. This is the best known attack on the KeeLoq block cipher working for the whole key space.

We demonstrated that several real-world applications are vulnerable to attacks on the KeeLoq block cipher, including KeeLoq IFF systems and KeeLoq hopping codes. After attacking one instance of KeeLoq using attacks presented in this paper, one can reduce the effective key length of all other KeeLoq systems of the same series to 32, 48, or 60 bits depending on the key management scheme applied. In some cases, the attacker is even able to obtain the encryption key instantly. All this imposes a real threat on the KeeLoq systems.

Acknowledgements. The author would like to thank Christof Paar for fruitful discussions and the anonymous referees for constructive observations.

References

1. Bogdanov, A.: Attacks in the KeeLoq Block Cipher and Authentication Systems. In: The 3rd Conference on RFID Security (RFIDSec 2007), Malaga, Spain (2007)
2. Wikipedia: Keeloq algorithm (2006), `http://en.wikipedia.org/wiki/KeeLoq`
3. HomeLink: Homelink and KeeLoq-based Rolling Code Garage Door Openers (2006), `http://www.homelink.com/home/keeloq.tml`

4. Microchip: Hopping Code Decoder using a PIC16C56, AN642 (1998), `http://en.wikipedia.org/wiki/KeeLoq` and `http://www.keeloq.boom.ru/decryption.pdf`
5. Microchip: An Introduction to KeeLoq Code Hopping (1996), `http://ww1.microchip.com/downloads/en/AppNotes/91002a.pdf`
6. Microchip: HCS101 Fixed Code Encoder Data Sheet (2001), `http://ww1.microchip.com/downloads/en/DeviceDoc/41115c.pdf`
7. Microchip: Using KeeLoq to Validate Subsystem Compatibility, AN827 (2002), `http://ww1.microchip.com/downloads/en/AppNotes/00827a.pdf`
8. Microchip: PIC12F635/PIC16F636/PIC16F639 Cryptographic Module General Overview, TB086 (2005), `http://ww1.microchip.com/downloads/en/DeviceDoc/91086A.pdf`
9. Biryukov, A., Wagner, D.: Slide Attacks. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, Springer, Heidelberg (1999)
10. Biryukov, A., Wagner, D.: Advanced Slide Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, Springer, Heidelberg (2000)
11. Biham, E., Dunkelman, O., Keller, N.: Improved Slide Attacks. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, Springer, Heidelberg (2007)
12. Courtois, N., Bard, G.: Algebraic and Slide attacks on KeeLoq (2007), `http://eprint.iacr.org/2007/062`
13. Siegenthaler, T.: Correlation-immunity of Nonlinear Combining Functions for Cryptographic Applications. IEEE Trans. on Inform. Theory IT-30 (1984)
14. Siegenthaler, T.: Decrypting a Class of Stream Ciphers Using Ciphertext Only. IEEE Trans. on Computers 34 (1985)
15. Chor, B., Goldreich, O., Hastad, J., Fridman, J., Rudich, S., Smolensky, R.: The Bit Extraction Problem or t-Resilient Functions. In: 26th Symposium on Foundations of Computer Science (1985)
16. Baigneres, T., Junod, P., Vaudenay, S.: How Far Can We Go Beyond Linear Cryptanalysis? In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, Springer, Heidelberg (2004)
17. Menezes, A., van Oorshot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
18. Microchip: HCS301 KeeLoq Code Hopping Encoder Data Sheet (2001), `http://ww1.microchip.com/downloads/en/devicedoc/21143b.pdf`
19. Microchip: HCS410 Keeloq Code Hopping Encoder and Transponder (2001), `http://ww1.microchip.com/downloads/en/DeviceDoc/40158e.pdf`

A Key Predistribution Scheme Based on 3-Designs*

Junwu Dong[1,2], Dingyi Pei[2], and Xueli Wang[3]

[1] College of Mathematics and Econometrics, Hunan University, Changsha 410082
[2] Institute of Information Security, Guangzhou University, Guangzhou 510006
[3] School of Mathematical Sciences, South China Normal University, Guangzhou 510631

Abstract. In wireless distributed sensor networks, it is important for sensor nodes to communicate securely each other. In order to ensure this security, many approaches have been proposed recently. One of them is to use key predistribution scheme (KPS). In this paper, we shall use the Möbius plane to present a key predistribution scheme for distributed sensor networks. The secure connectivity and resilience of the resulting sensor network will be analyzed in this paper. This KPS constructed in our paper has some better properties than the ones of KPSs constructed in [5],[7] and [9].

Keywords: sensor networks, key predistribution schemes, combinatorial designs, 3-designs, Möbius planes.

1 Introduction

Due to their wide applications, Distributed Sensor Networks (DSNs) have become an active research area recently. In applications, we often accept the following assumptions: (1) many sensor nodes are dropped, in a random way, to the target area. So that the network topology is unknown before the deployment. (2) the sensor nodes are typically low-cost, battery powered, and highly resource constrained, hence they should consume as little power as possible. (3) the sensor nodes have limited computation, storage, and communication capabilities, they can communicate with nodes only within a limited radius. We assume that the radio coverage area of each sensor node forms a circle of fixed radius whose center is that node. We call this circle a neighborhood of the given sensor node. Once the sensor nodes are deployed, they scan their neighborhoods and find out their neighbors.

In wireless distrubuted sensor networks, it is important for sensor nodes to communicate securely each other. Of course, public key infrastructure (PKI) can be used to establish pairwise secret keys between sensor nodes. However, the operations, which are based on the complex arithmatic of big integers, have

* This work was partially supported by the National Natural Science Foundation (China) under Grant 60473017 and 90604034.

to be implemented in the low-level environments. By now, it is not suitable to use PKI due to its expensive computational cost as well as storage consuming in each sensor node. Therefore it is natural to use the key predistribution scheme (KPS), where a set of secret keys is installed in each node before the sensor nodes are deployed. If two adjacent sensor nodes have at least one common keys, they can select one as the secret key and communicate securely by means of symmetric cryptography.

In general, a key predistribution scheme consists of three phases: key predistribution, shared key discovery, and path key establishment. First, a large pool of keys are specified, and each key is assigned a unique identifier. Then, every sensor node is loaded with a fixed number of keys chosen from the key pool, along with their key identifiers. After the deployment of the DSN, the shared key discovery phase takes place, where any two nodes in wireless communication range exchange their list of key identifiers to each other, and look for their common keys. If they share one or more common keys, they can pick one of them as their secret key for cryptographic communication. The path key establishment phase takes place if there is no common key between a pair of nodes which need to have cryptographic communication. We call a successive sequence of nodes a path, where any two adjacent nodes (also in the radio coverage range) have at least one common keys. If the sensor node i wants to communicate securely with the sensor node j, it needs to find a path between itself and the sensor node j. Thus messages from the sensor node i can reach the sensor node j securely.

In [1], Eschenauer and Gliger proposed a probabilistic key predistribution scheme. The main idea is to assign every sensor node randomly a set of keys from the given pool of keys before deployment, so any two sensor nodes have a certain probability of sharing at least one common keys. Extensions and variations of this approach can be found in [2,3,4].

To construct deterministic key predistribution scheme for DSN, using combinatorial design is another strategy in this area. This idea was first proposed in Çamtepe and Yener [5]. Further study in this context can be found in [6,7,8].

A combinatorial design is a pair of sets $(X, \mathscr{B})$, where $X = \{x_1, x_2, \cdots, x_v\}$ is a finite set, the elements of which are called points, and $\mathscr{B} = \{B_1, B_2, \cdots, B_b\}$ is a finite set of subsets of X, called blocks.

Example 1: Let

$$\begin{aligned} X &= \{0,1,2,3,4,5,6,7,8,9\}, \\ \mathscr{B} &= \{0129, 0134, 0158, 0167, 0235, 0247, 0268, 0369, 0378, 0456, \\ &\qquad 0489, 0579, 1236, 1245, 1278, 1357, 1389, 1468, 1479, 1569, \\ &\qquad 2348, 2379, 2469, 2567, 2589, 3459, 3467, 3568, 4578, 6789\}. \end{aligned}$$

In this design, X has ten points, $\mathscr{B}$ has thirty blocks, each block has four points of X. Furthermore, each point of X appears in twelve blocks, any pair of distinct points from X appears in four blocks, and any triple of distinct points from X appears in exactly one block. Such a combinatorial design will be called 3-$(10, 4, 1)$ design later.

Any combinatorial design can be used to establish a key predistribution scheme for a DSN. Let $X = \{x_1, x_2, \cdots, x_v\}$ and $\mathscr{B} = \{B_1, B_2, \cdots, B_b\}$ where each block B_j has k points of X. Let the sensor nodes be denoted by N_1, N_2, $\cdots$, N_b. For every $1 \leqslant i \leqslant v$, a key K_i is chosen randomly from some special key space. Hence there exists a 1-1 correspondence between X and $\{K_i \,|\, 1 \leqslant i \leqslant v\}$, and the symbol x_i can be called the label of K_i. Then for $1 \leqslant j \leqslant b$, the sensor node N_j receives the set of keys $\{K_i \,|\, x_i \in B_j\}$, that is, the block B_j is used to specify which keys are given to the node N_j. Thus each node receives k keys.

The effectiveness of a sensor network can be explained by the following two aspects: (1) the connective probability p_1, it is defined by the probability that any pair of sensor nodes shares a link, i.e., they have at least one common keys. Of course, this probability is expected as large as possible, since it measures the effectiveness of the sensor network. (2) the probability fail(1). If a sensor node is detected as being compromised, then all the keys it possesses should no longer be used by any node in the sensor network. Suppose the sensor nodes N_i and N_j have at least one common keys (which means that there is a link between the pair of N_i and N_j). If all the common keys of the pair of N_i and N_j are contained in the compromised sensor node, then N_i and N_j are no longer communicate directly, i.e., the link between N_i and N_j is lost. And the probability of links being affected is defined as

$$\text{fail}(1) = \frac{\text{the losted connectivities}}{\text{the original connectivities}}.$$

Generally, we expect fail(1) as small as possible, since it measures the resilience of the sensor network, when a random sensor node is compromised.

Çamtepe and Yener [5], constructed a key predistribution scheme for sensor network by use of finite geometry over projective planes. In [7], Lee and Stinson provided a key predistribution scheme by using a special combinatorial design $\text{TD}(k, N)$. In this scheme, any two sensor nodes share at most one common key.

In this paper, we shall construct a key predistribution scheme by using 3-designs. The most attractive feature of this key predistribution scheme is that when the number of sensor nodes b tends to ∞, we have $p_1 \to 1/2$ and fail$(1) \to 0$. In a previous paper[9] the authors found a key predistribution scheme, in which $p_1 \to 5/8$ and fail$(1) \to 0$ as $b \to \infty$. Although, the connective probabilities of this scheme is less than that of the scheme in [9], the probabilities fail(1) of this scheme is also much less than that of the scheme in [9].

The rest of this paper is arranged as follows. The key predistribution scheme based on 3-design will be presented in section 2, and the connective probabilities p_1 and fail(1) of the scheme will be computed. Some issues on implementation will be given in section 3. We compare the scheme of this paper with some known schemes in section 4.

2 New Scheme

Let $(X, \mathscr{B})$ be a combinatorial design with $|X| = v$ and every block B of $\mathscr{B}$ has exactly k elements of X. Let $t \geqslant 1$ be an integer, if any t distinct elements of X occur in λ blocks exactly, then it is called a t-(v, k, λ) design.

Now we are going to introduce a 3-design, which is called the Möbius plane or inverse plane in history. One may refer to [11] and [12] for more information on such topic.

Let $n \geqslant 2$ be an integer, and q a prime power. Let X be the set of all $q^n + 1$ points on the projective line $PG(1, \mathbb{F}_{q^n})$, which can be denoted by

$$\{(1, \alpha) \mid \alpha \in \mathbb{F}_{q^n}\} \cup \{(0, 1)\}.$$

The set X can be also denoted by $\mathbb{F}_{q^n} \cup \{\infty\}$. Let B denote $PG(1, \mathbb{F}_q)$, then B can be considered as a subset of X since $\mathbb{F}_q$ is a subfield of $\mathbb{F}_{q^n}$. Suppose $T \in GL_2(\mathbb{F}_{q^n})$, i.e., T is a non-singular 2×2 matrix over $\mathbb{F}_{q^n}$, then

$$\begin{aligned} PG(1, \mathbb{F}_{q^n}) &\longrightarrow PG(1, \mathbb{F}_{q^n}) \\ (x_0, x_1) &\longmapsto (x_0, x_1)T. \end{aligned}$$

defines a 1-1 transformation on X, which is called a projective transformation. It is easy to see that two matrix $T_1, T_2 \in GL_2(\mathbb{F}_{q^n})$ define the same transformation on X if and only if there exists an non-zero element α of $\mathbb{F}_{q^n}$ such that $T_1 = \alpha T_2$. Denote $G = PGL_2(\mathbb{F}_{q^n}) = GL_2(\mathbb{F}_{q^n})/\{\alpha I | \alpha \in \mathbb{F}_{q^n}\}$. Let $G_B = \{T \in G \mid T(B) = B\}$. It is easy to show that $G_B = PGL_2(\mathbb{F}_q)$.

Let $\mathscr{B} = \{T(B) \mid T \in G/G_B\}$, then we have

$$b = |\mathscr{B}| = |G|/|G_B| = \frac{(q^{2n}-1)(q^{2n}-q^n)}{q^n-1} \Big/ \frac{(q^2-1)(q^2-q)}{q-1} = \frac{q^n(q^{2n}-1)}{q(q^2-1)}.$$

The following proposition is well known in projective geometry.

Proposition 1. *Any three distinct points of $PG(1, \mathbb{F}_{p^n})$ can be mapped into any other three distinct points by a transformation in G.*

Proposition 2. *Let q be a prime power. Then there exists a 3-$(q^n+1, q+1, 1)$ design with the number of blocks $b = (q^n(q^{2n}-1))/(q(q^2-1))$.*

Proof. Let $X = PG(1, \mathbb{F}_{q^n})$, $G = PGL_2(\mathbb{F}_{q^n})$, and $\mathscr{B} = \{T(B) | T \in G\}$. Then $|X| = v = q^n + 1$, and each block of $\mathscr{B}$ is of $k = q + 1$ elements of X. By proposition 1, each 3-subset of X is at least contained in one block of $\mathscr{B}$. The total number of distinct 3-subsets of X is

$$\binom{q^n+1}{3} = \frac{(q^n+1)q^n(q^n-1)}{3!} = \frac{q^n(q^{2n}-1)}{3!}.$$

On the other hand, the number of taking three distinct elements from each block of $\mathscr{B}$ is

$$\binom{q+1}{3} = \frac{(q+1)q(q-1)}{3!} = \frac{q(q^2-1)}{3!},$$

Since the number of blocks in $\mathscr{B}$ is $b = (q^n(q^{2n}-1))/(q(q^2-1))$, we know that the total number of taking three distinct elements from all blocks of $\mathscr{B}$ is

$$b\binom{q+1}{3} = \frac{q^n(q^{2n}-1)}{q(q^2-1)} \cdot \frac{q(q^2-1)}{3!} = \frac{q^n(q^{2n}-1)}{3!} = \binom{q^n+1}{3},$$

This implies that each 3-subset of X is contained in exactly one block of $\mathscr{B}$. Therefore, the pair $(X, \mathscr{B})$ is a 3-$(q^n+1, q+1, 1)$ design.

Thus, we can construct a key predistribution scheme by the above 3-design:

Proposition 3. *Suppose q is a prime power, and $n \geqslant 2$ is an integer. Then there exists a key predistribution scheme for a DSN having $b = (q^n(q^{2n}-1))/(q(q^2-1))$ sensor nodes and every node containing exactly $q+1$ keys.*

Now we calculate the connective probability p_1 and fail(1).

First, we consider the case of $n = 2$. For any given i distinct elements, let λ_i $(1 \leqslant i \leqslant 3)$ denote the number of blocks in which these elements occur. Then we have the following:

Lemma 1. *In the 3-$(q^2+1, q+1, 1)$ design, the number of blocks is $b = q^3+q$, and $\lambda_1 = q^2+q$, $\lambda_2 = q+1$, and $\lambda_3 = 1$.*

Let C be a fixed block, and for every pair of elements $i, j \in C$, $i \neq j$, define

$$\mu'_C(i,j) = \#\{C' \in \mathscr{B} \,|\, C' \cap C = \{i,j\}\}.$$

which is the number of blocks which intersect with C at the set $\{i,j\}$ exactly. It is easy to see that the number $\mu'_C(i,j)$ is independent on the special block C and the special pair of elements i and j in C, hence denote it by $\mu'_C(2)$. Since there are $\lambda_2 = q+1$ blocks contain $\{i,j\}$, so that only $\lambda_2 - 1 = q$ blocks intersect with C at $\{i,j\}$ exactly, therefore $\mu'_C(2) = q$.

For every $i \in C$, let $\mu'_C(i)$ denote the number of blocks that intersect with C at $\{i\}$ exactly. It can be easily seen that $\mu'_C(i)$ is independent with the special block C and the special element i in C, and denote it by $\mu'_C(1)$. Similarly, as what we have done above, we know that

$$\mu'_C(1) = \lambda_1 - 1 - ((q+1)-1) * \mu'_C(2) = q-1.$$

Let C be a fixed block as above, and $\mu_C(1)$ and $\mu_C(2)$ denote the number of blocks that have only one or two common points with C, respectively. Then we have

$$\mu_C(1) = (q+1) * \mu'_C(1) = q^2 - 1$$
$$\mu_C(2) = \binom{q+1}{2}\mu'_C(2) = \frac{1}{2}q^3 + \frac{1}{2}q^2$$

Hence, the number of blocks that have some common keys with C is

$$\mu_C = \mu_C(1) + \mu_C(2) = \frac{1}{2}q^3 + \frac{3}{2}q^2 - 1.$$

So, the connective propobability of this KPS is

$$p_1 = \frac{\mu_C}{b-1} = \frac{(1/2)q^3 + (3/2)q^2 - 1}{q^3 + q - 1} \to \frac{1}{2}, \quad \text{when } q \to \infty$$

Next, we consider the resilience of the key predistribution scheme. All $q+1$ keys the compromised sensor node possesses should no longer be used by any node in the network. Furthermore, if all the common keys of sensor nodes N_i and N_j are contained in the compromised sensor node, then N_i and N_j can no longer communicate directly.

For any block C, there are μ_C blocks that have some common keys with it. Hence, the number of all the connections in this scheme is

$$\mu = \frac{b * \mu_C}{2} = \frac{1}{4}q^6 + \frac{3}{4}q^5 + \frac{1}{4}q^4 + \frac{1}{4}q^3 - \frac{1}{2}q.$$

Suppose that C_h is a block corresponding to the compromised sensor node N_h. For every pair i, $j \in C_h$, $i \neq j$, the following links are no longer used:

$$\{(C, C') \,|\, C \cap C' = \{i, j\}\}.$$

There are

$$\frac{\lambda_2 * \mu'_C(2)}{2} = \frac{q^2+q}{2}.$$

such links. Hence the number of pairs of C and C' such that $C \cap C' \subset C_h$ with $|C \cap C'| = 2$ is

$$\binom{q+1}{2} \frac{\lambda_2 * \mu'_C(2)}{2} = \frac{1}{4}q^4 + \frac{1}{2}q^3 + \frac{1}{4}q^2.$$

Similarly, the number of pairs of C and C' such that $C \cap C' \subset C_h$ with $|C \cap C'| = 1$ is

$$\binom{q+1}{1} \frac{\lambda_1 * \mu'_C(1)}{2} = \frac{1}{2}q^4 + \frac{1}{2}q^3 - \frac{1}{2}q^2 - \frac{1}{2}q.$$

So the probability that an arbitrary link is affected by the compromise of one node is

$$\text{fail}(1) = \frac{3q^2 + q - 2}{q^4 + 2q^3 - q^2 + 2q - 2} \to 0, \quad \text{when } q \to \infty.$$

The connective probability p_1 and fail(1) for some q in the case $n = 2$ are given in Table 3.

Now, we consider the case that $n \geqslant 3$. In order to simplify discussions, we only consider the probability of connectivity p_1.

Lemma 2. *In the* 3-$(q^n+1, q+1, 1)$ *design, we have*

$$b = \frac{q^n(q^{2n}-1)}{q(q^2-1)}, \;\; \lambda_1 = \frac{q^n(q^n-1)}{q(q-1)}, \;\; \lambda_2 = \frac{q^n-1}{q-1}, \;\; \lambda_3 = 1.$$

Suppose the symbols $\mu'_C(1)$, $\mu'_C(2)$, $\mu_C(1)$, $\mu_C(2)$, and $\mu(C)$ were defined as above. We consider the degrees of them as polynomials of q.

By lemma 2, we have

$$\deg(b) = 3n-3, \quad \deg(\lambda_1) = 2n-2, \quad \deg(\lambda_1) = n-1.$$

Since

$$\mu'_C(2) = \lambda_2 - 1, \quad \mu'_C(1) = (\lambda_1 - 1) - \big((q+1)-1\big)\mu'_C(2),$$

and

$$2n-2 = \deg(\lambda_1 - 1) > \deg(q * \mu'_C(2)) = n, \quad \text{(since } n > 2)$$

we have

$$\deg(\mu'_C(2)) = \deg(\lambda_2) = n-1, \quad \deg(\mu'_C(1)) = \deg(\lambda_1) = 2n-2.$$

Furthermore, by

$$\mu_C(1) = (q+1)\mu'_C(1), \quad \mu_C(2) = \binom{q+1}{2}\mu'_C(2)$$

we know that

$$\deg(\mu_C(1)) = 2(n-1)+1 = 2n-1, \quad \deg(\mu_C(2)) = (n-1)+2 = n+1,$$

Noting that $n \geqslant 3$, we have

$$2n-1 = \deg(\mu_C(1)) > \deg(\mu_C(2)) = n+1.$$

Therefore,

$$\deg(\mu_C) = \deg(\mu_C(1) + \mu_C(2)) = \deg(\mu_C(1)) = 2n-1,$$

and hence, by comparing the degrees,

$$p_1 = \frac{\mu_C}{b-1} \to 0, \qquad \text{when } \ q \to \infty.$$

So, we have the following theorem:

Theorem 1. *We have*

$$\lim_{q\to\infty} p_1 = \frac{1}{2} \qquad \textit{when} \qquad n = 2,$$

and

$$\lim_{q\to\infty} p_1 = 0 \qquad \textit{when} \qquad n > 2.$$

In fact, we can show that p_1 monotonously decreases to $1/2$ for the case $n = 2$ and $q > 2$. Let

$$f(q) = \mu_C = \frac{1}{2}q^3 + \frac{3}{2}q^2 - 1,$$
$$g(q) = b - 1 = q^3 + q - 1.$$

It is sufficient to show that $f'g - fg' < 0$. We have

$$f' = \frac{3}{2}q^2 + 3q, \quad g' = 3q^2 + 1,$$

and

$$gf' - fg' = -\Big(\frac{3}{2}q^4 - q^3 - 3q^2 + 3q - 1\Big)$$

Let

$$\begin{aligned} h(q) &= (-2) * (gf' - fg') \\ &= 3q^4 - 2q^3 - 6q^2 + 6q - 2. \end{aligned}$$

Assume $q > 2$, then

$$\begin{aligned} h(q) &= q^2(3q^2 - 2q - 6) + 6q - 2 \\ &= q^2\big(q(3q - 2) - 6\big) + 6q - 2 \\ &> q^2\big(2 * (6 - 2) - 6\big) + 6 * 2 - 2 \\ &= 2q^2 + 10 > 0 \end{aligned}$$

This proves what we wanted.

3 Implementation

Suppose that $n = 2$ and q is a prime in this section. The size of q is determined by the number of keys $k = q + 1$ per node. Take an irreducible polynomial $f(x)$ of degree 2 over the field $\mathbb{F}_q$. (for example, take $f(x) = x^2 - u$ where u is a non-square element of $\mathbb{F}_q$.) Let α be a root of $f(x)$, then we have

$$\mathbb{F}_{q^2} = \{a\alpha + b \,|\, a, b \in \mathbb{F}_q\}.$$

Let X be the set of $q^2 + 1$ points of the projective line $PG(1, \mathbb{F}_{q^2})$:

$$X = \{(1, \beta) \,|\, \beta \in \mathbb{F}_{q^2}\} \cup \{(0, 1)\}.$$

and B be the set of $q + 1$ points of the projective line $PG(1, \mathbb{F}_q)$:

$$\text{B}=\{(1, \beta) \,|\, \beta \in \mathbb{F}_q\} \cup \{(0, 1)\}.$$

as defined in section 2. Each point of X is assigned with a key chosen randomly from a key pool, and each key has a unique identifier by an integer from $(0, 1, \cdots, q^2)$.

We have the set of blocks $\mathscr{B} = \{T(B) \,|\, T \in G/G_B\}$. Each block is assigned to a unique sensor node which will receive the set of keys assigned to the points contained in this block. For assigning keys for each node, we need to know all the blocks in $\mathscr{B}$. This is equivalent to have a representation system of the cosets of $G_B = PGL_2(\mathbb{F}_q)$ in $G = PGL_2(\mathbb{F}_{q^2})$. We may find the system by computer if q is not too large.

When q grows larger, the number of blocks $b = q^3 + q$ in $\mathscr{B}$ may become too large for application. In this case we can only use a part of the blocks chosen randomly from $\mathscr{B}$. The experiments (Tables 1, 2 where t is the number of blocks used) shows that the parameters p_1 and fail(1) have only a small disturbance when only a part of blocks is used.

Table 1. Experimental results of p_1

q	29	59	79	109	139	179
$t = 1000$	0.551381	0.524314	0.518308	0.511952	0.512332	0.509700
$t = 2000$	0.552091	0.526103	0.519160	0.513462	0.512361	0.507129
$t = b$	0.551050	0.525271	0.518903	0.513718	0.510765	0.508364

Table 2. Experimental results of fail(1)

q	29	59	79	109	139	179
$t = 1000$	0.003311	0.000831	0.000469	0.000245	0.000155	0.000093
$t = 2000$	0.003305	0.000830	0.000467	0.000243	0.000155	0.000091
$t = b$	0.003376	0.000838	0.000471	0.000249	0.000153	0.000093

Choosing randomly a matrix $T \in GL_2(\mathbb{F}_{q^2})$, we can find a block

$$T(B) = \{(1, \beta)T \mid \beta \in \mathbb{F}_q\} \cup \{(0, 1)T\}.$$

Hence we may find a set of blocks by choosing a set of matrices in $GL_2(\mathbb{F}_{q^2})$. Suppose that the matrices T_1 and T_2 have been chosen successively, it is necessary to check whether T_1 and T_2 generate the same blocks $T_1(B)$ and $T_2(B)$. If this is the case we may choose another T_2. Repeating this process until enough blocks are found.

After the deployment of the distributed sensor network, any two nodes in the wireless communication range exchange their list of key identifiers to each other, and look for their common keys. If they have common keys, they can pick one of them as their secret key for cryptographic communication.

4 Comparisons

In [5], Çamtepe and Yener constructed a key predistribution scheme for sensor network by use of finite geometry over projective planes. Let q be a prime power, X the projective plane $PG(2, \mathbb{F}_q)$, $\mathscr{B}$ the set of projective lines in $PG(2, \mathbb{F}_q)$, then it is easily seen that $(X, \mathscr{B})$ is a $2-(q^2+q+1, q+1, 1)$ design. Each pair of lines has exactly one common point, so the connective probability $p_1 = 1$. However, sine $b = |\mathscr{B}| = q^2 + q + 1$, the number of keys per node is $k = q + 1 \approx \sqrt{b}$. When the size of DSN is large, it may be impossible for its heavy storage requirement.

For example, suppose that we want to construct a key predistribution scheme, by the Çamtepe and Yener's method, for a DSN having 1000000 nodes. Then the smallest prime power q such that $q^2 + q + 1 \geqslant 1000000$ is $q = 1009$. The resulting KPS would assign 1010 keys to every node.

In our schemes, the smallest prime power q such that $b = q^3 + q \geqslant 1000000$ is that $q = 101$. If we take $q = 101$, then the resulting key predistribution scheme for a DSN can support 1000000 sensor nodes and each node stores $k = q+1 = 102$ keys, which is much less than that in Çamtepe and Yener's scheme.

In [7], Lee and Stinson constructed a key predistribution scheme by using of transversal design TD(k, N).

Let $k \geqslant 2$ and $N \geqslant 1$. A transversal design TD(k, N) is a triple $(X, \mathscr{B}, \mathscr{G})$ such that the following properties are satisfied: (1) X is a set of kN elements called points, (2) $\mathscr{G}$ is a partition of X into k subsets of size N called groups, (3) $\mathscr{B}$ is a set of k-subsets of X called blocks, (4) any group and any block contain exactly one common point, and (5) every pair of points from distinct groups is contained in exactly one block. For further introduction, or construction to transversal designs, one can refer to [10].

A class of transversal design TD(k, N), where N is a prime and $k < N$, was constructed in [7]. In the resulting scheme, every two sensor nodes share at most one common key, the number of nodes $b = N^2$, the connective probability $p'_1 = k/(N + 1)$ and Fail(1) $= (N - 2)/(N^2 - 2)$. We compare the scheme based on this class of TD(k, N) with the scheme of this paper with $n = 2$. For a given prime power q, let $k = q + 1$ and N be the largest prime such that $N^2 \leqslant q^3 + q$. The comparison between these two schemes is given in Table 3.

The Table 3 shows that when these two schemes have the same number k of keys per node and approximately the same number b of nodes, the scheme of this paper has greater probability p_1 and lower fail(1).

Table 3. The comparison 1

q	b	v	k	p_1	fail(1)	N	p'_1	Fail(1)
5	130	26	6	0.767442	0.090909	11	0.500000	0.075630
7	350	50	8	0.699140	0.049836	17	0.444444	0.067265
11	1342	122	12	0.630872	0.021625	31	0.375000	0.030240
13	2210	170	14	0.611589	0.015788	47	0.291667	0.020390
17	4930	290	18	0.586123	0.009475	67	0.264706	0.014486
29	24418	842	30	0.551050	0.003376	151	0.197368	0.006535
59	205438	3482	60	0.525271	0.000838	449	0.133333	0.002217
79	493118	6242	80	0.518903	0.000471	701	0.113960	0.001422
109	1295138	11882	110	0.513718	0.000249	1129	0.097345	0.000884
139	2685758	19322	140	0.510765	0.000153	1637	0.085470	0.000610
179	5735518	32042	180	0.508364	0.000093	2393	0.075188	0.000418

Table 4. The comparison 2

q	b	v	k	p_1	fail(1)	N	$b' = N^2$	p'_1	Fail(1)
5	130	26	6	0.767442	0.090909	7	49	0.750000	0.106383
7	350	50	8	0.699140	0.049836	11	121	0.666667	0.075630
11	1342	122	12	0.630872	0.021625	19	361	0.600000	0.047354
13	2210	170	14	0.611589	0.015788	23	529	0.583333	0.039848
17	4930	290	18	0.586123	0.009475	31	961	0.562500	0.030240
29	24418	842	30	0.551050	0.003376	59	3481	0.500000	0.014659
59	205438	3482	60	0.525271	0.000838	127	16129	0.468750	0.007751
79	493118	6242	80	0.518903	0.000471	157	24649	0.506329	0.006289
109	1295138	11882	110	0.513718	0.000249	223	49729	0.491071	0.004444
139	2685758	19322	140	0.510765	0.000153	277	76729	0.503597	0.003584
179	5735518	32042	180	0.508364	0.000093	359	128881	0.500000	0.002723

Table 5. The comparisin 3

q	b_2	p_2	FAIL(1)	b	p_1	fail(1)
5	3100	0.7677	0.1250	130	0.767442	0.090909
7	16758	0.7331	0.0877	350	0.699140	0.049836
11	58968	0.7118	0.0671	1342	0.630872	0.021625
13	160930	0.6975	0.0542	2210	0.611589	0.015788
17	371124	0.6873	0.0453	4930	0.586123	0.009475
29	20510308	0.6541	0.0194	24418	0.551050	0.003376
59	714920818	0.6396	0.0093	205438	0.525271	0.000838
79	3077050158	0.6359	0.0069	493118	0.518903	0.000471
109	15386227668	0.6330	0.0050	1295138	0.513718	0.000249
139	51888825378	0.6312	0.0039	2685758	0.510765	0.000153
179	183765964858	0.6299	0.0030	5735518	0.508364	0.000093

In both schemes, we take the same k, and approximately the same connective probability p_1 by choosing suitable N in TD(k, N), then we compare the number b and fail(1). Some of the results are listed in the following Table 4, where b', p'_1, and Fail(1) denote the corresponding parameters of the scheme in [7].

In [9], Pei et. al. constructed a key predistribution scheme based on rational normal curves over finite fields. The Table 5 gives a comparison between the scheme in [9] and the scheme of this paper when both schemes have the same number of keys $k = q + 1$ per note, where b_2, p_2 and FAIL(1) denote the corresponding parameters of the scheme in [9].

The Table 5 shows that when the number of keys per node is equal, the block number in scheme of [9] is larger than that of this scheme, the connective probability of [9] is larger than that of this scheme, and the fail(1) of the scheme of this paper is much less than that of [9].

5 Conclusions

In this paper, we construct a key predistribution scheme by using a the 3-design. Comparing with the scheme from [7], our scheme has higher connective probability and lower fail(1). And comparing with scheme from [9], our scheme has lower fail(1).

References

1. Eschenauer, L., Gligor, V.B.: A key–management scheme for distributed sensor networks. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 41–47. ACMCCS (2002)
2. Chan, H., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Research in Security and Privacy (2003)
3. Du, W., Deng, J., Han, Y.S., Varsheney, P.K.: A pairwise key predistribution scheme for wireless sensors. In: Proceeding of the 10th ACM Conference on Computer and Communications Security (CCS), October 2003, pp. 42–51 (2003)
4. Liu, D., Ning, P.: Establishing pairwise keys in distributed sensor networks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, ACMCCS (2003)
5. Çamtepe, S.A., Yener, B.: Combinatorial design of key distribution mechanisms for wireless sensor networks. Technical Report TR–04–10, RPI Dept. of Computer Science (April 2004)
6. Lee, J., Stinson, D.R.: Deterministic key predistribution schemes for sonsor networks. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 294–307. Springer, Heidelberg (2004)
7. Lee, J., Stinson, D.R.: A combinatorial approach to key predistribution for distributed sensor networks. In: IEEE Wireless Computing and Networking Conference (WCNC 2005), New Orleans, LA, USA (2005)
8. Wei, R., Wu, J.: Product construction of key distribution schemes for network. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, Springer, Heidelberg (2004)
9. Pei, D., Dong, J., Rong, C.: A novel key pre-distribution scheme for wireless distributed sensor networks (preprint)
10. Street, A.P., Street, D.J.: Combinatorics of experimental design. Clarendon Press, Oxford (1987)
11. Pei, D.: Authentication codes and combinatorial designs. Chapman & Hall / CRC (2006)
12. Hua, L.K., Wan, Z.X.: Classical groups. Shanghai Science and Technology Press (1963)

Provably Secure N-Party Authenticated Key Exchange in the Multicast DPWA Setting[⋆]

Weijia Wang[1,2], Lei Hu[2], and Yong Li[3,2]

[1] School of Science,
Beijing Jiaotong University, Beijing 100044, P.R. China
[2] State Key Laboratory of Information Security
(Graduate University of Chinese Academy of Sciences)
Beijing 100049, P.R. China
[3] School of Electronics and Information Engineering,
Beijing Jiaotong University, Beijing 100044, P.R. China
{wwj,hu,yli}@is.ac.cn

Abstract. Until now, it is still an open problem to provide a provably secure and efficient protocol for treating the case in which n communication parties can authenticate each other and establish a secure session key with their respective passwords shared with a trusted server. Accordingly, in this paper we propose a solution in a formal way. Firstly, we review the strengthened EKE-M protocol—a maiden attempt to resolve the setting above and point out a subtle flaw in it that may cause unknown key sharing attacks. Next, based on previous work in the adversary model for key establishment protocols, we provide an extended one for the N-party setting. Finally, we propose a constant-round and provably secure generic construction of N-party different password-authentication (DPWA) key exchange protocols in the multicast setting.

Keywords: Password, authenticated key exchange, key distribution, multi-party protocols.

1 Introduction

In the last few years, password-based authenticated key exchange (PAKE) protocols have received much attention due to their simplicity and convenience, in which communication parties can establish authenticated session key for later secure communications only by using short secrets—passwords, which occupy little memory space and facilitate human being's remembering. Such resolutions are particularly attractive for the practical environments in which communication clients are light-weight or hand-held devices that can not afford a heavyweight infrastructure such as public key infrastructure (PKI).

There are two classes of PAKE protocols. The first is the *shared password-authentication* (SPWA) scheme [14] which uses a password shared among communication parties to implement authentication and session key establishment.

[⋆] This work was supported by NSFC60573053.

The second is the *different password-authentication* (DPWA) scheme [14] in which parties authenticate each other and build a common session key by the help of a trusted server, with which each of them share a distinct password.

Generally, SPWA-type schemes are used for the setting in which there are only a few participators. It is a very difficult task for a large number of parties to keep sharing a single password synchronously and securely, since the compromise or corruption of any one of these parties will lead to so serious a trouble on the whole group that the remaining parties must re-communicate each other and update the shared password, which can hardly be implemented in a real-world network circumstance, especially for a large scale group. Also, previous work in the SPWA-type in the literature mainly focus on the 2-party setting, where one party C, a client and the other S, a server implement authenticated key exchange by using their shared password.

In contrast, DPWA-type schemes are well suited for the multi-party case where the parties of a group only need to hold and manage their respective passwords shared with a trusted server. Even if finding someone being corrupted, the remaining honest parties can easily rebuild a new session to exclude him by using their respective passwords. Certainly, the main drawback of this type scheme is that the trusted server is always needed throughout the establishment of all communication.

Related work. In the initial stages of the research on PAKE, most researchers consider different aspects of SPWA-type schemes in the 2-party case [6, 7, 5, 9, 11, 18, 19, 20, 24, 25, 27]. Recently, based upon the previous work on group key exchange (GKE) [10, 12, 22], several N-party SPWA-type schemes [17, 23, 1, 4, 8] were proposed, and some of them [1, 4, 8] are equipped with formal treatments for the security.

Simultaneously, the importance of the multi-party DPWA-type PAKE also began to be realized by the security community, especially followed by an increasing recognition that strict formal treatments for those of the DPWA-type were needed. Abdalla et al. [2] recently provided a formal security model for DPWA-type PAKE protocols in the 3-party setting, based on those of Bellare and Rogaway [7] for key distribution schemes and that of Bellare, Pointcheval, and Rogaway [5], and presented a provably secure generic construction for this class of protocols under their model. Subsequently, Abdalla et al. [3] proposed an simple and efficient 3-party PAKE protocol and verified its security under their security model. Most recently, Wang and Hu [29] provided a modified security model for the treatment of resisting undetectable on-line dictionary attacks, and also proposed a new provably secure and efficient construction for 3-party PAKE protocols.

On the other hand, Byun and Lee [14] firstly extended DPWA-type schemes to the N-party case, and proposed two password-based Diffie-Hellman key exchange schemes, referred to as the EKE-U and EKE-M protocols, for two distinct network environments: *unicast* network and *multicast* network. The former assumes that one client can send messages one by one to the next client in one direction; The latter assumes that any client can send messages to multiple recipients only in one round (in parallel) during the protocol. However, although both of these two schemes are claimed to be provably secure, Tang and Chen [28] soon showed that the EKE-U protocol suffers from off-line dictionary attacks and the EKE-M protocol is susceptible to undetectable on-line dictionary attacks. Subsequently, Byun and Lee [15] provided immediate countermeasures (two strengthened versions) on Tang et al.'s attacks in which they strengthen their two schemes by adding key confirmation procedures after each clients and the server build the session key. And then they adapt the strengthened EKE-M to the multi-layer MANET environment in [16]. Nevertheless, very recently, Phan and Goi [26] made an extensive analysis of the EKE-U protocol and its strengthened version, based on the prior work of Tang and Chen [28]. They pointed out that the strengthened EKE-U is still susceptible to a variant of the unknown key-share attack even if temporary session keys are used to encrypt the channel between clients and the trusted server in the up-flow stage instead of passwords being used in the original version. They also review that the strengthened EKE-M cannot provide *key privacy*[1] due to its key distribution structure.

Our contribution. In this paper, we consider the N-party DPWA-type PAKE in the multicast network case, which allows a group of members to build a group session key concurrently in a multicast environment with respective distinct passwords by the help of a trusted server. Firstly, we further analyze the strengthened EKE-M protocol [15], which is the countermeasure version of the EKE-M [14] on Tang et al's attacks [28], and present a trivial but interesting attack against it. Secondly, based on the prior work on the security model for 3-party PAKE case [2, 29] and those on the security for group key exchange protocols [12, 20], we present an extended one for N-party DPWA-type PAKE in the multicast network case. Finally, we propose a constant-round and provably secure generic construction of N-party DPWA-type PAKE protocols in the multicast setting, in which the secure 2-party PAKE schemes between each client and the trusted server and a Burmester-Desmedt (BD) group key exchange protocol [13] among clients, as two independent and parallel components, are perfectly connected by the secure Message authentication code (MAC) schemes.

Outline of the paper. Our paper is organized as follows. In section 2, we briefly review the strengthened EKE-M protocol [15] and discuss the attacks against it. In section 3, we present a security model for N-party DPWA-type PAKE protocols in the multicast setting based on some prior work. In section 4, several security primitives are introduced, which will be used in the following work. In section 5, we propose a generic construction for such protocol and provide its security results. Finally, conclusions and discussions are given in section 6.

[1] The trusted server can not obtain any information of the session key in a passive way.

2 Strengthened EKE-M Protocol

In the sequel, we briefly review the strengthened EKE-M scheme which is proposed as an improved version of the original protocol to resist the attacks presented by Tang and Chen [28], and point out that it still has a subtle flaw which may cause a weak variant of *unknown key share attack*[2], under which the session key generated by each client can be easily interfered by an adversary. Note that another variant of unknown key share attack in N-party case was provided by Phan and Goi [26], where each client (except C_{n-1}) believes it is sharing a session key with all other clients including C_{n-1} which is rightly so, but C_{n-1} is not present and does not know that such a key has been established.

2.1 Description of the Strengthened EKE-M Protocol

Let $G = \langle g \rangle$ be a cyclic group of prime order q, $sid' = \varepsilon_{pw_1}(g^{x_1})||\varepsilon_{pw_2}(g^{x_2})||...$ $||\varepsilon_{pw_{n-1}}(g^{x_{n-1}})$ and $SIDS = sid'||sk_1 \oplus N||sk_2 \oplus N||...||sk_{n-1} \oplus N$.

	S	C_1	C_2	...	C_{n-1}
Round 1	$s_i \leftarrow [1, q-1]$ $\varepsilon_{pw_i}(g^{s_i})$	$x_1 \leftarrow [1, q-1]$ $\varepsilon_{pw_1}(g^{x_1})$	$x_2 \leftarrow [1, q-1]$ $\varepsilon_{pw_2}(g^{x_2})$	... ...	$x_{n-1} \leftarrow [1, q-1]$ $\varepsilon_{pw_{n-1}}(g^{x_{n-1}})$
Round 2	$H_2(sk_i\|\|S)$	$H_2(sk_1\|\|C_1)$	$H_2(sk_2\|\|C_2)$	...	$H_2(sk_{n-1}\|\|C_{n-1})$
Round 3	$N \leftarrow [1, q-1]$ $sk_1 \oplus N\|\|...\|\|sk_{n-1} \oplus N$				

Fig. 1. The strengthened EKE-M protocol

- In Round 1, the server S sends $\varepsilon_{pw_i}(g^{s_i})$ to the clients C_1,C_2,...,C_{n-1} concurrently. Simultaneously each client C_i, $1 \le i \le n-1$, sends $\varepsilon_{pw_i}(g^{x_i})$ to S. Upon receiving their respective message above, S and C_i,$1 \le i \le n-1$ share an ephemeral Diffie-Hellman Key, $sk_i = H_1(sid'||g^{x_i s_i})$.
- In Round 2, S and C_i, $1 \le i \le n-1$, authenticate each other by broadcasting authenticators $H_2(sk_i||S)$ and $H_2(sk_i||C_i)$, and checking their validity, respectively.
- In Round 3, S chooses a random value N from Z_q^* and then sends $N \oplus sk_i$ to C_i, $1 \le i \le n-1$, concurrently. After obtaining the message above, every client generates a common key , $sk = H_2(SIDS||N)$.

If it is necessary for some applications to provide the mutual authentication (key confirmation), the additional authenticator $H_4(sk||i)$ can be used as described in [15].

[2] An unknown key share attack on a 2-party case is where one party A believes it is sharing a session key with B which is true, but B instead believes it is sharing a session key with $E \neq A$.

2.2 Analysis of the Strengthened EKE-M

In fact, the original version of this protocol has not the property of mutual authentications between each client and the trusted server, so Tang and Chen [28] point out that an adversary, who may be an inside attacker, can perform undetectable on-line dictionary attacks, namely, it can guess the password of another client among the group, start the protocol with both the illegal and its own legal identities and finally verify its guess on-linely in each session by comparing the two session keys obtained by using his two identities, respectively, while the server can not detect it.

By adding mutual authentication between each client and the trusted server, Byun and Lee [15] resolved the problem on undetectable on-line dictionary attacks. Unfortunately, we find that the modified version is still not so secure as they claimed, as shown as follows.

The nonce N that the server distribute to each client in the last message can be replaced with any value by an adversary. Specifically, it is assumed that there are totally three clients C_1, C_2 and C_3 and the protocol among them has been executed in the key distribution stage so that three messages $N \oplus sk_i$, where $1 \leq i \leq 3$, have been delivered by the server. An adversary can interfere them by substituting the three messages $N \oplus sk_i$ $(1 \leq i \leq 3)$ with any three different values. As a result, if the optional key confirmation stage does not exist in some cases, the three clients will generate different session keys but they do not know.

However, even if the key confirmation stage is involved in the protocol, an adversary still can intercept the three message aforementioned and compute $sk_1 \oplus sk_3 = (N \oplus sk_1) \oplus (N \oplus sk_3)$ and $sk_2 \oplus sk_3 = (N \oplus sk_2) \oplus (N \oplus sk_3)$. Next, it sends to C_1, C_2 and C_3 three values $sk_1 \oplus sk_3$, $sk_2 \oplus sk_3$ and 0 instead of $N \oplus sk_1$, $N \oplus sk_2$ and $N \oplus sk_3$. Finally, the nonce received by each party would be the session key sk_3 between the server and C_3. Certainly, the attack above is trivial since the adversary does not know the modified nonce, but it is still a hidden danger for a secure protocol that the nonce distributed by the server can be changed easily by any attacker.

In fact, the above attack is based on the observation that in the strengthened EKE-M, authentication is absent in the key distribution stage. To resist this attack, the protocol can be enhanced by sending the nonce with its authentication information by using each session key between each client and the server. However, strictly speaking, the EKE-M scheme is essentially a multi-party key distribution protocol, which is absent of some important properties desirable in the real applications such as forward security [21] and privacy security [2] and is exceedingly dependent on the server. So how to build a secure N-party DPWA-type PAKE protocol is still an open problem. For this, in the following sections, we present a provably secure general construction of secure N-party DPWA-type PAKE protocols, which consists of three independent components: a semantically secure 2PAKE protocol, a secure MAC scheme, and the Burmester-Desmedt group key exchange protocol [13].

3 Security Model

Generally, a formal model for protocols is composed of two main parts: the description of the ability of the adversary and the security definitions. In this section we present a security model, which specifies the N-party DPWA-type PAKE setting, based on the prior work of [2, 12, 22, 29]. Different from the one of Byun and Lee [14], which is a simple extension of the work of [12], our model for N-party DPWA-type PAKE protocols considers the ability of the adversary attacking the server, and as in [2, 29], introduces a new oracle *SendSever* to simulate this ability. Further, in order to treat formally the security against undetectable on-line dictionary attacks, we add the authentication security notion in the security definitions. On the other hand, although our model is more likely a generalization of the models of Abdalla et al. [2] and Wang and Hu [29], only the common semantic security notion which is also referred to as the Find-Then-Guess (FTG) model [2], rather than the stronger model—Real-Or-Random (ROR) model [2], is employed in our model so that the scheme provable under it has stronger security. The difference between the FTG and the ROR models is that in the former the *Reveal* query is allowed and only single $Test$ query can be asked in all sessions, while in the latter the *Reveal* query is not available but $Test$ can be queried once for each fresh instance.

3.1 Protocol Syntax

Protocol Participants. In a N-party DPWA-type PAKE protocol P, there are two types of participants: clients and a trusted server. We denote by $\mathcal{U} = \{U_1,...,U_n\}$, where the number n of players is up bounded polynomially in the security parameter k, the former that can participate in the protocol P, and by S the latter that is supposed to be always online. Here the set $\mathcal{U}$ can be further divided into two disjoint subsets: $\mathcal{C}$, the set of honest clients and $\mathcal{E}$, the set of malicious clients, and the set of all client participators $\mathcal{U}$ is the union $\mathcal{C} \bigcup \mathcal{E}$. The malicious set $\mathcal{E}$ corresponds to the set of inside attackers, which is the particular case in the DPWA-type setting.

Each participants may engage in several distinct, possibly concurrent, executions of the protocol P so that they initiate the corresponding instances called oracles. We denote by U^i (S^j) the instance i (j) of a participant U (S).

Long-lived keys. Each client party U_i keeps a low-entropy password pw_i, which is assumed to be uniformly drawn from a small dictionary $\mathcal{D}$. The trusted server S holds in its database a vector $pw_S = \langle pw_S[U] \rangle_{U \in \mathcal{U}}$ of the password verifiers of all the parties, and we denote by pw_E, where $E \in \mathcal{E}$, the set of passwords held by the inside attackers.

Communication model. In the model, it is assumed that an adversary $\mathcal{A}$ potentially control all communications in the network. During the execution of the protocol, the interaction between an adversary and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real system. We consider these queries as follows:

1. $Execute(\{U_1^{i_1}, ..., U_n^{i_n}\}, S^j)$: This query models passive attacks, where $\mathcal{A}$ gets access to honest executions among the client instances $U_1^{i_1}, ..., U_n^{i_n}$ and the trusted server instance S^j by eavesdropping. The output of this query consists of the resulting transcript that was exchanged during the honest execution of the protocol P.
2. $SendClient(U^i, m)$: This query models an active attack against a client, in which $\mathcal{A}$ sends a message m to client instance U^i and gets back the response the client generates in processing the message according to the protocol P. This query can be utilized by $\mathcal{A}$ to perform various active attacks such as impersonation attacks and man-in-the-middle attacks through modifying and inserting the messages of the protocol. A query $Send(Start)$ initializes a new instance of the protocol P, and thus the adversary receives the initial flows sent out by the instance.
3. $SendServer(S^j, m)$: This query models an active attack against the trusted server, in which the adversary sends a message m to server instance S^j. It outputs the message which server instance S^j generates upon receipt of the message.
4. $Reveal(U^i)$: This query models known key attacks (or Denning-sacco attacks), in which $\mathcal{A}$ gets the group session keys of the protocol instance involving the client instance U^i. Only if the group session key between the client instance U^i and partnered instances is defined, the query is available and returns it to the adversary.
5. $Test(U^i)$: This query is used to measure the semantic security of the group session key of client instance U^i, which models the misuse of the session key by a client instance in a way. If the session key is not defined, it returns $\perp$. Otherwise, it returns either the session key held by client instance U^i if $b = 0$ or a random number of the same size if $b = 1$, where b is a random bit preselected.

Notation. An instance U^s is said to be *opened* if the query $Reveal(U^s)$ has been made by the adversary. We say an instance U^s is *unopened* if it is not *opened*. An instance U^s is said to be *accepted* if it goes into an accept state after receiving the last expected protocol message, which also means it has enough information to generate the session key.

Session IDS and Partnering. Our approach defining the two notions is from [5], where the session IDS (SIDS) for the client instance U_i^s is defined as $\mathrm{SIDS}(U_i^s) = \{\mathrm{SID}_{ij} : j \in \mathrm{ID}\}$, where SID_{ij} is the concatenation of all transcript that the instance U_i^s exchanges with the instance U_j^t during an execution of P. It should be noted that SIDS is public and independent of the group session key.

Following the definition of SIDS, we say that two instances U_i^s and U_j^t are *directly partnering* if both of them are accepted and $\mathrm{SIDS}(U_i^s) \cap \mathrm{SIDS}(U_j^t) \neq \emptyset$ holds, denoted as $U_i^s \leftrightarrow U_j^t$. Based on this, we can further define that two instances U_i^s and U_j^t are *partnering* if there exists a route with several instances as nodes and the two ones above as the two ends, in which each pair of adjoining nodes are directly partnering (For more details, one can refer to [12]). We denote this partnering as $\mathrm{SIDS}(U_i^s) \leftrightsquigarrow \mathrm{SIDS}(U_j^t)$.

Freshness. If an instance U^i has been accepted, both the instance and its partners are unopened and they are both instances of honest clients, we say the instance U^i is *fresh*.

AKE semantic security. The security notion is defined in the context of executing a N-party DPWA PAKE protocol P under the control of an adversary $\mathcal{A}$. During execution of the protocol, an adversary $\mathcal{A}$ is allowed to send multiple queries to the $Execute$, $SendClient$ and $SendServer$, but ask at most one $Test$ oracle to a fresh instance. Finally $\mathcal{A}$ outputs its guess b' for the bit b hidden in the $Test$ oracle. An adversary $\mathcal{A}$ is said to be successful if $b' = b$. We denote this event by $Succ$. Given that passwords are drawn from dictionary $\mathcal{D}$, we define the advantage of $\mathcal{A}$ in violating the AKE semantic security of the protocol P and the advantage function of the protocol P, respectively, as follows:

$$Adv_{P,\mathcal{D}}^{ake}(\mathcal{A}) = 2 \cdot Pr[Succ] - 1$$
$$Adv_{P,\mathcal{D}}^{ake}(t, R) = \max_{\mathcal{A}}\{Adv_{P,\mathcal{D}}^{ake}(\mathcal{A})\},$$

where $maximum$ is over all adversaries $\mathcal{A}$ with time-complexity at most t and using resources at most R (such as the number of oracle queries).

We say a N-party PAKE protocol P is semantically secure if the advantage $Adv_{P,\mathcal{D}}^{ake}(t, R)$ is only negligibly larger than $nq/|\mathcal{D}|$, where q is the number of active sessions and n is the number of the honest participants.

Authentication security. As in [29], to measure the security of a N-party DPWA-type protocol resisting the undetectable on-line dictionary attacks, we consider the unilateral authentication from the client to the trusted server. We denote by $Succ_P^{auth(C\to S)}(\mathcal{A})$ the probability that an adversary $\mathcal{A}$ successfully impersonates a client instance during executing the protocol P while the trusted server does not detect it. Further, $Succ_P^{auth(C\to S)}(t, R) = \max_{\mathcal{A}}\{Succ_P^{auth(C\to S)}(\mathcal{A})\}$ is defined as the maximum over all adversaries $\mathcal{A}$ running in time at most t and using resources at most R. We say a N-party DPWA-type protocol P is *client-to-server authentication secure* if $Succ_P^{auth(C\to S)}(t, R)$ is negligible in the security parameter.

4 Security Assumptions

In this section, we briefly review the definitions [2,22] of the cryptographic primitives which are used as building blocks in our scheme.

Decisional Diffie-Hellman assumption (DDH): Let $\mathbb{G}$ be a cyclic group of prime order q and let g be an arbitrary generator of $\mathbb{G}$. We consider two experiments: $Exp_{\mathbb{G}}^{ddh-real}$ and $Exp_{\mathbb{G}}^{ddh-rand}$. In the former, g^x, g^y and g^{xy} are given, and in the latter g^x, g^y and g^z are provided, where x, y and z are drawn at random from $\{1, ..., q\}$. We define $Adv_{\mathbb{G}}^{ddh}(t)$ as the maximum value, over all probabilistic polynomial algorithms Δ running in time at most t, of:

$$|Pr[Exp_{\mathbb{G}}^{ddh-real}(\Delta) = 1] - Pr[Exp_{\mathbb{G}}^{ddh-rand}(\Delta) = 1]|.$$

We say the DDH assumption in $\mathbb{G}$ holds if $Adv_{\mathbb{G}}^{ddh}(t)$ is a negligible function of t.

Parallel Decisional Diffie-Hellman assumption (PDDH): [22]Let us consider an extension of the DDH assumption, where there are the two distributions as follows:

$$PDDH_n^* = (g^{x_1}, ..., g^{x_n}, g^{x_1x_2}, ..., g^{x_{n-1}x_n}, g^{x_nx_1} | x_1, ..., x_n \in_R \mathbb{Z}_q),$$
$$PDDH_n^{\#} = (g^{x_1}, ..., g^{x_n}, g^{y_1}, ..., g^{y_n} | x_1, ..., x_n, y_1, ..., y_n \in_R \mathbb{Z}_q).$$

Δ is assumed to be a probabilistic polynomial (t, ϵ)-distinguisher for these two cases with the advantage $Adv_{\mathbb{G}}^{pddh_n}(\Delta)$, so that the advantage function $Adv_{\mathbb{G}}^{pddh_n}(t)$ is defined as the maximum value over all Δ with at most time complexity t.

Lemma 1. *The $PDDH_n$ is equivalent to the DDH for any prime order group $\mathbb{G}$, any integer n and any time complexity T,*

$$Adv_{\mathbb{G}}^{ddh}(T) \leq Adv_{\mathbb{G}}^{pddh_n}(T) \leq nAdv_{\mathbb{G}}^{ddh}(T)$$

Proof. The proof of this lemma already exists and one can refer to [22, 1] for more details.

Message authentication codes (MAC). A message authentication code MAC = (Tag; Ver) is defined by the following two algorithms:

– A MAC generation algorithm Tag, possibly probabilistic, which produce a tag μ with the input of a message m and a secret key sk.
– A MAC verification algorithm Ver, which takes a tag μ, a message m, and a secret key sk as the input, and then outputs 1 if μ is a valid tag for m under sk or 0 otherwise.

A MAC scheme is existential unforgeability under chosen-message attacks (euf-cma) [2] if the adversary can not create a new valid message-tag pair, even after obtaining many valid message-tag pairs. Formally, let us consider the experiment, in which let l be a security parameter and sk be a secret key selected uniformly at random from $\{0, 1\}^l$, and let $\mathcal{A}$ be the adversary attacking the security of MAC, who is allowed to ask a MAC generation oracle Tag(sk; $\cdot$) and a MAC verification oracle Ver(sk; $\cdot, \cdot$) and outputs a message-tag pair $(m; \mu)$. Let $Succ$ denote the event in which $\mathcal{A}$ generates a legal message-tag pair that was not outputted by the Tag(sk; $\cdot$) oracle on input m. The advantage of $\mathcal{A}$ in violating euf-cma is defined as $Adv_{\mathcal{A}}^{euf-cma} = Pr[Succ]$. We define $Adv_{MAC}^{euf-cma}(t, q_g, q_v)$ as the maximal value of $Adv_{\mathcal{A}}^{euf-cma}$ over all $\mathcal{A}$ running in time at most t and asking at most q_g and q_v queries to its MAC generation and verification oracles, respectively.

5 Our Scheme

In this section, we present a generic construction (referred as to DGPAKE) of the DPWA-type PAKE protocols in N-party settings. As mentioned earlier, our scheme is essentially a compiler that mainly includes two independent components: a semantically secure 2-party password-based authenticated key exchange scheme and the Burmester-Desmedt group key exchange scheme [13]. As a matter of fact, any N-party DPWA-type PAKE scheme, as a complex protocol, can be regarded as a combiner of 2-party PAKE protocol and a group key exchange protocol, whose security also depends on both the securities of the above two primitives to some extent. But if either primitive is broken to pieces which are combined with the other primitive or its pieces in the designation of an N-party DPWA-type PAKE scheme, it will lose its original security, which raises the complexity of the security analysis of the resulting scheme and the possibility of generating insecure protocols such as the case of the two schemes of Byun and Lee [14]. So, it is reasonable to consider using the two primitives as independent units in building an N-party DPWA-type PAKE scheme, respectively. By doing this, their original securities remains and the security of resulting scheme can be directly reduced to the securities of the two primitives, which can largely simplify the security proof of the whole protocol. Moreover, if semantically secure 2-party password-based key exchange protocols already exist between the server and each group member in a distributed system, they can be reused as building blocks in the construction of our N-party DPWA-type PAKE scheme.

On the other hand, the security analysis of our scheme does not resort to any random oracle. That is, if the underlying building primitives are secure in the standard model, so is the resulting scheme. For instance, if one makes use of the KOY protocol [20] as the 2PAKE building blocks, one gets a N-party PAKE scheme whose security can be proved in the standard model.

5.1 Description of Our Scheme

Let $U_1, ..., U_n$ be the users, who share a password with the trusted server S, respectively and wish to establish a session key, and let $U = U_1| \cdots |U_j| \cdots |U_n$. Further, we assume that these participants mentioned above are arranged in a ring with respect to the lexicographic order of their identities so that $U_1 = U_{n+1}$.

Phase 1: Each user U_i begins by choosing a random nonce $r_i \in \{1, 0\}$ and broadcasting $U_i|0|r_i$. After receiving the initial broadcast message from all other parties, each instance stores $U|r_1|, ..., |r_n|$ as part of its state information. The session $N = U|r_1|, ..., |r_n|$ is then defined.

Phase 2: Each group principal U_i engages in a secure 2-party password-based authenticated key exchange protocol to build a session key sk_i with the trusted server concurrently.

Phase 3: After the session key sk_i between it and the trusted server S has been established, each player U_i chooses a random exponent x_i, computes $z_i = g^{x_i}$ and sends to the server S $MacMsg(z_i, U_i, S) = \{z_i, U_i, S, N, MAC_{sk_i}(z_i, U_i, S, N)\}$, which is the authentication message from client to server.

Phase 4: After receiving the authentication message from U_{i-1} and U_{i+1}, the trusted server S checks them. If both of them are legal, S returns to user U_i two messages $MacMsg(z_{i-1}, S, U_{i-1}) = \{z_{i-1}, S, U_{i-1}, N, MAC_{sk_i}(z_{i-1}, S, U_{i-1}, N)\}$ and $MacMsg(z_{i+1}, S, U_{i+1}) = \{z_{i+1}, S, U_{i+1}, N, MAC_{sk_i}(z_{i+1}, S, U_{i+1}, N)\}$.

Phase 5: Each player U_i checks the authentication messages from the server S. If valid, it computes $T_i = z_{i-1}^{x_i}$ and $T_{i+1} = z_{i+1}^{x_i}$, and then broadcasts message $Z_i = T_{i+1}/T_i$.

Phase 6: Each player U_i computes the group session key $SK = (T_i)^n {Z_i}^{n-1} {Z_{i+1}}^{n-2} \cdot\cdot\cdot Z_{i+n-2}$. This key is equal to $g^{x_1x_2+x_2x_3+\cdots+x_nx_1}$, which is same for all $1 \leq i \leq n$.

Finally, the key confirmation as an additional phase is necessary in our scheme, in which each player computers its authenticator as in [15] and broadcasts it. Our scheme is shown schematically in the figure 2 (the key confirmation stage is omitted).

5.2 Security Analysis of Our Scheme

The fundamental security goal for a N-party DPWA-type PAKE protocol to achieve is Authenticated Key Exchange (with "implicit" authentication), in which each participator is assured that no other player aside from the arbitrary pool of participators can learn any information about the session key. Another stronger highly desirable goal for such scheme to provide is Mutual Authentication (MA), which include two aspects: one is MA among participators (or called the key confirmation), namely, each player is assured that its partners actually have possession of the distributed session key; the other is MA between each client and the server, in which the two parts identify each other by the shared password. As mentioned above, the former is always achieved by adding a key confirmation round, but the latter is an intractable hidden trouble for all password-based scheme. In this section, we consider the AKE semantical security and the authentication security from clients to the server on the DGPAKE scheme, and provide the two corresponding security theorems as follows.

Theorem 1. *Let us consider the DGPAKE scheme, where it is assumed that there are n honest users who participate $q_{session}$ honest executions of the scheme, and where 2PAKE is a semantically secure 2-party PAKE protocol and MAC is a secure MAC algorithm. Let q_{exe} and q_{reveal} represent the number of queries to Execute and Reveal oracles, and let $q_{send}^{U_i^s}$ represent the number of queries to the SendClient and SendServer oracles with respect to 2PAKE protocols between the user instance U_i^s and the corresponding server instance S^l. Then,*

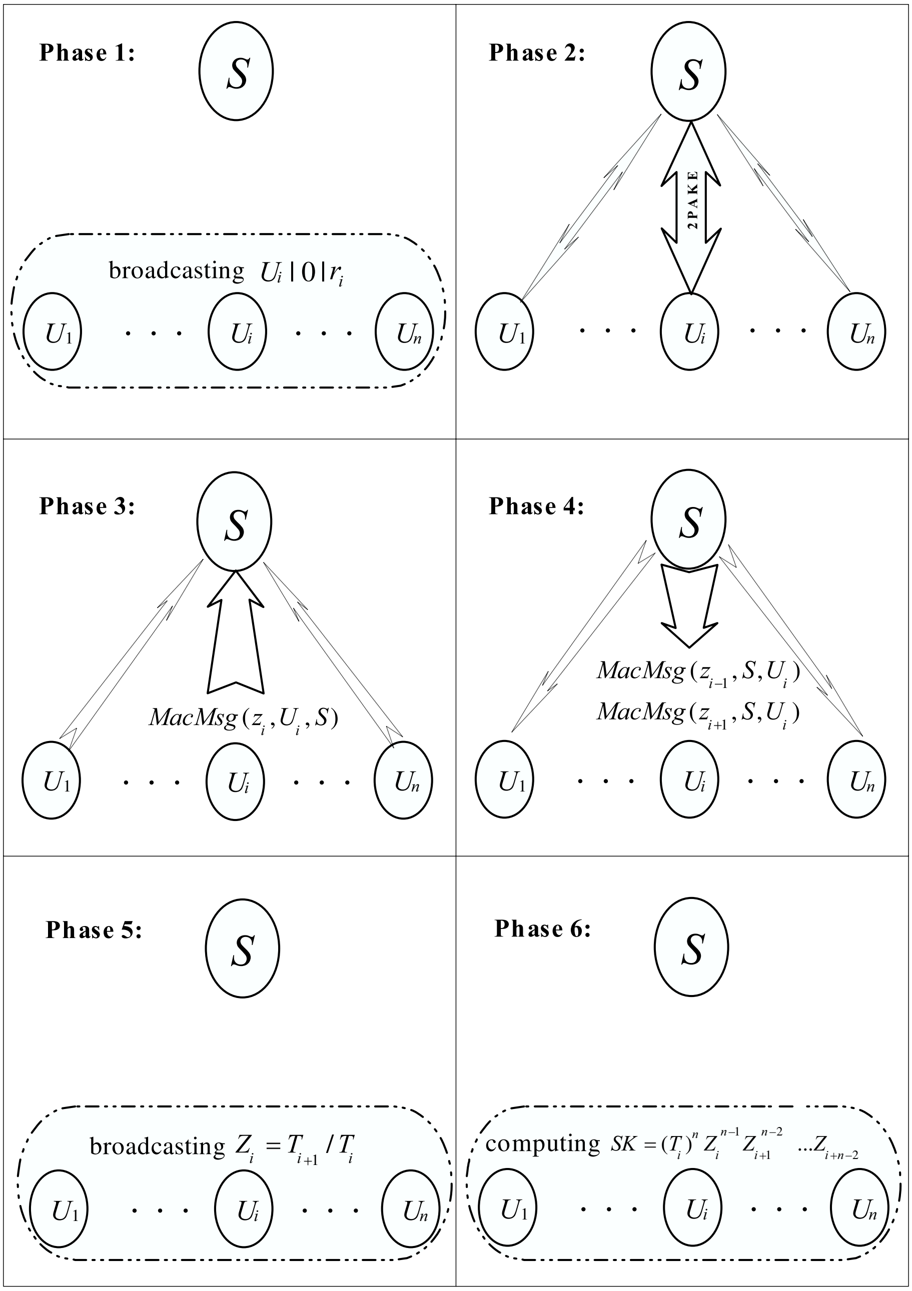

Phase 1:
S
broadcasting $U_i \| 0 \| r_i$
U_1 · · · U_i · · · U_n
Phase 2:
S
2PAKE
U_1 · · · U_i · · · U_n
Phase 3:
S
$MacMsg(z_i, U_i, S)$
U_1 · · · U_i · · · U_n
Phase 4:
S
$MacMsg(z_{i-1}, S, U_i)$
$MacMsg(z_{i+1}, S, U_i)$
U_1 · · · U_i · · · U_n
Phase 5:
S
broadcasting $Z_i = T_{i+1} / T_i$
U_1 · · · U_i · · · U_n
Phase 6:
S
computing $SK = (T_i)^n Z_i^{n-1} Z_{i+1}^{n-2} \ldots Z_{i+n-2}$
U_1 · · · U_i · · · U_n

Fig. 2. DGPAKE: a general construction of N-party DPWA-type PAKE protocol

$$\begin{aligned} &Adv^{ake}_{DGPAKE,\mathcal{D}}(t, q_{exe}, n \cdot q_{session} \cdot q^{U}_{send}, q_{session}) \leq \\ &\quad 2 \cdot n \cdot q_{session} \cdot Adv^{ake}_{2PAKE,\mathcal{D}}(t, q_{exe}, q^{U}_{send}, q_{exe} + q^{U}_{send}) \\ &\quad + 2 \cdot n \cdot q_{session} \cdot Adv^{euf-cma}_{MAC}(t, 3, 0) \\ &\quad + 2 \cdot n \cdot q_{session} \cdot Adv^{ddh}_{\mathbb{G}}(t), \end{aligned}$$

where $q^{U}_{send} = \max_{(i,s)}\{q^{U_i^s}_{send}\}$.

Theorem 2. *Consider the DGPAKE scheme, where n honest users are assumed to participate each of $q_{session}$ honest executions of the scheme, and where 2PAKE is a semantically secure 2-party PAKE protocol and MAC is a secure MAC algorithm. Let q_{exe} and q_{reveal} represent the number of queries to Execute and Reveal oracles, and let $q^{U_i^s}_{send}$ represent the numbers of queries to the SendClient and SendServer oracles with respect to 2PAKE protocols between the user instance U_i^s and the corresponding server instance S^l. Then,*

$$\begin{aligned} &Succ^{auth(C \to S)}_{DGPAKE}(t, q_{exe}, n \cdot q_{session} \cdot q^{U}_{send}, q_{session}) \leq \\ &\quad n \cdot q_{session} \cdot Adv^{ake}_{2PAKE,\mathcal{D}}(t, q_{exe}, q^{U}_{send}, q_{exe} + q^{U}_{send}) \\ &\quad + n \cdot q_{session} \cdot Adv^{euf-cma}_{MAC}(t, 3, 0), \end{aligned}$$

where $q^{U}_{send} = \max_{(i,s)}\{q^{U_i^s}_{send}\}$.

The proofs of the theorems above can be found in the full version of this paper [30].

As a full treatment of resisting undetectable on-line dictionary attacks, it should consider mutual authentication securities both among clients and between clients and the server. As mentioned above, we only present the authentication security from clients to the server, while the one from the server to clients and the one among clients are not provided. As matter of fact, it is generally considered that the two securities may be achieved by adding the key confirmation stage mentioned above in section 4.1, but its formal proof is still an open problem and may resort to the random oracle.

On the other hand, though key privacy and forward security of our scheme DGPAKE is not formally considered in this paper, it is not difficult to prove the two securities in DGPAKE if we formally deal with our scheme as in [2, 22].

6 Conclusion

In this paper, we present the first N-party DPWA-type PAKE scheme, DG-PAKE, that is of constant round and prove its semantical security and unilateral authentication security in the standard model. Furthermore, all the securities of DGPAKE are held in the Find-Then-Guess model, instead of in the Real-Or-Random model defined by Abdalla et al. [2]. This is since that we consider that our scheme as a general construction for N-party DPWA-type PAKE protocols, should adapt for more cases as possible. The latter model have been proved stronger than the former one [2]. That is, it is possible that some 2PAKE protocols are secure under the former but not secure under the latter. Although Abdalla et al. [2] claimed that KOY protocol [20] and PAK suit protocols [24] are still secure under the latter, no strict proof for them is presented until now. Thus, it is desirable for us to prove the scheme secure under the former even with a bit more complex proof procedures.

References

1. Abdalla, M., Bresson, E., Chevassut, O., Pointcheval, D.: Password-based group key exchange in a constant number of rounds. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 427–442. Springer, Heidelberg (2006)
2. Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
3. Abdalla, M., Pointcheval, D.: Interactive diffie-hellman assumptions with applications to password-based authentication. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 341–356. Springer, Heidelberg (2005)
4. Abdalla, M., Pointcheval, D.: A scalable password-based group key exchange protocol in the standard model. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 332–347. Springer, Heidelberg (2006)
5. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
6. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
7. Bellare, M., Rogaway, P.: Provably secure session key distribution: the three party case. In: Proceedings of STOC 1995, pp. 57–66. ACM, New York (1995)
8. Bohli, J.-M., Vasco, M.I.G., Steinwandt, R.: Password-authenticated constant-round group key establishment with a common reference string. In: Cryptology ePrint Archive, Report 2006/214 (2006), `http://eprint.iacr.org/`
9. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using diffie-hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
10. Bresson, E., Chevassut, O., Pointcheval, D.: Provably authenticated group diffie-hellman key exchange - the dynamic case. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 290–309. Springer, Heidelberg (2001)
11. Bresson, E., Chevassut, O., Pointcheval, D.: New security results on encrypted key exchange. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 145–158. Springer, Heidelberg (2004)
12. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.-J.: Provably authenticated group diffie-hellman key exchange. In: Proceedings of CCS 2001, November 2001, vol. 2248, pp. 255–264. ACM Press, New York (2001)
13. Burmester, M., Desmedt, Y.: A secure and efficient conference key distribution system (extended abstract). In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)

14. Byun, J.W., Lee, D.H.: N-party encrypted diffie-hellman key exchange using different passwords. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 75–90. Springer, Heidelberg (2005)
15. Byun, J.W., Lee, D.H., Lim, J.: Password-based group key exchange secure against insider guessing attacks. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3802, pp. 143–148. Springer, Heidelberg (2005)
16. Byun, J.W., Lee, S.-M., Lee, D.H., Hong, D.: Constant-round password-based group key generation for multi-layer ad-hoc networks. In: Clark, J.A., Paige, R.F., Polack, F.A.C., Brooke, P.J. (eds.) SPC 2006. LNCS, vol. 3934, pp. 3–17. Springer, Heidelberg (2006)
17. Dutta, R., Barua, R.: Password-based encrypted group key agreement. International Journal of Network Security 3(1), 30–41 (2006)
18. Goldreich, O., Lindell, Y.: Session-key generation using human passwords only. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 408–432. Springer, Heidelberg (2001)
19. Jablon, D.P.: Strong password-only authenticated key exchange. ACM Computer Communication Review 26, 5–26 (1996)
20. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
21. Katz, J., Ostrovsky, R., Yung, M.: Forward secrecy in password-only key exchange protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 29–44. Springer, Heidelberg (2003)
22. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
23. Lee, S.-M., Hwang, J.Y., Lee, D.H.: Efficient password-based group key exchange. In: Katsikas, S.K., López, J., Pernul, G. (eds.) TrustBus 2004. LNCS, vol. 3184, pp. 191–199. Springer, Heidelberg (2004)
24. MacKenzie, P.D.: The pak suite: Protocols for password-authenticated key exchange. In: Submission to IEEE P1363.2 (2002)
25. MacKenzie, P.D., Patel, S., Swaminathan, R.: Password-authenticated key exchange based on rsa. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 599–613. Springer, Heidelberg (2000)
26. Phan, R.C.-W., Goi, B.-M.: Cryptanalysis of the n-party encrypted diffie-hellman key exchange using different passwords. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 226–238. Springer, Heidelberg (2006)
27. Shin, S.H., Kobara, K., Imai, H.: A lower-bound of complexity for rsa-based password-authenticated key exchange. In: Chadwick, D., Zhao, G. (eds.) EuroPKI 2005. LNCS, vol. 3545, pp. 191–205. Springer, Heidelberg (2005)
28. Tang, Q., Chen, L.: Weaknesses in two group diffie-hellman key exchange protocols. In: Cryptology ePrint Archive, Report 2005/197 (2005), `http://eprint.iacr.org/`
29. Wang, W., Hu, L.: Efficient and provably secure generic construction of three-party password-based authenticated key exchange protocols. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 118–132. Springer, Heidelberg (2006)
30. Wang, W., Hu, L., Li, Y.: Provably secure n-party authenticated key exchange in the multicast dpwa setting. Full version of current paper. Available from authors' web pages and ePrint Archive

A Provably Secure One-Pass Two-Party Key Establishment Protocol

K. Chalkias, S.T. Halkidis, D. Hristu-Varsakelis, G. Stephanides, and A. Alexiadis

Computational Systems and Software Engineering Laboratory,
Department of Applied Informatics,
University of Macedonia, Thessaloniki, Greece
{chalkias,halkidis}@java.uom.gr,{dcv,steph}@uom.gr,talex@java.uom.gr

Abstract. For two parties to communicate securely over an insecure channel, they must be able to authenticate one another and establish a common session key. We propose a new secure one-pass authenticated key establishment protocol which is well suited to one-way communication channels. The protocol is examined using an extension of the Bellare-Rogaway model proposed by Blake-Wilson et. al., and is shown to be provably secure, in the sense that defeating the protocol is equivalent to solving a CDH problem. We compare our protocol to existing approaches, in terms of security and efficiency. To the best of our knowledge, ours is the only one-pass protocol that resists general key-compromise impersonation attacks, and avoids certain vulnerabilities to loss of information attacks found in other protocols of its class.

Keywords: One-pass protocols, two-party key agreement, key-compromise impersonation, loss of information.

1 Introduction

In the last few years, there has been increasing interest in secure two-party key agreement protocols, in large part because of the need for protecting communications over public, unreliable channels. In that context, the protection and authenticity of the messages to be exchanged hinges on the establishment of a group symmetric session key. The pioneering work in two-party key establishment was the Diffie-Hellman protocol [14], which nevertheless suffered from several security problems, such as vulnerability to man-in-the-middle attacks. Efforts to improve on the early Diffie-Hellman protocol have given rise to various non-ID based authenticated two-party key agreement protocols, including recently proposed one round, [18,24], two round [6,25] and three round protocols [8,10,22]. A complementary approach has focused on ID-based schemes [27,13,31,30], which achieve their main security goals but may be quite slow because of their extensive use of bilinear pairings [32]. A common disadvantage of the protocols mentioned here is that they impose either a high computational cost or high communication cost in order to provide authentication. Moreover, their majority requires at least two "rounds", making them unsuitable for one-way communication channels (e.g., e-mail, sms, store-and-forward applications).

This paper proposes a new one-pass two-party key establishment protocol that rectifies some of the problems associated with existing one-pass approaches. The proposed scheme is a slight adaptation on the basic elliptic curve Diffie-Hellman (ECDH) protocol, equipped with an authentication mechanism based on bilinear pairings. To establish a session key, the sender (the initiator of the one-way protocol) generates an ephemeral key-pair and sends its public part to the receiver. To achieve authentication, a similar technique to the short signature scheme proposed in [9] is used. Finally, time-stamp and identities are used to tie quantities to particular entities and reduce the replay vulnerability.

Our approach is robust to unknown key-share (UK-S) attacks and achieves the highest possible level of security against key-compromise impersonation (K-CI) attacks for one-pass protocols, where an attacker who somehow learns a user's private key can impersonate any other entity to the victim, potentially gaining much more knowledge than by simply having access to the victim's past or future conversations. We will have more to say about this in Section 5.4. Furthermore, our protocol is not affected by a loss of information (LoI) attack which can be mounted against other one-pass approaches.

The remainder of this paper is organized as follows: In Section 2 we fix notation and review some required definitions. Section 3 describes the proposed protocol. A security analysis is presented in Section 4. We discuss various desirable attributes of our protocol in Section 5. Section 6 makes comparisons with other widely used schemes.

2 Preliminaries

For the purposes of this work, we will require an abelian, additive finite group $\mathbb{G}_1$, of prime order q, and an abelian multiplicative group, $\mathbb{G}_2$, of the same order. For example, $\mathbb{G}_1$ may be the group of points on an elliptic curve. We will let P denote the generator of $\mathbb{G}_1$. Also, H_1, H_2, will be two secure hash functions, with $H_1 : \{0,1\}^* \mapsto \mathbb{G}_1$ and $H_2 : \{0,1\}^* \mapsto \{0,1\}^k$, where $k \in \mathbb{Z}_+^*$. We will write $a \in_R S$ to denote an element a chosen at random from S. Finally, $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ will be a bilinear pairing, defined below.

Definition 1. *Let $\mathbb{G}_1$ be an additive cyclic group of prime order q generated by P, and $\mathbb{G}_2$ be a multiplicative cyclic group of the same order. A map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ is called a bilinear pairing if it satisfies the following properties:*

– *Bilinearity: $\hat{e}(aV, bQ) = \hat{e}(abV, Q) = \hat{e}(V, abQ) = \hat{e}(V, Q)^{ab}$ for all $V, Q \in \mathbb{G}_1$ and $a, b \in Z_q^*$.*
– *Non-degeneracy: there exist $V, Q \in \mathbb{G}_1$ such that $\hat{e}(V, Q) \neq 1$.*
– *Efficiency: there exists an efficient algorithm to compute the bilinear map.*

Admissible bilinear pairings can be constructed via the Weil and Tate pairings [12,32]. For a detailed description of pairings and conditions under which they can be applied to elliptic curve cryptography, see [12,32].

3 Proposed Protocol

In this Section we describe a new one-pass two-party authentication key establishment protocol. It is composed of two phases, *protocol initialization* and *protocol running*, described next.

3.1 Protocol Initialization

Consider two players, A and B, which are to establish a common session key. First, A chooses a random $x_A \in \mathbb{Z}_q^*$ as her private key, and computes $Y_A = x_A P$ to be her corresponding public key. Similarly, B chooses a random $x_B \in \mathbb{Z}_q^*$ as his private key, and computes his public key $Y_B = x_B P$. We will let the strings ID_A, ID_B denote the identities of A and B, respectively.

3.2 Protocol Running

To establish a session key, A and B obtain the public key of one another and execute the protocol shown in Table 1.

Table 1. Proposed two-party authenticated key agreement protocol

A $(x_A, Y_A = x_A P)$		**B** $(x_B, Y_B = x_B P)$
$\alpha \xleftarrow{R} \mathbb{Z}_q^*, \quad X_1 = \alpha P$		
$X_2 = \alpha Y_B = \alpha x_B P$		
$Q = H_1(X_2 \vert\vert ID_A \vert\vert ID_B \vert\vert T_1)$		
$Y_1 = x_A Q$		
$sk = H_2(X_2 \vert\vert ID_A \vert\vert ID_B \vert\vert T_1)$	$\xrightarrow{(X_1, Y_1, T_1, ID_A)}$	$X_2 = x_B X_1 = \alpha x_B P$
		$Q = H_1(X_2 \vert\vert ID_A \vert\vert ID_B \vert\vert T_1)$
		$\hat{e}(Y_A, Q) \overset{?}{=} \hat{e}(P, Y_1)$
		$sk = H_2(X_2 \vert\vert ID_A \vert\vert ID_B \vert\vert T_1)$

1. Player A selects a random number $\alpha \in \mathbb{Z}_q^*$, and computes: $X_1 = \alpha P$, $X_2 = \alpha Y_B$, $Q = H_1(X_2||ID_A||ID_B||T_1)$, $Y_1 = x_A Q$, and the session key $sk = H_2(X_2||ID_A||ID_B||T_1)$, where T_1 is a time-stamp and the symbol $||$ denotes string[1] concatenation. Then, A sends X_1, Y_1, T_1 and its identity, ID_A, to B.

[1] When elements of a group appear as arguments of a hash function (e.g., $H_1(X_2||ID_A||ID_B||T_1)$), it will be understood that string representations of these elements are used.

2. Upon receipt of (X_1, Y_1, T_1, ID_A), player B computes $X_2 = x_B X_1 = \alpha Y_B$, $Q = H_1(X_2||ID_A||ID_B||T_1)$ and checks whether $\hat{e}(Y_A, Q) \overset{?}{=} \hat{e}(P, Y_1)$. If equality does not hold, B terminates the protocol. Otherwise, he computes the session key $sk = H_2(X_2||ID_A||ID_B||T_1)$.

Correctness: In an honest execution of the protocol A and B share the common key $sk = H_2(X_2||ID_A||ID_B||T_1)$.

Efficiency: Our *one-pass* protocol requires four integer-to-point multiplications, two map-to-point hashes, two plain hashes and two pairings. The computational cost of pairings will be discussed further in Section 6.3.

4 Provable Security

Although the concept of provable security has occasionally taken some criticism, it remains one of the few formal approaches used to make precise statements regarding the security of protocols, and therefore continues to be widely used. In this work, we prove the security of our protocol in an environment which is an extension of the Bellare-Rogaway model [5] proposed by Blake-Wilson et. al. [7]. Our choice of [7] is partly motivated by the fact that in our protocol, authentication and key generation are intertwined (for example, X_2 is used to generate both Q and the secret key). In particular, this means that the protocol is not amenable to the "modular" approach of Canneti-Krawzcyk [11] where a protocol is first proved secure in the absence of authentication and then an appropriate authentication layer is added.

4.1 The Computational Diffie-Hellman Problem

The security of our protocol will turn out to be linked to the well-known Computational Diffie-Hellman (CDH) problem. The CDH problem in $\mathbb{G}_1$ [14], is to compute $\alpha\beta P$ given $P, \alpha P$, and βP, for some fixed $\alpha, \beta \in \mathbb{Z}_q^*$. It is considered computationally hard, assuming an adversary that runs in polynomial time. In our protocol, we have precisely an instance of the CDH problem, because the adversary must compute $\alpha x_B P$ given αP (transmitted by A) and $x_B P$ (B's public key), in order to find the session key.

4.2 Security Analysis

We proceed to show that the proposed protocol is secure in the random oracle model [5,7], assuming that the CDH problem is hard in $\mathbb{G}_1$. We first give a brief, intuitive description of the main assumptions regarding the environment in which the protocol is run. Additional details can be found in [7].

We consider a collection of entities-players which may communicate with one another by initiating the protocol. An adversary can eavesdrop on communications, reroute or change the content of messages, initiate sessions between entities, engage in multiple sessions with the same entity at the same time, and ask an entity to enter a session with itself. The adversary is a probabilistic polynomial-time Turing Machine and has access to a collection of oracles $\{\Pi^s_{i,j}\}$, where $\Pi^s_{i,j}$ behaves as entity i carrying out a protocol session in the belief that it is running the protocol with entity j for the s-th time. We will assume that $1 \leq s \leq \tau(k)$, where τ is polynomial in the security parameter k of the protocol.

The adversary may choose to give an oracle a message (of the form $(X_1, Y_1, ID_A, ID_B, T)$) as an input, via a `Send` query. A `Reveal` query tells an oracle to reveal its current session key. An oracle which has been asked a `Reveal` query is said to be *opened*. Finally, a `Corrupt` query, targeted at entity i, tells all $\Pi^s_{i,j}$ oracles to reveal i's long-term private key to the adversary and to replace i's key pair with any valid key pair chosen by the adversary. An oracle $\Pi^s_{i,j}$ for which a `Corrupt` query has been asked for i, is said to be *corrupted*. We say informally that an oracle has *accepted*, if it has successfully terminated a protocol run. An oracle Π^s_{ij} is *fresh* if it has accepted, i and j are both uncorrupted, it is unopened, and there is no opened oracle $\Pi^t_{j,i}$ with which it has had a *matching conversation*[2], as defined in [5,7].

In the setting described above, the adversary begins by asking the oracles all the queries it wishes to make. He then selects any fresh oracle $\Pi^s_{i,j}$ and asks a single new query, called `Test`. To answer the query, the oracle flips a fair coin $b \overset{R}{\leftarrow} \{0,1\}$ and returns the session key if $b = 0$ or a random value if $b = 1$. The adversary "wins" at this experiment if he correctly guesses the value of b.

Definition 2. *A protocol P is a secure One-Pass Authenticated Key establishment (OPAK) protocol if:*

1. *In the presence of a benign adversary on $\Pi^s_{i,j}$ and $\Pi^t_{i,j}$, (i.e., an adversary that does not interfere with the routing or contents of messages) both oracles always accept holding the same session key κ, and this key is distributed uniformly at random on $\{0,1\}^k$, where k is the protocol's security parameter.*
2. *If uncorrupted oracles $\Pi^s_{i,j}$ and $\Pi^t_{i,j}$ have matching conversations then both oracles accept and hold the same session key κ.*
3. *The adversary makes no* `Reveal` *queries.*
4. *Let $GoodGuess^E(k)$ be the probability that the adversary, E, correctly guesses the coin flip at the* `Test` *query. Then,*
   $advantage^E(k) = \left|Pr[GoodGuess^E(k)] - \frac{1}{2}\right|$ is negligible[3].

In the following we will show that the protocol described in Table 1 is a secure OPAK protocol.

[2] Intuitively, two oracles are said to have matching conversations if one of them is the initiator of an exchange of messages, and the messages sent by each of the two are identical to those received by the other, and are in the same temporal order. See [5,7] for a precise definition. In our case, there is only one message $msg = (X_1, Y_1, T_1, ID_A)$ transmitted during the protocol, so that matching conversations have a particularly simple form: the initiator oracle takes as input the empty string, λ, and transmits msg; the responder takes msg as input and transmits λ.

[3] A real-valued function $\epsilon(k)$ is called negligible if for every $c > 0$ there exists a $k_c > 0$ such that $\epsilon(k) < k^{-c}$ for all $k > k_c$.

Remark 1. We note that Def. 2 is a modified version of Def. 10 in [7]. Our condition 3 is necessary because all one-pass protocols are prone to replay attacks, thus the strongest property of Authenticated Key (AK) security [7] cannot be achieved. If `Reveal` queries were allowed then an adversary E could win the `Test` by a replay attack. E could submit the same properly formatted message to two receiver instances (formally, E initiates oracles Π^s_{AB}, Π^u_{AB}, corresponding to two distinct sessions between A and B). Both receiver instances will accept and produce the same output session key. Then, E `Reveals` the key of one of the sessions, `Tests` against the other session, and wins.

Remark 2. If one insists on keeping `Reveal` queries in play, then the weakness with respect to replay attacks can be somewhat rectified in practical settings if each entity maintains a list of secret keys used during the latest time period. The length of that list would depend on the frequency with which T_1 is updated. An entity would be alerted to a replay attack if a session key it generates is present in the list. Session keys from previous time periods need not be memorized because they could not be successfully replayed to a received.

Remark 3. The notion of OPAK security could actually be strengthened slightly, by allowing the adversary to make `Reveal` queries, but stipulating that if such a query is issued to an oracle Π^s_{AB}, all oracles Π_{AB} are marked as opened.

Theorem 1. *The protocol shown in Table 1 is a secure OPAK protocol, provided that the CDH problem is computationally hard and H_1, H_2 are independent random oracles.*
Proof: See Appendix.

5 Protocol Attributes

In the following we discuss a series of security attributes as they pertain to our protocol. We will forgo formal proofs where applicable because of space limitations.

5.1 Known Session-Keys

Known session-key security [7], also known as known-key security (K-KS), means that a protocol still achieves its goal in the face of an adversary who has learned some previous session keys. One-pass protocols without time-stamps cannot achieve K-KS since an adversary can simply replay the information from a previous protocol run. time-stamps allow a one-pass protocol to achieve some measure of K-KS, meaning that they reduce but do not eliminate the vulnerability to attacks. More specifically, entity B can check the time-stamp T_1 sent by A, and terminate the protocol if too much time has elapsed since then. Of course, this requires synchronization of A's and B's clocks, to within some reasonable tolerance. Depending on the transmission delay imposed by the communication channel, an entity can set a time threshold that leaves a potential attacker little time to mount a known session key attack. If A's and B's clocks are perfectly synchronized and the transmission delay is known with certainty, then the time left for an attack could be made arbitrarily small. The question of what is an acceptable time threshold will generally be application-dependent, and will not be discussed further here.

5.2 Forward Secrecy

Perfect forward secrecy (PFS) means that if long term secrets of one or more entities are compromised, the secrecy of previously computed session keys is not affected. Our protocol does not achieve PFS. To see this, note that if the adversary learns the secret value of B, x_B, then knowing $X_1 = \alpha P$, which is transmitted in the clear, the adversary can compute $X_2 = x_B \alpha P$. Because timestamps are also transmitted in the clear, the adversary can also compute the session key $sk = H_2(X_2||ID_A||ID_B||T_1)$ of a previous session. This is not surprising in light of the fact that there exists no protocol for implicit authentication that achieves PFS with two or fewer messages [21]. However, similarly to the majority of one-pass approaches [21,24], our protocol does achieve partial forward secrecy, because if the adversary learns the secret value of A, x_A, he still faces the CDH problem of computing $\alpha x_B P$ from αP and $x_B P$, before finding any previous session key.

5.3 Unknown Key-Share

Prevention of unknown key-shares (UK-S) as defined in [7], is a property held by all AK-secure protocols. In that setting, an entity i is coerced into sharing a key κ with entity f, while entity j is coerced into holding κ in the belief that it is shared with i. Our OPAK protocol is also secure against UK-S of that type, as well as against the more typical (and broader) UK-S attack [20,28,24,4], where f is not required to possess the key held by i and j, but merely to confuse j as to the identity of i, with whom j shares a key. In our case, UK-S is prevented via the inclusion of the parties' identities in the computation of the session key. A formal proof can easily be constructed around the following basic argument. In order to accomplish a UK-S, the adversary must at least alter the ID information transmitted by i to j. Even if the altered data are such that j's bilinear pairing test is successful, the key computed by j will not be the same as that held by i, thus no key sharing is accomplished. This technique seems to be a general mechanism for preventing UK-S attacks [23].

5.4 Key-Compromise Impersonation

Resistance to key-compromise impersonation (K-CI) attacks means that if i's secret value is disclosed to an adversary, then the adversary cannot impersonate other entities to i [7]. We stress the importance of a protocol being secure against impersonation threats, as an attacker of this type can feign a trusted entity to the victim, and thus ask for (and receive) important information[4].

In fact, there is a special K-CI attack that apparently succeeds with all one-flow protocols. An intruder C that learns B's secret key and then eavesdrops one message from A, (X_1, Y_1, T_1, ID_A), would be able to impersonate A (but *no one else*) to B and only for the current session, because C is also able to create the current session key. However, this attack is more limited than the general K-CI attack, in which the intruder can impersonate *any* entity and at *any* time, to which other one-pass schemes are open (see comparisons in Sec. 6.2).

In our case, if an adversary, C, obtains the secret value of B, x_B, he cannot impersonate another entity, J, to B (excluding the special case described in the previous paragraph); in that sense, our protocol achieves a reduction of the K-CI vulnerability to the greatest extent possible for one-pass approaches. To see why that is, note that C must "pass" the bilinear pairing test, $\hat{e}(Y_J, Q) = \hat{e}(P, Y_1)$, performed by B to successfully impersonate J. If C knows B's private key, x_B, he can initially create a X_1 value from which he can compute Q. C is then faced with the task of computing an appropriate Y_1 such that $\hat{e}(Y_J, Q) = \hat{e}(P, Y_1)$. One can easily show that if C can do this, then with high probability he can compute $x_J Q$ from Q and $x_J P$. By rewriting $Q = kP$ for some unknown k, the last statement is equivalent to C finding (with high probability) $x_J kP$ from kP and $x_J P$, which is an instance of the CDH problem.

Finally, we note that there is no need to examine the case where an adversary obtains the secret value x_A of A and impersonates B to A, since B does not reply to A. In that sence, all of the modern one-pass approaches, including ours, achieve K-CI resilience in the initiator's side [8].

5.5 Loss of Information

Loss of the value X_2 (or loss of Q) from a previous protocol run does not affect the security of subsequent protocol runs, because the computation of X_2 is based on the random value α. Usually a loss-of-information attack succeeds when authentication is used with values which are not random. Some examples are the protocols proposed in [19] and [18], where authentication is based on the computation of $x_A x_B P$ [5]. In contrast to the majority of one-pass key establishment schemes [21,24,1,26,15,29], we do not make use (directly or indirectly) of $x_A x_B P$ values in our approach. We will return to the implications of this in Sec. 6.2. Finally, note that the private key of the receiver (B) is as secure as can be, because B transmits no information during the protocol run.

[4] If a private key is compromised, the attacker is able to intercept messages by eavesdropping on past or future conversations (e.g., e-mails). However, if a communication protocol is vulnerable to K-CI, the attacker would also be able to elicit additional information that may never have been communicated otherwise.

[5] This quantity is widely used in a number of cryptographic protocols; to date, the consequences of its exposure have not been adequately studied in the literature.

5.6 No Key Control and Message Independence

A protocol is said to have the "No Key Control" property if no participant (or adversary) can either control or predict the value of the session key. This property cannot be provided by any one-flow protocol, due to the fact that only the initiator of the protocol produces a random value, and so he can a-priori compute the session key (the receiver does not reply at all). In order to avoid key control, two or more passes are required.

A protocol is said to achieve message independence if individual flows of a protocol run between two honest entities are unrelated [7]. Because in our OPAK protocol only one flow exists, message independence is achieved. This is in contrast to AKC (AK with Key confirmation) protocols [7].

6 Comparison with Existing One-Pass Schemes

This section compares the security and efficiency of the proposed protocol against those of existing one-pass protocols.

6.1 Existing One-Pass Key Establishment Protocols

Because of the need for security in applications where only one entity is on-line, (e.g., secure e-mail, sms, store-and-forward), there have been numerous attempts at designing a secure and efficient one-pass key agreement protocol. The majority of existing one-pass schemes were created as variants of previously-proposed two-way or even two-round key agreement schemes. Some of the best known and most successful approaches to one-pass key establishment protocols include:

- The one-pass variant of the Key Exchange Algorithm (KEA) designed by the NSA and declassified in 1998 [29]. KEA is the key agreement protocol in the FORTEZZA suite of cryptographic algorithms designed by NSA in 1994 and it is similar to the Goss [15] and MTI/A0 [26] protocols.
- The one-pass variant of the Unified Model, proposed by Ankney, Johnson and Matyas [1]; it is an AK protocol that is in the draft standards ANSI X9.42 [2], ANSI X9.63 [3], and IEEE P1363 [17].
- The one-pass variant of the MQV protocol [24] that is in the draft standards ANSI X9.42 [2], ANSI X9.63 [3], and IEEE P1363 [17].
- The one-pass HMQV protocol [21], a variant of MQV.

6.2 Comparison in Terms of Security

Table 2 compares the proposed protocol to those listed in Section 6.1, in terms of various classes of attacks and security-related attributes.

Being one-pass, the protocols listed cannot provide PFS or No Key Control; also, none of the existing one-pass protocols provide K-KS, due to the possibility of replay attacks. The use of time-stamps and the independence of session keys

Table 2. Comparison of Security Properties. IKA denotes implicit key authentication, PFS perfect forward secrecy, K-KS known-key security, UK-S unknown key-share, K-CI key-compromise impersonation, and LoI loss of information ($x_A x_B P$). $\surd$ denotes assurance; $\surd$? indicates that the assurance is provided modulo a technicality; $\surd^-$ denotes assurance, excluding the vulnerability to eavesdropping attacks discussed in Sec. 5.4; $\times^+$ indicates that the assurance is not provided by the protocol, unless modifications are made; $\times$ indicates that the assurance is not provided by the protocol.

	IKA	PFS	K-KS	UK-S	K-CI	LoI
KEA	$\surd$	$\times$	$\surd$?	$\times^+$	$\times$	$\times$
Unified Model	$\surd$	$\times$	$\surd$?	$\surd$	$\times$	$\times$
MQV	$\surd$	$\times$	$\surd$?	$\times^+$	$\times$	$\times$
HMQV	$\surd$	$\times$	$\surd$?	$\times^+$	$\times$	$\times$
Proposed OPAK	$\surd$	$\times$	$\surd$?	$\surd$	$\surd^-$	$\surd$

help reduce (but not eliminate) the vulnerability to known-key attacks. In Table 2 we have used the symbol $\surd$? to indicate the fact that K-KS security should be regarded "modulo" the technicalities discussed in Section 5.1, namely clock synchronization and the time threshold used to decide whether the message A sends to B is "too old". We have marked all protocols listed with $\surd$? under K-KS because we assume that time-stamps can be added to any of them, just as we have done for ours.

Regarding security against UK-S attacks, [20] has presented an on-line UK-S attack on the MQV protocol, and [28] described a UK-S attack on the one-pass HMQV protocol. The KEA one-pass protocol is also prone to UK-S attacks as was shown in [23]. Of course, these protocols can be modified by incorporating parties identities in the computation of a session key, and thus become secure against UK-S threats[6]. This is the reason we have marked KEA, MQV and HMQV with $\times^+$ under UK-S. Unlike these protocols, there are no known UK-S attacks either on the Unified Model or on the protocol proposed here (see also the discussion in Section 5.3).

Of the protocols shown in Table 2, ours provides the highest level of security against K-CI attacks. This is because the bilinear pairing verification works as a short digital signature [9] on the secret key for the specified time period whereas, none of the previous approaches includes a sender verification mechanism. For instance, an exponential challenge-response (XCR) signature (from a player A to a player B), used in the HMQV protocol [21], can also be constructed by anyone who has knowledge of the recipient's private key. The latter means, that if an attacker has knowledge of B's private key, he is able to create a signature of this type and impersonate A to B. At this point, there do not seem to be any obvious modifications that would eliminate the K-CI vulnerability in the first four protocols shown in Table 2. We have marked our protocol with $\surd^-$ under

[6] In some cases (e.g., HMQV), the incorporation of parties' identities is already applied in some of the intermediate steps of the protocol; following our suggestion regarding the computation of the session key then leads to a significantly different protocol.

K-CI because no one-flow scheme can provide perfect K-CI security (for the reasons mentioned in Sec. 5.4). However, our protocol minimizes the vulnerability to such attacks: the class of entities which an adversary in possession of B's private key can impersonate is reduced from any entity / any time (with the KEA, MQV, HMQV and Unified Model protocols) to only an entity that attempts to communicate with B (and only for that session).

Finally, considering the possible loss of the static Diffie-Hellman exponent, $x_A x_B P$, our approach remains secure, as explained in Section 5.5. However, the security of the rest of the protocols examined here depends on the secrecy of this value. In particular, if $x_A x_B P$ were known to an adversary, an impersonation attack can easily take place. For instance, in the one-pass HMQV protocol, an adversary, say E, knowing[7] $L = x_A x_B P$, can initiate a new session with B by impersonating A to him. Using the same notation for the private/public keys as in our protocol, a session key between A and B, $sk = (\alpha + x_A d)Y_B$ (where $d = \bar{H}(X_1, ID_A, ID_B)$ [21]), could also be computed by E as $sk = \alpha Y_B + dL$. The same attack can be mounted against the Unified Model, KEA and MQV one-pass protocols.

6.3 Comparison in Terms of Computational Efficiency

Among the protocols examined, MQV and HMQV are the most efficient. They require two scalar multiplications for the initiator (A) and two for the receiver (B). For the KEA and Unified Model protocols, the corresponding figures are three and two, respectively. Under our protocol, the computational cost for the initiator is a bit higher (three scalar multiplications and one map-to-point hash)[8]. The recipient performs one scalar multiplication in order to create the session key, but must also compute one map-to-point hash and two bilinear pairings in order to verify the sender. Based on the estimates in [16,33], the computational cost of one bilinear pairing is approximately equal to that of four scalar multiplications in the abelian group $\mathbb{G}_1$when using a subgroup of order q in a supersingular elliptic curve E over F_p, where p is a 512 bit prime and q is a 160 bit prime. The pairing computation is the most expensive part of our protocol, but this is the price to be paid for improved security. We emphasize, however, that the reason for choosing the pairing-based short signature (BLS) scheme of [9] is that its signature length is half the size of a DSA signature for a similar level of security and thus the lowest communication cost is achieved.[9]. Because of this, and because pairings are only computed by the recipient, our protocol

[7] this would require E to compromise an earlier session key of B; prior to that, it is possible for E to send a properly formatted message to B such that B's session key is a known multiple of L.

[8] We ignored operations whose cost is negligible compared to that of a scalar multiplication in $\mathbb{G}_1$. These include generating random numbers, integer multiplication, plain hashes and point additions in $\mathbb{G}_1$.

[9] It is a fact that in cases where computational complexity on the receiver's side is of much more importance, one could replace BLS signatures with a more efficient scheme based on discrete logs.

is best-suited to systems where initiators use mobile or low-power devices, as well as in cases where data is sent over a low-bandwidth channel. For example, in smartcard transactions, the initiator of the protocol (e.g., hardware on the smartcard) will not have to compute any pairings; that cost will be borne by the typically more powerful CPU attached to the card reader.

7 Conclusions

This work proposed a new OPAK protocol that addresses the major security problems existing in protocols of its category. We have used the extension to the Bellare-Rogaway model [5] by Blake-Wilson et al. [7] to prove the security of the protocol. We have compared the proposed protocol to existing one-pass schemes, including the MQV (proposed by NSA as the standard key exchange protocol for the US government) and HMQV protocols, and discussed its strengths in several aspects of security. Our protocol achieves the highest level of security possible with one-pass approaches against all widely studied security attacks/properties (IKA, K-KS, UK-S, K-CI, partial forward secrecy and LoI), at the expense of slightly higher computational cost.

The proposed protocol is well-suited to applications with one-way communication channels. Examples include e-mail or sms, where the receiver cannot immediately reply, and store-and-forward applications (e.g., printers) where messages are sent to resources which need not reply at all. The low processing requirements for the protocol's initiator make our approach ideal for use in very low-end computing systems such as smartcards or high-load servers, because the sender does not have to compute any pairings, which are the most costly part of the protocol. Opportunities for further work include the conversion of our protocol to provide security in the standard model, perhaps using [18] as a basis.

References

1. Ankney, R., Johnson, D., Matyas, M.: The Unified Model, Contribution to X9F1 (1995)
2. ANSI X9.42, Agreement of Symmetric Algorithm Keys Using Diffie-Hellman, Working Draft (1998)
3. ANSI X9.63, Elliptic Curve Key Agreement and Key Transport Protocols, Working Draft (1998)
4. Baek, J., Kim, K.: Remarks on the Unknown Key-Share Attacks. IEICE Transactions on Fundamentals E83-A(12) (2000)
5. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 110–125. Springer, Heidelberg (1994)
6. Bird, R., Gopal, I., Herzberg, A., Janson, P., Kutten, S., Molva, R., Yung, M.: Systematic Design of Two-Party Authentication Protocols. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 44–61. Springer, Heidelberg (1992)
7. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 30–45. Springer, Heidelberg (1997)

8. Blake-Wilson, S., Menezes, A.: Authenticated Diffie-Hellman key agreement protocols. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 339–361. Springer, Heidelberg (1999)
9. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology 17(4), 297–319 (2004)
10. Boyd, C., Mao, W., Paterson, K.G.: Key Agreement Using Statically Keyed Authenticators. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 248–262. Springer, Heidelberg (2004)
11. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
12. Cha, J., Cheon, J.: An Id-Based Signature from Gap-Diffie-Hellman Groups. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, Springer, Heidelberg (2002)
13. Chen, L., Kudla, C.: Identity Based Authenticated Key Agreement Protocols from Pairings. In: Proc. 16th IEEE Computer Security Foundations Workshop - CSFW 2003 (2003)
14. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
15. Goss, K.C.: Cryptographic Method and Apparatus for Public Key Exchange with Authentication, U.S. Patent 4956865 (1990)
16. Harrisson, K., Page, D., Smart, N.P.: Software Implementation of Finite Fields of Characteristic Three for Use in Pairing Based Cryptosystems. LMS Journal of Computer Mathematics 5(1), 181–193 (2002)
17. IEEE 1363, Standard Specifications for Public Key Cryptography, Working Draft (1998)
18. Jeong, I.R., Katz, J., Lee, D.H.: One-Round Protocols for Two-Party Authenticated Key Exchange. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 220–232. Springer, Heidelberg (2004)
19. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. In: Bosma, W. (ed.) ANTS 2000. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)
20. Kaliski Jr., B.S.: An Unknown Key-Share Attack on the MQV Key Agreement Protocol. ACM Transactions on Information and Systems Security 4(3), 275–288 (2001)
21. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
22. Kwon, T.: Authentication and Key Agreement via Memorable Password. In: NDSS 2001 Symposium Conference Proc. (2001)
23. Lauter, K., Mityagin, A.: Security Analysis of KEA Authenticated Key Exchange Protocol. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 378–394. Springer, Heidelberg (2006)
24. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Designs, Codes and Cryptography 28(2), 119–134 (2003)
25. Lu, R., Cao, Z., Su, R., Shao, J.: Pairing-Based Two-Party Authenticated Key Agreement Protocol, `http://eprint.iacr.org/2005/354.pdf`
26. Matsumoto, T., Takashima, Y., Imai, H.: On Seeking Smart Public-Key Distribution Systems. Transactions of the IECE of Japan E69, 99–106 (1986)

27. McCullagh, N., Barreto, P.S.L.M.: A New Two-Party Identity-Based Authenticated Key Agreement. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 262–274. Springer, Heidelberg (2005)
28. Menezes, A.: Another look at HMQV, IACR Eprint archive (2005), `http://eprint.iacr.org/2005/205`
29. NIST, SKIPJACK and KEA Algorithm Specification (1998), `http://csrc.nist.gov/encryption/skipjack/skipjack.pdf`
30. Scott, M.: Authenticated ID-based Key Exchange and remote log-in with simple token and PIN Number (2002), `http://eprint.iacr.org/2002/164`
31. Smart, N.P.: An Identity Based Authenticated Key Agreement Protocol based on the Weil Pairing. Electronics Letters 38(13), 630–632 (2002)
32. Stögbauer, M.: Efficient Algorithms for Pairing-Based Cryptosystems, Diploma Thesis: Department of Mathematics, Darmstadt University of Technology (2004)
33. Yoon, H.J., Cheon, J.H., Kim, Y.: Batch verifications with ID-based signatures. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 233–248. Springer, Heidelberg (2005)

Appendix: Proof of Theorem 1

Conditions 1 and 2: From the protocol definition it is clear that, in the presence of a benign adversary on Π_{ij}^s and Π_{ji}^t, as well as when uncorrupted oracles Π_{ij}^s and Π_{ji}^t have matching conversations, both oracles accept and hold the same session key $sk = H_2(ax_jP \| ID_i \| ID_j \| T_1)$.
Condition 3: Consider an adversary, E. We will show that if $advantage^E(k)$ is non-negligible we are led to a contradiction, and in particular that it is then possible to construct a new adversary, F, which succeeds in solving the CDH problem with non-negligible probability.

Suppose that E chooses some fresh oracle, $\Pi_{i,j}$, to ask its `Test` query. The key held by such an oracle will be of the form $H_2(ax_jP \| ID_i \| ID_j \| T_1)$. Let Γ_k be the event that H_2 has previously been queried on an input that begins with ax_jP, by E or by some oracle other than a $\Pi_{i,j}$ or $\Pi_{j,i}$ oracle. If $advantage^E(k)$ is non-negligible, then $Pr[E\ succeeds] = \frac{1}{2} + n(k)$, for some non-negligible function $n(k)$. At the same time,

$$Pr[E\ succeeds] = Pr[E\ succeeds|\Gamma_k]Pr[\Gamma_k] + Pr[E\ succeeds|\bar{\Gamma}_k]Pr[\bar{\Gamma}_k]. \quad (1)$$

Because H_2 is a random oracle and $\Pi_{i,j}^s$ remains unopened (by the definition of a fresh oracle), $Pr[E\ succeeds|\bar{\Gamma}_k] = 1/2$. Therefore,

$$\frac{1}{2} + n(k) \le Pr[E\ succeeds|\Gamma_k]Pr[\Gamma_k] + \frac{1}{2}, \quad (2)$$

which implies that $Pr[E\ succeeds|\Gamma_k] \ge n(k)$, and $Pr[\Gamma_k] \ge n(k)$, are both non-negligible. Let $Pr[\Gamma_k] = n_1(k)$, for some non-negligible function $n_1(k)$. E can now be used to construct an adversary F that solves the CDH problem with non-negligible probability. Given P (the generator of the group $\mathbb{G}_1$), aP (akin to the X_1 value transmitted by i), and x_jP (j's public key), with randomly chosen $a \in_R \mathbb{Z}_q^*$ and $x_j \in_R \mathbb{Z}_q^*$, F must guess ax_jP. This is precisely an instance of the

CDH problem. The construction of F follows that in Case 1 of Theorem 9 in [7]. Let I be the set of entities with which E may communicate. F chooses a pair of entities $i, j \in_R I$, guessing that E, or an oracle other than $\Pi_{i,j}$ or $\Pi_{j,i}$ will query H_2 with $ax_jP \| ID_i \| ID_j \| T_1$ at some time T_1. F performs E's experiment and picks all entities' secret values at random, except of those of $\Pi_{i,j}$ and $\Pi_{j,i}$. F records the public keys of all entities and starts E. F answers all H_1 and H_2 queries at random, just like a random oracle. For all `Send`, `Reveal` and `Corrupt` queries made to oracles other than $\Pi_{i,j}$ or $\Pi_{j,i}$, F answers as an honest player would. If E asks i or j a `Corrupt` query then F gives up. When E asks $\Pi_{i,j}$ or $\Pi_{j,i}$ a `Send` query, instead of computing the value $sk = H_2(ax_jP \| ID_i \| ID_j \| T_1)$ (the key that would have been used by the oracle), F must select a random κ to represent sk, since F does not actually know ax_jP. Finally, if E sends a `Reveal` query to $\Pi_{i,j}$ or $\Pi_{j,i}$, then instead of revealing $H_2(ax_jP \| ID_i \| ID_j \| T_1)$, F responds with his guess, κ, at the session key.

Let $\tau_2(k)$ be a polynomial bound on the number of H_2 queries made by E and its oracles. F chooses $l \in \{1, \ldots, \tau_2(k)\}$, guessing that the l-th distinct input on which H_2 is queried will be $ax_jP \| ID_i \| ID_j \| T_l$, where T_l the time of the l-th query. We note that $T_1, T_2, \ldots, T_{\tau_2(k)}$ are known to the adversary because time-stamps are always transmitted in the clear, as are the entities' IDs. When the l-th distinct H_2 call is made (say, on input g), F stops and outputs g as its guess at $ax_jP \| ID_i \| ID_j \| T_l$ (or, equivalently, F outputs the first part of g corresponding to ax_jP, by discarding the identifiers and time stamp). If E halts before the l-th distinct input is queried then F gives up. Clearly, if the l-th distinct H_2 query made by E or its oracles is on an input that begins with ax_jP, then F wins at this experiment and has solved an instance of the CDH problem.

Of course, H_2 may have been queried on $ax_jP \| ID_i \| ID_j \| T_{l'}$ some time *before* the l-th distinct input, in which case F will have answered at random, and his answer may have been in contradiction to one of the keys that has been used by some $\Pi_{i,j}$ or $\Pi_{j,i}$ oracle. In such a case, E's behavior is not specified, thus E is not guaranteed to halt in that case. This problem can be circumvented as in [7], by letting $\tau_3(k)$ be a polynomial bound on E's runtime under ordinary circumstances and requiring that F gives up if he runs E for longer than $\tau_3(k)$, surmising that he must have missed an H_2 query with input $ax_jP \| ID_i \| ID_j \| T_{l'}$.

We conclude that the probability of F coming up with the correct value ax_jP is at least

$$\frac{n_1(k)}{\tau_1^2(k)\tau_2(k)}, \tag{3}$$

where $\tau_1(k)$ is a polynomial bound on the number of entities[10]. The quantity in (3) is non-negligible. We conclude that the polynomial time adversary F has succeeded in finding ax_jP given aP and x_jP, with non-negligible probability, contradicting the assumption that the CDH problem is hard.

[10] We note that the bound (3) can be improved further, because they adversary can limit its attention to H_2 queries made by E where the ID's included in the argument are ID_i and ID_j, as opposed to guessing over $\tau_1^2(k)$ queries.

Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model under Standard Assumption*

Yi Deng^{1,2} and Dongdai Lin^{1}

^{1} The state key laboratory of information security, Institute of software, Chinese Academy of sciences, Beijing, 100080, China
^{2} Graduate School of Chinese Academy of Sciences
{ydeng,ddlin}@is.iscas.ac.cn

Abstract. In this paper we present the *first* constant round resettable zero knowledge arguments with concurrent soundness for $\mathcal{NP}$ in the bare public-key (BPK for short) model assuming only collision-resistant hash functions against *polynomial-time* adversaries. This resolves the problem whether there exist such protocols for $\mathcal{NP}$ in BPK model without assuming *sub-exponential hardness*. In our protocol, the resettable zero knowledge is demonstrated via a black-box simulator, while the concurrent soundness is proved by using the malicious prover strategy in *non-black-box* manner.

Keywords: Resettable Zero Knowledge, Concurrent Soundness, the Bare Public-Key Model, Resettably sound Zero Knowledge.

1 Introduction

Zero knowledge (ZK for short) proof, a proof that reveals nothing but the validity of the assertion, is put forward in the seminal paper of Goldwasser, Micali and Rackoff [18]. Since its introduction, especially after the generality demonstrated in [17], ZK proofs have become a fundamental tools in design of some cryptographic protocols. In recent years, the research is moving towards extending the security to cope with today's malicious communication environment. In particular, Dwork et al. [13] introduced the concept of concurrent zero knowledge, and initiated the study of the effect of executing ZK proofs concurrently in some realistic and asynchronous networks like the Internet. Though the concurrent zero knowledge protocols have wide applications, unfortunately, they requires logarithmic rounds for languages outside $\mathcal{BPP}$ in the plain model for the black-box case [6] and therefore are of round inefficiency. In the Common Reference String model, Damgaard [7] showed that 3-round concurrent zero-knowledge can be achieved efficiently. Surprisingly, by developing non-black-box technique,

* This work is partly supported by the National Natural Science Foundation of China under Grant No. 60673069, National 863 Project of China(No. 2007AA01Z447) and National 973 Project(No. 2004CB318004).

Barak [1] constructed a constant round non-black-box bounded concurrent zero knowledge protocol in the plain model though it is very inefficient.

Resettable ZK and the BPK model. Motivated by the application in which the prover (such as the user of a smart card) may encounter resetting attack, Canetti et al. [5] introduced the notion of resettable zero knowledge (rZK for short). An rZK formalizes security in a scenario in which the verifier is allowed to reset the prover in the middle of proof to any previous stage. Obviously, the notion of resettable zero knowledge is stronger than that of concurrent zero knowledge and therefore we can not implement such (black-box) protocols in constant round in the plain model for non-trivial languages. To get constant round rZK, the work [5] also introduced a very attracting model, the bare public-key model (BPK). In this model, Each verifier deposits a public key in a public file and stores the associated secret key before any interaction with the prover begins. Note that no protocol needs to be run to publish the public key, and no authority needs to check its property. Consequently the BPK model is considered as a very weak set-up assumption compared to previously models such as common reference model and PKI model.

Though the BPK model is very simple, the notion of soundness in this model turned out to be more subtle. As Micali and Reyzin [20] pointed out, there are four distinct notions of soundness: one time, sequential, concurrent and resettable soundness, each of which implies the previous one. Moreover they also showed that there is NO black-box rZK satisfying resettable soundness for non-trivial language and the original rZK arguments in the BPK model of [5] do not seem to be concurrently sound. Since then, a lot of effort has been devoted to construct rZK with concurrent soundness. Several such protocols were suggested in some stronger variants of the BPK model, and the 4-round (optimal) rZK arguments with concurrent soundness in the BPK model was proposed by Di Crescenzo et al. in [11].

However, all above rZK arguments in BPK model (even those rZK arguments in stronger variants of BPK model) heavily rely on some *stronger* assumptions that there exist cryptographic primitives secure against sub-exponential time adversaries. To realize such protocols under some standard assumptions is an interesting problem and has been explicitly posed by Di Crescenzo et al [11]. The first step towards weakening this kind of hardness assumptions was made by Barak et al. [2]: by using non-black-box techniques, they obtained a constant-round rZK argument of knowledge assuming only collision-free hash functions secure against supperpolynomial-time algorithms[1], but their approach fails to achieve concurrent soundness. Thus, the existence of constant round rZK arguments with concurrent soundness in BPK model under only polynomial-time hardness assumption, is still left open.

Our results and techniques. In this paper we resolve the above open problem by presenting a constant-round rZK argument with concurrent soundness in BPK

[1] Using idea from[3], this results also holds under standard assumption that there exist hash functions that are collision-resistent against all polynomial-time adversaries.

model for $\mathcal{NP}$ under the standard assumptions that there exist hash functions collision-resistant against *polynomial time* adversaries.

Most often, the proof of soundness for an argument goes by reduction, that is, using the cheating prover strategy, we construct algorithm to break some problems that is assumed to be hard. There are two obstacles in establishing concurrent soundness in our setting: 1) In many cases, the reduction algorithm's goal is to use a cheating prover to break the cryptographic primitive specified by the public key, but this algorithm seems to need itself knowledge of secret key in order to be able to use the cheating prover's power. Note that in the typical paradigm for this kind of protocols, the prover must be convinced that the verifier knows the secret key before showing his ability that we want to use. 2) The rewinding technique that commonly-used by reduction algorithm seems to be helpless when the cheating prover strategy is used in black-box manner. This is because the malicious verifier is also allowed to rewind the prover in real interaction and will not benefit from this kind of interaction (guaranteed by the resettable zero knowledge property).

One approach to overcoming these obstacles is to let the reduction algorithm has more computational power than a polynomial time verifier, i.e. be a sub-exponential time algorithm. However, on the other hand, we have to assume some target problems to be more harder–they can not be solved in the running time of the reduction algorithm–in order to justify the concurrent soundness. This is so-called complexity leveraging technique initiated in [5] and also used in [11, 25]. Another one is to use non-black-box technique. In Barak et al.'s construction [2], the verifier needs to proves the knowledge the secret key (a committed random seed to a pseudorandom function)to the prover via a resettably-sound (non-black-box) zero knowledge argument of knowledge. The *sequential* soundness of their protocol proceeds by contradiction: If a prover is able to cheat with non-negligible probability, we can construct an reduction algorithm that break either the pseudorandomness of the pseudorandom function or the soundness of the underlying parallel version of the basic proof of Hamilton Cycle. Though this protocol can be built based on more general (polynomial time hardness) assumption, it fails to obtain concurrent soundness: for the analysis of concurrent soundness go through, the reduction algorithm has to simulate the prover's view in full concurrent setting without knowing the random seed to the pseudorandom function, and completing such a simulation seems beyond the reach of current techniques (we do not know so far how to construct constant round concurrent zero knowledge for language outside $\mathcal{BPP}$).

We adopt different approach to bypass those difficulties. First, we let each verifier register two public keys and use a witness indistinguishable (WI) argument to prove that he knows one of secret keys corresponding to the public keys (it is also used in [12, 25], and can be dated back to [14] in spirit). We note that the WI is preserved under concurrent composition and, more important, for the reduction algorithm (used to justify the concurrent soundness), knowledge of a random secret key does not give rise to the first obstacle described above any more because it still has chance to use a cheating prover to break the cryptographic primitive specified by the other public key. We note that this approach will result in a way to prove the soundness from different from the one in [2]: we need to use a *simulated* public keys to justify the soundness (because we need to know one of the secret keys).

To avoid the second obstacle, we use a constant round resettable WI *argument of knowledge* [2] for the prover in which it proves the knowledge of the witness for the statement to be proven or one of the secret keys (we stress this need a little adjustment to avoid the malleable attack, see [12]). Note that this typically involves in a resettably-sound non-black-box ZK argument in which the verifier proves that a challenge matches the one he committed to in a previous step. The major concern that arises here is that we need to run the simulator for the underlying resettably-sound zero knowledge on a false statement in order to show concurrent soundness, and as mentioned above such a simulation in the concurrent setting seems beyond the limits of our current knowledge. The key observation that allows for such a simulation (therefore enables the analysis of concurrent soundness) is that we just need to simulate *only one execution* among all concurrent executions of the resettably-sound zero knowledge argument for justifying concurrent soundness, instead of simulating all these concurrent executions, and this can be done easily (see section 3 for details).

2 Preliminaries

In this section we recall some definitions and tools that will be used later. Due to space limitations, we refer readers to [16] for some basic primitives, such as pseudorandom functions and commitments.

In the following we say that function $f(n)$ is negligible if for every polynomial $q(n)$ there exists an N such that for all $n \geq N$, $f(n) \leq 1/q(n)$.

The BPK Model. The bare public-key model(BPK model)assumes that:

- A public file F that is a collection of records, each containing a verifier's public key, is available to the prover.
- An (honest)prover P is an interactive deterministic polynomial-time algorithm that is given as inputs a secret parameter 1^n, a n-bit string $x \in L$, an auxiliary input y, a public file F and a random tape r.
- An (honest) verifier V is an interactive deterministic polynomial-time algorithm that works in two stages. In stage one, on input a security parameter 1^n and a random tape w, V generates a key pair (pk, sk) and stores pk in the file F. In stage two, on input sk, an n-bit string x and an random string w, V performs the interactive protocol with a prover, and outputs "accept x" or "reject x".

Definition 1. *We say that the protocol* $< P, V >$ *is complete for a language* L *in* $\mathcal{NP}$*, if for all* n*-bit string* $x \in L$ *and any witness* y *such that* $(x, y) \in R_L$*, here* R_L *is the relation induced by* L*, the probability that* V *interacting with* P *on input* y*, outputs "reject* x*" is negligible in* n*.*

Malicious provers and Its attacks in the BPK model. Let s be a positive polynomial and P^* be a probabilistic polynomial-time algorithm on input 1^n.

P^* is a *s-concurrent malicious* prover if on input a public key pk of V, performs at most s interactive protocols as following: 1) if P^* is already running $i-1$ interactive protocols $1 \leq i-1 \leq s$, it can output a special message "Starting x_i," to start a new protocol with V on the new statement x_i; 2) At any point it can output a message for any of its interactive protocols, then immediately receives the verifier's response and continues.

A concurrent attack of a *s-concurrent malicious* prover P^* is executed in this way: 1) V runs on input 1^n and a random string and then obtains the key pair (pk, sk); 2) P^* runs on input 1^n and pk. Whenever P^* starts a new protocol choosing a statement, V is run on inputs the new statement, a new random string and sk.

Definition 2. *$< P, V >$ satisfies* concurrent soundness *for a language L if for all positive polynomials s, for all* s-concurrent malicious *prover P^*, the probability that in an execution of concurrent attack, V ever outputs "accept x" for $x \notin L$ is negligible in n.*

The notion of resettable zero-knowledge was first introduced in [5]. The notion gives a verifier the ability to rewind the prover to a previous state (after rewinding the prover uses the same random bits), and the *malicious* verifier can generate an arbitrary file F with several entries, each of them contains a public key generated by the malicious verifier. We refer readers to that paper for intuition of the notion. Here we just give the definition.

Definition 3. *An interactive argument system $< P, V >$ in the BPK model is black-box resettable zero-knowledge if there exists a probabilistic polynomial-time algorithm S such that for any probabilistic polynomial-time algorithm V^*, for any polynomials s, t, for any $x_i \in L$, the length of x_i is n, $i = 1, ..., s(n)$, V^* runs in at most t steps and the following two distributions are indistinguishable:*

1. *the view of V^* that generates F with $s(n)$ entries and interacts (even concurrently) a polynomial number of times with each $P(x_i, y_i, j, r_k, F)$ where y_i is a witness for $x_i \in L$, r_k is a random tape and j is the identity of the session being executed at present for $1 \leq i, j, k \leq s(n)$;*
2. *the output of S interacting with on input $x_1, ...x_{s(n)}$.*

Witness Indistinguishable Argument of Knowledge (WIAOK) [15]. We will use 3-round WIAOK with negligible knowledge error as a component. Let (a, e, z) be the three messages exchanged in an execution of a WIAOK, we assume a special feature with the knowledge extraction: it is easy to extract a witness form two different transcripts with same first message (a, e, z) and (a, e', z'). Note that we can obtain such a protocol by parallelizing the basic proof of Hamiltonian Cycle [4] or uisng techiques form [8].

In addition, as required in [12], the WIAOK used by the verifier to prove that it knows one of secret keys corresponding to its public key in our construction also needs to satisfy *partial-witness-independence* property: the message sent at its first round should have distribution independent from any witness for the statement to be proved. We note that the basic proof of Hamiltonian Cycle [4] meets this requirement, and we also can establish this property using [9].

The basic 3-round proof of Hamiltonian Cycle is based on the existence of one-way permutation families. If the commitment scheme used by the protocol in [1] is implemented using the scheme in [21] from any pseudo-random generator family, then the assumption can be reduced to the existence of one-way function families, at the cost of adding one preliminary message (i.e., the first message of the Naor's commitment scheme) from the verifier. We stress that adding one message does not affect the properties of WIAOK that we need.

3 A Simple Observation on Resettably-Sound Zero Knowledge Arguments

Resettably-sound zero knowledge argument is a zero knowledge argument with stronger soundness: for all probabilistic polynomial-time prover P^*, even P^* is allowed to reset the verifier V to previous state (after resetting the verifier V uses the same random tape), the probability that P^* make V accept a false statement $x \notin L$ is negligible.

In [2] Barak et al. transform a constant round public-coin zero knowledge argument $< P, V >$ for a $\mathcal{NP}$ language L into a constant round resettably-sound zero knowledge argument $< P, W >$ for L as follows: equip W with a collection of pseudorandom functions, and then let W emulate V except that it generate the current round message by applying a pseudorandom function to the transcript so far.

We will use a resettably-sound zero knowledge argument as a building block in which the verifier proves to the prover that a challenge matches the one that he has committed to in previous stage. The simulation for such a sub-protocol plays a important role in our security reduction, but there is a subtlety in the simulation itself. In the scenario considered in this paper, in which the prover (i.e., the verifier in the underlying resettably-sound zero knowledge argument) is allowed to interact with many copies of the verifier and schedule all sessions at his wish, the simulation seems problematic because we do not know how to simulate concurrent executions of the resettably-sound zero knowledge argument[2]. However, fortunately, it is not necessary to simulate all the concurrent executions of the underlying resettably-sound zero knowledge argument. Indeed, in order to justify concurrent soundness, we just need to simulate *only one execution* among all concurrent executions of the resettably-sound zero knowledge argument. We call this property *one-many simulatability.* We note that Pass and Rosen [24]

[2] Indeed, Barak also presented a constant round bounded concurrent ZK arguments, hence we can obtain a constant round resettably-sound *bounded concurrent* ZK argument by applying the same transformation technique to the bounded concurrent ZK argument. We stress that in this paper we do not require the bounded concurrent zero knowledge property to hold for the resettably-sound ZK argument.

made a similar observation (in a different context) that enables the analysis of concurrent non-malleability of their commitment scheme.

Now we recall the Barak's constant round public-coin zero knowledge argument [1], and show this protocol satisfies *one-many simulatability*, and then so does the resettably-sound zero knowledge argument transformed from it.

Informally, Barak's protocol for a $\mathcal{NP}$ language L consists of two subprotocol: a general protocol and a WI universal argument. An real execution of the general protocol generates an instance that is unlikely in some properly defined language, and in the WI universal argument the prover proves that the statement $x \in L$ or the instance generated in general protocol is in the properly defined language. Let n be security parameter and $\{\mathcal{H}_n\}_{n\in\mathbb{N}}$ be a collection of hash functions where a hash function $h \in \mathcal{H}_n$ maps $\{0,1\}^*$ to $\{0,1\}^n$, and let C be a statistically binding commitment scheme. We define a language Λ as follows. We say a triplet $(h, c, r) \in \mathcal{H}_n \times \{0,1\}^n \times \{0,1\}^n$ is in Λ, if there exist a program Π and a string $s \in \{0,1\}^{poly(n)}$ such that $z = \mathsf{C}(h(\Pi), s)$ and $\Pi(z) = r$ within superpolynomial time (i.e., $n^{\omega(1)}$).

The Barak's Protocol [1]
Common input: an instance $x \in L$ ($|x| = n$)
Prover's private input: the witness w such that $(x, w) \in R_L$
$V \rightarrow P$: Send $h \leftarrow_R \mathcal{H}_n$;
$P \rightarrow V$: Pick $s \leftarrow_R \{0,1\}^{poly(n)}$ and Send $c = \mathsf{C}(h(0^{3n}, s)$;
$V \rightarrow P$: Send $r \leftarrow_R \{0,1\}^n$;
$P \Leftrightarrow V$: A WI universal argument in which P proves $x \in L$ or $(h, c, r) \in \Lambda$.

Fact 1. The Barak's protocol enjoys *one-many simulatability*. That is, For every malicious probabilistic polynomial time algorithm V^* that interacts with (arbitrary) polynomial s copies of P on true statements $\{x_i\}, 1 \le i \le s$, and for every $j \in \{1, 2, ..., s\}$, there exists a probabilistic polynomial time algorithm S, takes V^* and all witness but the one for x_j, such that the output of $\mathsf{S}(V^*, \{(x_i, w_i)\}_{1\le i\le s, i\ne j}, x_j)$ (where $(x_i, w_i) \in R_L$) and the view of V^* are indistinguishable.

It is easy to check the above fact. We can construct a simulator $\mathsf{S} = (\mathsf{S}_{real}, \mathsf{S}_j)$ as follows: S_{real}, taking as inputs $\{(x_i, w_i)\}_{1\le i\le s, i\ne j}$, does exactly what the honest provers do on these statements and outputs the transcript of all but the jth sessions (in jth session $x_j \in L$ is to be proven), and S_j acts the same as the simulator associated with Barak's protocol in the session in which $x_j \in L$ is to be proven, except that when S_j is required to send a commitment value (the second round message in Barak's protocol), it commits to the hash value of the **joint** residual code of V^* and S_{real} at this point instead of committing to the hash value of the residual code of V^* (that is, we treat S_{real} as a subroutine of V^*, and it interacts with V^* internally). We note that the next message of the joint residual code of V^* and S_{real} is only determined by the commitment message from S_j, so as showed in [1], S_j works. On the other hand, the S_{real}'s behavior is identical to the honest provers. Thus, the whole simulator S satisfies our requirement.

When we transform a constant round public-coin zero knowledge argument into a resettably-sound zero knowledge argument, the transformation itself does not influence the simulatability (zero knowledge) of the latter argument because the zero knowledge requirement does not refer to the honest verifier (as pointed out in [2]). Thus, the same simulator described above also works for the resettably-sound zero knowledge argument in concurrent settings. So we have

Fact 2. The resettably-sound zero knowledge argument in [2] enjoys *one-many simulatability.*

Despite the obviousness of the above facts, it is very useful in analysis of security of some global system like ours, in which such protocols are used as subroutine and all statement for those protocols are selected by the honest party on the fly. We will show it in next section.

4 rZK Argument with Concurrent Soundness for $\mathcal{NP}$ in the BPK Model under Standard Assumption

In this section we present a constant-round rZK argument with concurrent soundness in the BPK model for all $\mathcal{NP}$ languages without assuming any subexponential hardness.

For the sake of readability, we give some intuition before describe the protocol formally.

We construct the argument in the following way: build a concurrent zero knowledge argument with concurrent soundness and then transform this argument to a resettable zero knowledge argument with concurrent soundness. Concurrent zero knowledge with concurrent soundness was presented in [12] under standard assumption (without using "complexity leveraging"). For the sake of simplification, we modify the *flawed* construction presented in [25] to get concurrent zero knowledge argument with concurrent soundness. Considering the following two-phase argument in BPK model: Let n be the security parameter, and f be a one way function that maps $\{0,1\}^{\kappa(n)}$ to $\{0,1\}^n$ for some function $\kappa : \mathbb{N} \rightarrow \mathbb{N}$. The verifier chooses two random numbers $x_0, x_1 \in \{0,1\}^{\kappa(n)}$, computes $y_0 = f(x_0)$, $y_1 = f(x_1)$ then publishes y_0, y_1 as he public key and keep x_0 or x_1 secret. In phase one of the argument, the verifier proves to the prover that he knows one of x_0, x_1 using a *partial-witness-independently* witness indistinguishable argument of knowledge protocol Π_v. In phase two, the prover proves that the statement to be proven is true or he knows one of preimages of y_0 and y_1 via a witness indistinguishable argument of knowledge protocol Π_p.

Though the above two-phase argument does not enjoy concurrent soundness (see [12]), it is still a good start point and we can use the same technique in [12] in spirit to fix the flaw: in phase two, the prover uses a commitment scheme Com_1 to compute a commitments to a random strings s, $c = \mathsf{Com}_1(s, r)$ (r is a random string needed in the commitment scheme), and then the prover proves that the statement to be proven is true or he committed to a preimage of y_0 or y_1 using Π_p.

Given the above (modified) concurrent zero knowledge argument with concurrent soundness, we can transform it to resettable zero knowledge argument with concurrent soundness by modifying the WIAOK Π_p to be *resettable* (i.e. resettable WIAOK [2]). So we make the following adjustments: 1) using a statistically-binding commitment scheme Com_0, the verifier computes a commitment $c_e = \mathsf{Com}_0(e, r_e)$ (r_e is a random string needed in the scheme) to a random string e in the phase one, and then he sends e (note that the verifier does not send r_e, namely, it does not open the commitment c_e) as the second message (i.e the challenge) of Π_p and prove that e is the string he committed to in the first phase using resettably-sound zero knowledge argument; 2) equipping the prover with a pseudorandom function, whenever the random bits is needed in a execution, the prover applied the pseudorandom function to what he have seen so far to generate random bits.

Let's Consider concurrent soundness of the above protocol. Imagine that a malicious prover convince a honest verifier of a false statement on a session (we call it a cheating session) in an execution of concurrent attack with high probability. Then we can use this session to break some hardness assumption: after the first run of this session, we rewind it to the point where the verifier is required to send a challenge and chooses an arbitrary challenge and run the simulator for this underlying resettably-sound zero knowledge proof. At the end of the second run of this session, we will extract one of preimages of y_0 and y_1 from the two different transcripts, and this contradicts either the witness indistinguishability of Π_v or the binding property of the commitment scheme Com_1. Note that in the above reduction we just need to simulate the *single* execution of the resettably-sound zero knowledge argument in that cheating session, and do not care about other sessions that initiated by the malicious prover (in other sessions we play the role of honest verifier). We have showed the simulation in this special concurrent setting can be done in a simple way in last section.

The Protocol (rZK argument with concurrent soundness in BPK model)

Let $\{prf_r : \{0,1\}^* \to \{0,1\}^{d(n)}\}_{r \in \{0,1\}^n}$ be a pseudorandom function ensembles, where d is a polynomial function, Com_0 be a *statistically-binding* commitment scheme, and let Com_1 be a general commitment scheme (can be either statistically-binding or computational-binding[3]). Without loss of generality, we assume both the preimage size of the one-way function f and the message size of Com_1 equal n.

Common input: the public file F, n-bit string $x \in L$, an index i that specifies the i-th entry $pk_i = (f, y_0, y_1)$ (f is a one-way function) of F.
P**'s Private input:** a witness w for $x \in L$, and a fixed random string $(r_1, r_2) \in \{0,1\}^{2n}$.
V**'s Private input:** a secret key α ($y_0 = f(\alpha)$ or $y_1 = f(\alpha)$).

[3] If the computational-binding scheme satisfies perfect-hiding, then this scheme requires stronger assumption, see also [23, 22].

Phase 1: V Proves Knowledge of α and Sends a Committed Challenge to P.

1. V and P runs the 3-round *partial-witness-independently* witness indistinguishable protocol Π_v in which V prove knowledge of α that is one of the two preimages of y_0 and y_1. the randomness bits used by P equals r_1;
2. V computes $c_e = \mathsf{Com}_0(e, r_e)$ for a random e (r_e is a random string needed in the scheme), and sends c_e to P.

Phase 2: P Proves $x \in L$.

1. P checks the transcript of Π_v is accepting. if so, go to the following step.
2. P commits to a random string using Com_1 and computes the first message a of the 3-round witness indistinguishable argument of knowledge Π_p in which it proves either it knows the witness for $x \in L$ or it committed to one preimage of y_0 or y_1. That is, P does the following: chooses a random string s, $|s| = n$, and compute $c = \mathsf{Com}_1(s, r_s)$ by picking a randomness r_s; forms a new relation R'=$\{(x, y_0, y_1, c, w') \mid (x, w') \in R_L \vee (w' = (w'', r_{w''}) \wedge y_0 = f(w'') \wedge c = \mathsf{Com}_1(w'', r_{w''})) \vee (w' = (w'', r_{w''}) \wedge y_1 = f(w'') \wedge c = \mathsf{Com}_1(w'', r_{w''}))\}$; invokes Π_p in which P prove knowledge of w' such that $(x, y_0, y_1, c; w') \in R'$, computes and sends the first message a of Π_p.

   All randomness bits used in this step is obtained by applying the pseudorandom function prf_{r_2} to what P have seen so far, including the common inputs, the private inputs and all messages sent by both parties so far.
3. V sends e to P, and execute a resettably sound zero knowledge argument with P in which V proves to P that $\exists\ r_e$ s.t. $c_e = \mathsf{Com}_0(e, r_e)$. Note that the subprotocol will cost several (constant) rounds. Again, the randomness used by P is generated by applying the pseudorandom function prf_{r_2} to what P have seen so far.
4. P checks the transcript of resettably sound zero knowledge argument is accepting. if so, P computes the last message z of Π_p and sends it to V.
5. V accepts if only if (a, e, z) is accepting transcript of Π_p.

Theorem 1. Let L be a language in $\mathcal{NP}$, If there exist one-way permutations and hash functions collision-resistant(both against any polynomial time adversary), then there exists a constant round rZK argument with concurrent soundness for L in BPK model.

Remark on complexity assumption. We prove this theorem by showing the protocol described above is a rZK argument with concurrent soundness. Indeed, our protocol requires collision-resistant hash functions and one-way *permutations*, this is because the 3-round WIAOK for $\mathcal{NP}$ assumes one-way permutations and the resettably sound zero knowledge argument assumes collision-resistant hash functions. However, we can build 4-round WIAOK for $\mathcal{NP}$ assuming existence of one-way functions by adding one message (see also discussions in section 2), and our security analysis can be also applied to this variant. We also note that collision-resistant hash functions implies one-way functions which suffices to build statistically-binding commitment scheme [21] (therefore computational-binding scheme), thus, if we proved our protocol is a rZK argument with concurrent soundness, then we get theorem 1. Here we adopt the 3-round WIAOK just for the sake of simplicity.

Remark on the scheme Com_1. In contrast to [12], we show that computational binding commitment scheme suffices to achieve concurrent soundness. In fact, the statistically binding commitment scheme in [12] could also be replaced with computational binding one without violating the concurrent soundness.

Proof. **Completeness.** Straightforward.

Resettable (*black-box*) Zero Knowledge. The analysis is very similar to the analysis presented in [5, 11]. Here we omit the tedious proof and just provide some intuition. As usual, we can construct a simulator Sim that extracts all secret keys corresponding to those public keys registered by the malicious verifier from Π_v and then uses them as witness in executions of Π_p, and Sim can complete the simulation in expected polynomial time. We first note that when a malicious verifier resets a an honest prover, it can not send two different challenge for a fixed commitment sent in Phase 1 to the latter because of statistically-binding property of Com_0 and resettable soundness of the underlying sub-protocol used by the verifier to prove the challenge matches the value it has committed to in Phase 1. To prove the property of rZK, we need to show that the output of Sim is indistinguishable form the real interactions. This can be done by constructing a non-uniform hybrid simulator HSim and showing the output of HSim is indistinguishable from both the output of Sim and the real interaction. HSim runs as follows. Taking as inputs all these secret keys and all the witnesses of statements in interactions, HSim computes commitments (at step 2 in Phase 2) exactly as Sim does but executes Π_p using the same witness of the statement used by the honest prover. It is easy to see that the output of the hybrid simulator is indistinguishable from both the transcripts of real interactions (because of the computational-hiding property of Com_1) and the output of Sim (because of the witness indistinguishability of Π_p), therefore, we proved the output of Sim is indistinguishable form the real interactions.

Concurrent Soundness. Proof proceeds by contradiction.

Assume that the protocol does not satisfy the concurrent soundness property, thus there is a s-concurrently malicious prover P^*, concurrently interacting with V, makes the verifier accept a false statement $x \notin L$ in jth session with non-negligible probability p.

We now construct an algorithm B that takes the code (with randomness hard-wired in)of P^* as input and breaks the one-wayness of f.

B runs as follows. On input the challenge f, y (i.e., given description of one-way function, B finds the preimage of y), B randomly chooses $\alpha \in \{0,1\}^n$, $b \in \{0,1\}$, and guess a session number $j \in \{1, ..., s\}$(guess a session in which P^* will cheat the verifier successfully on a false statement x. Note that the event that this guess is correct happens with probability $1/s$, so from now on, we assume the guess is always right without loss of generality), then B registers $pk = (f, y_0, y_1)$ as the public key, where $y_b = f(\alpha)$, $y_{1-b} = y$. For convenience we let $x_b = \alpha$, and denote by x_{1-b} one of preimages of y_{1-b} ($y_{1-b} = y = f(x_{1-b})$). Our goal is to find one preimage of y_{1-b}.

We write B as $\mathsf{B} = (\mathsf{B}_{real}, \mathsf{B}_j)$. B interacts with P^* as honest verifier (note that B knows the secret key α corresponding the public key pk) for all but jth session. Specifically, B employs the following extraction strategy:

1. B acts as the honest verifier in this stage. That is, it completes Π_v using $\alpha = x_b$ as secret key, and commits to e, $c_e = \mathsf{Com}_0(e, r_e)$ in phase 1 then runs resettably sound ZK argument in Phase 2 using e, r_e as the witness. In particular, B uses B_j to play the role of verifier in the jth session, and uses B_{real} to play the role of verifier in all other sessions. At the end of jth session, if B gets an accepting transcript (a, e, z) of Π_p, it enters the following rewinding stage; otherwise, B halts and output "$\perp$".
2. B_j rewind P^* to the point of beginning of step 3 in Phase 2 in jth session, it chooses a random string $e' \neq e$ and simulates the underlying resettably sound ZK argument in the same way showed in section 3: it commits to the hash value of the joint residual code of P^* and B_{real} in the second round of the resettably sound ZK argument (note this subprotocol is transformed from Barak's protocol) and uses them as the witness to complete the proof for the following *false* statement: $\exists\ r_e$ s.t. $c_e = \mathsf{Com}_0(e', r_e)$. If this rewinds incurs some other rewinds on other sessions, B_{real} always acts as an honest verifier on those sessions. When B get another accepting transcript (a, e', z') of Π_p at step 5 in Phase 2 in jth session, it halts, computes the witness from the two transcripts and outputs it, otherwise, B plays step 3 in jth session again.

We denote this extraction with EXTRA.

We first note that B's simulation of P^*'s view only differs from P^*'s view in real interaction with an honest verifier in the following: In the second run of Π_p in jth session B proves a *false* statement to P^* via the resettably sound zero knowledge argument instead of executing this sub-protocol honestly. We will show that this difference is computationally indistinguishable by P^*, otherwise we can use P^* to violate the zero knowledge property of the underlying resettably sound zero knowledge argument or the computationally-hiding property of the commitment scheme Com_0. We also note that if the simulation is successful, B gets an accepting transcript of Π_p in stage 1 with probability negligibly close to p, and once B enters the rewinding stage (stage 2) it will obtain another accepting transcript in expected polynomial time because p is non-negligible. In another words, B can outputs a valid witness with probability negligibly close to p in the above extraction.

Now assume B outputs a valid witness w' such that $(x, y_0, y_1, c, w') \in R'$, furthermore, the witness w' must satisfy $w' = (w'', r_{w''})$ and $y_b = f(w'')$ or $y_{1-b} = f(w'')$ because $x \notin L$. If $y_{1-b} = f(w'')$, we break the one-way assumption of f (find the one preimage of y_{1-b}), otherwise(i.e., w'' satisfies $y_b = f(w'')$), we fails. Next we claim B outputs $w' = (w'', r_{w''})$ such that $y_{1-b} = f(w'')$ with non-negligible probability.

Assume otherwise, with at most a *negligible* probability q, B outputs one preimage of y_{1-b}. Then We can construct a non-uniform algorithm B' (incorporating the code of P^*)to break the witness indistinguishability of Π_v or the computational binding of the commitment scheme Com_1.

The non-uniform algorithm B' takes as auxiliary input (y_0, y_1, x_0, x_1) (with input both secret keys) and interacts with P^* under the public key (y_0, y_1). It performs the following experiment:

1. *Simulation* (*until* B' *receives the first message* a *of* Π_p *in* j*th session*). B' acts exactly as the B. Without loss of generality, let B' uses x_0 as witness in all executions of Π_v that completed before step 2 in Phase 2 of the jth session. Once B' receives the first message a of Π_p in jth session, it splits this experiment and continues independently in following games:
2. *Extracting Game 0*. B' continues the above simulation and uses the same extraction strategy of B. In particular, it runs as follows. 1) continuing to simulate: B' uses x_0 as witness in all executions of Π_v that take place during this game; 2) extracting: if B' obtained an accepting transcript (a, e_0, z_0) at the end of the first run of Π_p in jth session, it rewinds to the point of beginning of step 3 in Phase 2 in jth session and replays this round by sending another random challenge $e' \neq e$ until he gets another accepting transcript (a, e_0', z_0') of Π_p, and then B' outputs a valid witness, otherwise outputs "$\perp$".
3. *Extracting Game 1*: B' repeats Extracting Game 0 but B' uses x_1 as witness in all executions of Π_v during this game (i.e., those executions of Π_v completed after the step 2 in Phase 2 in the jth session). At the end of this game, B' either obtains two accepting transcripts (a, e_1, z_1), (a, e_1', z_1') and outputs an valid witness, or outputs "$\perp$". Note that an execution of Π_v that takes place during this game means at least the last (third) message of Π_v in that execution has not yet been sent before step 2 in Phase 2 in jth session. Since the Π_v is *partial-witness-independent* (so we can decide to use which witness at the last (third) step of Π_v), B' can choose witness at its desire to complete that execution of Π_v after the step 2 in Phase 2 in the jth session.

We denote by EXP_0 the *Simulation* in stage 1 described above with its first continuation *Extracting Game 0*, similarly, denote by EXP_1 the same *Simulation* with its second continuation *Extracting Game 1*.

Note that the P^*'s view in EXP_0 is identical to its view in EXTRA in which B uses x_0 $(b = 0)$as witness in all executions of Π_v, so the outputs of B' at the end of EXP_0 is identical to the outputs of B taking x_0 as the secret key in EXTRA, that is, with non-negligible probability p B' outputs one preimage of y_0, and with negligible probability q it outputs one preimage of y_1.

Consider B's behavior in EXTRA when it uses $x_1 (b = 1)$as the secret key. The behavior of B only differs from the behavior of B' in EXP_1 in those executions of Π_v that completed before the step 2 in Phase 2 in the jth session: B' uses x_0 as witness in all those executions, while B uses x_1 as witness. However, the P^* cannot tell these apart because Π_v is a witness indistinguishable and all those executions of Π_v have not been rewound during both EXTRA and EXP_1 (note that B' does not rewind past the step 2 in Phase 2 in the jth session in the whole experiment). Thus, we can claim that at the end of EXP_1, B' outputs one preimage of y_1 with probability negligibly close to p, and it outputs one preimage of y_0 with probability negligibly close to q.

In the above experiment conducted by B', the first message a sent by P^* in the jth session contains a commitment c and this message a (therefore c) remains unchanged during the above whole experiment. Clearly, with probability negligibly close to p^2 (note that q is negligible), B' will output two valid witness $w_0' = (w_0'', r_{w_0''})$ and $w_1' = (w_1'', r_{w_1''})$ (note that $w_0'' \neq w_1''$ except for a very small probability) from the above two games such that the following holds: $y_0 = f(w_0'')$, $y_1 = f(w_1'')$, $c = \mathsf{Com}_1(w_0'', r_{w_0''})$ and $c = \mathsf{Com}_1(w_1'', r_{w_1''})$. This contradicts the computational-binding property of the scheme Com_1.

In sum, we proved that if Com_1 enjoys computational-binding and Π_v is witness indistinguishable protocol with *partial-witness-independence* property, then B succeeds in breaking the one-wayness of f. In another words, if the one-way assumption on f holds, it is infeasible for P^* to cheat an honest verifier in concurrent settings with non-negligible probability. □

Acknowledgements. Yi Deng thanks Giovanni Di Crescenzo, Rafael Pass, Ivan Visconti, Moti Yung, and Yunlei Zhao for many helpful discussions and clarifications. We thank anonymous reviewers for their encouragement and valuable comments.

References

[1] Barak, B.: How to go beyond the black-box simulation barrier. In: Proc. of IEEE FOCS 2001, pp. 106–115 (2001)
[2] Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resettably sound Zero Knowledge and its Applications. In: Proc. of IEEE FOCS 2001, pp. 116–125 (2001)
[3] Barak, B., Goldreich, O.: Universal Arguments and Their Applications. In: Proc. of IEEE CCC 2002, pp. 194–203 (2002)
[4] Blum, M.: How to Prove a Theorem so No One Else can Claim It. In: Proc. of ICM 1986, pp. 1444–1451 (1986)
[5] Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable Zero Knowledge. In: Proc. of ACM STOC (2000)
[6] Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Concurrent Zero-Knowledge requires $\Omega(logn)$ rounds. In: Proc. of ACM STOC 2001, pp. 570–579 (2001)
[7] Damgaard, I.: Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 174–187. Springer, Heidelberg (2000)
[8] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
[9] De Santis, A., Di Crescenzo, G., Persiano, G., Yung, M.: On Monotone Formaula Closure of SZK. In: Proc. of IEEE FOCS (1994)

[10] Di Crescenzo, G., Ostrovsky, R.: On Concurrent Zero Knowledge with Preprocessing. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 485–502. Springer, Heidelberg (1999)
[11] Di Crescenzo, G., Persiano, G., Visconti, I.: Constant Round Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 237–253. Springer, Heidelberg (2004)
[12] Di Crescenzo, G., Visconti, I.: Concurrent Zero Knowledge in the Public-Key Model. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 816–827. Springer, Heidelberg (2005)
[13] Dwork, C., Naor, M., Sahai, A.: Concurrent Zero-Knowledge. In: Proc. of ACM STOC 1998, pp. 409–418 (1998)
[14] Feige, U., Shamir, A.: Zero Knowledge Proof of Knowledge in Two Rounds. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 526–545. Springer, Heidelberg (1990)
[15] Feige, U., Shamir, A.: Witness Indistinguishability and Witness Hiding Protocols. In: Proc. of ACM STOC 1990, pp. 416–426 (1990)
[16] Goldreich, O.: Foundation of Cryptography-Basic Tools. Cambridge University Press, Cambridge (2001)
[17] Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or All languages in NP have zero-knowledge proof systems. J. ACM 38(3), 691–729 (1991)
[18] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM. J. Computing 18(1), 186–208 (1989)
[19] Hastad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A Pseudorandom Generator from Any One-Way Functions. SIAM Journal on Computing 28(4), 1364–1396 (1999)
[20] Micali, S., Reyzin, L.: Soundness in the Public-Key Model. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 542–565. Springer, Heidelberg (2001)
[21] Naor, M.: Bit Commitment using Pseudorandomness. Journal of Cryptology 4(2), 151–158 (1991)
[22] Naor, M., Ostrovsky, R., Venkatesan, R., Yung, M.: Perfect Zero-Knowledge Arguments for NP Using Any One-Way Permutation. Journal 11(2), 87–108 (1998)
[23] Pedersen, T.P.: Non-Interactive and Information-Theoretical Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
[24] Pass, R., Rosen, A.: Concurrent Non-Malleable Commitments. In: Proc. of IEEE FOCS 2005, pp. 563–572 (2005)
[25] Zhao, Y.: Concurrent/Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model and its Applications. Cryptology ePrint Archive, Report 2003/265

Secure Two-Party Computation of Squared Euclidean Distances in the Presence of Malicious Adversaries

Marc Mouffron[1], Frederic Rousseau[1], and Huafei Zhu[2]

[1] EADS Secure Networks, France
{marc.mouffron,frederic.rousseau}@eads.com
[2] Institute for Infocomm Research, Singapore
huafei@i2r.a-star.edu.sg

Abstract. Squared Euclidean Distance metric that uses the same equation as the Euclidean distance metric, but does not take the square root (thus clustering with the Squared Euclidean Distance metric is faster than clustering with the regular Euclidean Distance) is an efficient tool for clustering databases. Since there appears to be no previous implementation of secure Squared Euclidean Distance protocols in the malicious model, this paper studies two-party computation of Squared Euclidean Distance protocols in the presence of malicious adversaries based on state-of-the art homomorphic cryptographic primitives without using Yao-style circuit. The security of our protocol is analyzed by comparing what an adversary can do in the a real protocol execution to what it can do in an ideal scenario. We show that the proposed scheme is provably secure against malicious adversary assuming that the underlying homomorphic commitment is statistically hiding and computationally binding and the homomorphic encryption scheme is semantically secure in the common reference string model.

Keywords: Secure two-party computation, Squared Euclidean Distance, Stand-alone and simulation-based model.

1 Introduction

Privacy-preserving clustering algorithms group similar databases populated at distributed locations to improve data qualities and enable accurate data analysis and thus provide fundamental security components for distributed data mining with privacy concerns. Squared Euclidean Distance metric that uses the same equation as the Euclidean Distance metric, but does not take the square root (thus clustering with the Squared Euclidean Distance metric is faster than clustering with the regular Euclidean Distance) is a common tool for clustering databases.

Squared Euclidean Distance for computing k-clustering in hybrid databases: let $D = \{d_1, d_2, \cdots, d_n\}$ be a database set consisting of n objects.

Each object d_i is described by the value of the l numerical attributes and partitioned into two disjoint subsets d_i^A and d_i^B. Let μ_j^A (resp. μ_j^B) for $1 \leq j \leq k$ denote Alice's (resp., Bob's) share of the j-th mean. The candidate cluster centers are given by $\mu_j^A + \mu_j^B$ for $1 \leq j \leq k$. For each object d_i, Alice and Bob securely compute the Squared Euclidean Distance between the object and each of the k cluster centers. The result of the distance calculation is learned as random shares between Alice and Bob. Using the random shares, Alice and Bob can further securely compute the closest cluster for each object in the database. The described procedure continuous until the specified termination criteria is satisfied. We refer to the reader [12], [14], [19] and [13] for more details.

General solution in the semi-honest model using Yao-style circuit: The theory of general secure multiparty computation shows that any two-party function can be computed securely [3,20,21] and thus two participants can calculate Squared Euclidean Distance even without help of any trusted third party in the semi-honest model. That is, for fixed security parameter k (from which secure oblivious transfers are derived), system parameters ι (from which a domain is defined), and l (from which a dimension is defined), we can design a circuit for computing of Squared Euclidean Distance protocols that uses $O(\iota^2 l)$ AND gates and $O(\iota^2 l)$ XOR gates using the formulation of secure two-party evaluation of Goldreich [10]. The communication complexity is $O(\iota^2 lk)$-bit (using $O(\iota^2 l)$ oblivious transfers). As a result, the cost of an implementation of Squared Euclidean Distance protocols from Yao-style circuit invokes one oblivious transfer for each wire of the circuit as described in [10] and [16]. Consequently, the number of cryptographic operations performed is proportional to the size of the circuit computing $f(x)$ which implies that the computation complexity and communication complexity of general secure multiparty computation may be prohibitively expensive even for relatively simple functions.

1.1 Efficient Computation of Squared Euclidean Distance in the Semi-honest Model

Secure computation of Squared Euclidean Distance protocols in the semi-honest model can be efficiently reduced that of shared-scalar-product protocols[1]. As a result, the existence of privacy-preserving shared-scalar-product protocols in the semi-honest model implies the existence of privacy-preserving Squared Euclidean Distance protocols in the semi-honest model.

Squared Euclidean Distance protocols in the semi-honest model provide weak security guarantees. Regarding malicious adversaries, it has been shown that under suitable cryptographic assumptions, any multi-party function can be securely computed [11] and [10]. However this methodology (Yao-style circuit +

[1] Informally, an shared-scalar-product protocol states the following thing: there are two participants Alice who holds her input vector $(x_1, \cdots, x_l)$ and Bob who holds his input vector $(y_1, \cdots, y_l)$. They wish to compute random shares s_A and s_B of the scalar-product protocol such that $\sum_{i=1}^{l} x_i y_i = s_A + s_B$. At the end of the protocol, Alice holds the value s_A while Bob holds s_B.

compiler) comes at a price. These feasible results of secure computation typically do not yield protocols that are efficient enough to actually be implemented and used in practice. Their importance is more in telling us that it is perhaps worthwhile searching for other efficient solutions, because at least we know that a solution exists in principle. Since there appears to be no previous implementation of secure Squared Euclidean Distance protocols in the malicious model without using Yao-style circuit, we thus provide the following

Research problem: *how to securely implement two-party computation of Squared Euclidean Distance protocols in the presence of malicious adversaries without using Yao-style circuit?*

1.2 This Work

This paper studies secure two-party computation of Squared Euclidean Distance protocols in the presence of malicious adversaries without using Yao-style circuit. Informally, a two-party computation of Squared Euclidean Distance protocol states the following thing: Alice who holds an input vector $inp_A = (x_1, \cdots, x_l)$ and Bob who holds an input vector $inp_B = (y_1, \cdots, y_l)$, wish to compute $\sum_{i=1}^{l}(x_i - y_i)^2$, where $x_i \in I$ and $y_i \in I$ $(1 \leq i \leq l)$ and I is a prescribed interval, say, $I = \{0,1\}^{\iota}$. The output of Alice and Bob is Squared Euclidean Distance of two input vectors (By Δ: $= \sum_{i=1}^{l}(x_i - y_i)^2$, we denote Squared Euclidean Distance of two input vectors inp_A and inp_B).

Our security definition is standard (i.e., the standard definition for secure multi-party computation, we refer to the reader [10] for more detail). That is, the security of our protocol is analyzed by comparing what an adversary can do in the a real protocol execution to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an ideal computation involving an incorruptible trusted third party to whom the parties send their inputs. This trusted third party computes the functionality on the inputs and returns to each party its respective output. We will show that the two-party protocol for computing Squared Euclidean Distance presented in this paper is provably secure in the malicious model assuming that the underlying commitment is statistically hiding and computationally binding and the homomorphic encryption scheme is semantically secure in the common reference string model.

The computation complexity of our implementation is bounded by $O(l)$ encryptions (here we simply assume that the cost of a commitment is same as that of an encryption) while the communication complexity is bounded by $O(l(k + |I| + s))$, where k is a security parameter of Paillier's encryption scheme, $|I|$ is a system parameter and s is a security parameter of Fujisaki-Okamoto commitment scheme and thus our implementation is more efficient than the general implementation using Yao-style circuit where the communication complexity is $O(|I|^2 lk)$ (see Section 4.3 for more details).

Road map: The rest of this paper is organized as follows: In Section 2, syntax, functionality and security definition of two-party protocols for computing

Squared Euclidean Distances are introduced and formalized. In Section 3, building blocks that will be used to implement the primitive are briefly sketched. In Section 4, an implementation that does not emulate the circuit for computing Squared Euclidean Distance in the malicious model is proposed. We are able to show that the protocol is provably secure against malicious adversary assuming that the underlying homomorphic commitment is statistically hiding and computationally binding and the underlying homomorphic encryption scheme is semantically secure. We conclude our work in Section 5.

2 Syntax, Functionality and Definition of Security

2.1 Syntax

A protocol π for computing Squared Euclidean Distance involves the following two probabilistic polynomial time (PPT) Turing machines Alice and Bob.

- On input l and I (a pre-described interval, e.g., $I = \{0,1\}^{\iota}$), Alice generates an input vector $inp_A = (x_1, x_2, \cdots, x_l)$, $x_i \in I$ $(1 \le i \le l)$;
- On input l and I, Bob generates an input vector $inp_B = (y_1, y_2, \cdots, y_l)$, $y_i \in I$ $(1 \le i \le l)$;
- On input inp_A and inp_B, Alice and Bob jointly compute squared Euclidean distance (over the integer domain Z)

$$\Delta = \sum_{i=1}^{l} (x_i - y_i)^2$$

- The output of Alice is Δ; The output of Bob is Δ.

2.2 Functionality

By $\mathcal{F}_{SED}$, we denote the functionality of <u>S</u>quared <u>E</u>uclidean <u>D</u>istances. For given system parameters l and I, the functionality $\mathcal{F}_{SED}$ can be abstracted as follows:

- Alice has her input vector $inp_A = (x_1, \cdots, x_l)$; Bob has his input vector $inp_B = (y_1, \cdots, y_l)$; Each participant sends the corresponding input vector to $\mathcal{F}_{SED}$ − an imaginary trusted third party in the ideal world via a secure and private channel.
- Upon receiving inp_A and inp_B, $\mathcal{F}_{SED}$ checks whether $x_i \in I$ and $y_i \in I$ $(1 \le i \le l)$. If the conditions are satisfied, then $\mathcal{F}_{SED}$ computes $\Delta = \sum_{i=1}^{l} (x_i - y_i)^2$;
  If there is an invalid input $x_i \in inp_A$ but $x_i \notin I$, then $\mathcal{F}_{SED}$ chooses a random string $r_i \in_r I$ and sets $x_i \leftarrow r_i$; Similarly, if there is an invalid input $y_i \in inp_B$ but $y_i \notin I$, then $\mathcal{F}_{SED}$ chooses a random string $r_i \in_r I$ and sets $y_i \leftarrow r_i$; Once the valid input vectors inp_A and inp_B are generated, $\mathcal{F}_{SED}$ computes $\Delta = \sum_{i=1}^{l} (x_i - y_i)^2$.

- If $inp_A = \phi$, then $\mathcal{F}_{SED}$ chooses $x_1 \in_r I, \cdots, x_l \in_r I$ uniformly at random and sets $inp_A = \{x_1, \cdots, x_l\}$; Similarly, if $inp_B = \phi$, then $\mathcal{F}_{SED}$ chooses $y_1 \in_r I, \cdots, y_l \in_r I$ uniformly at random and sets $inp_B = \{y_1, \cdots, y_l\}$. Once two input vectors inp_A and inp_B are generated, $\mathcal{F}_{SED}$ computes $\Delta = \sum_{i=1}^{l}(x_i - y_i)^2$.
- The output of Alice is Δ. The output of Bob is Δ.

2.3 Security Definition

In this section, we briefly sketch the standard definition for secure two-party computation and refer to the reader [10] and [15] for more details.

Execution in the ideal model: Let $f=(f_1, f_2)$ be a two-party functionality, and P_1 and P_2 be two parties. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial time machine, and let $\mathcal{C} \subseteq \{1, 2\}$ be the indices of a corrupted party, controlled by an adversary $\mathcal{A}$. An ideal execution of proceeds as follows:

- Inputs: each party obtains an input(the i^{th} party's input is denoted by x_i, $i \in \{1, 2\}$). The adversary $\mathcal{A}$ receives an auxiliary input denoted by σ.
- Send inputs to the trusted party: any honest party P_i sends its received input x_i to the trusted third party. The corrupted party controlled by $\mathcal{A}$ may either abort, send its received input, or some other input to the trusted party. This decision is made by $\mathcal{A}$ and may depend on the value x_j for $j \in \mathcal{C}$ and its auxiliary input σ. Denote the vector of inputs sent to the trusted party by $\overline{\omega}=(\omega_1, \omega_2)$. Notice that $\overline{\omega}$ does not necessary equal $\overline{x}=(x_1, x_2)$;
  If the trusted party does not receive valid messages, then it replies two party with a special symbol $\perp$ and the ideal execution terminates. Otherwise, the execution proceeds to the next step.
- The trusted party sends output to $\mathcal{A}$: the trusted party computes $f(\overline{\omega})$ and sends $f_i(\overline{\omega})$ to the party P_i, for $i \in \mathcal{C}$;
- $\mathcal{A}$ instructs the trusted third party to continue or halt: $\mathcal{A}$ sends either continue or halt to the trusted party. If it sends continue, the trusted party sends $f_j(\overline{\omega})$ to P_j, for $j \notin \mathcal{C}$. Otherwise, if it sends halt, the trusted party sends $\perp$ to P_j, for $j \notin \mathcal{C}$.
- Outputs: an honest party always outputs the message it obtained from the trusted party. The corrupted party outputs nothing. The adversary $\mathcal{A}$ outputs any arbitrary function of the initial input $\{x_i\}$ and the message $f_i(\overline{\omega})$, $i \in \mathcal{C}$ obtained from the trusted party.

The ideal execution of f on inputs $\overline{x}$, auxiliary input σ to $\mathcal{A}$ and security parameter k, denoted by $\mathrm{IDEAL}_{f,\mathcal{A}(\sigma),\mathcal{C}}(\overline{x}, k)$ is defined as the output vector of the honest party and the adversary $\mathcal{A}$ from the above ideal execution.

Execution in the real model: The adversary sends all messages in place of the corrupted parity, and may follow an arbitrary polynomial-time strategy. In contrast, the honest party follows the instructions of the protocol π for computing f.

The real execution of π on input $\overline{x}$, auxiliary information σ to $\mathcal{A}$ and security parameter k, denoted by $\mathrm{REAL}_{\pi,\mathcal{A}(\sigma),\mathcal{C}}(\overline{x},k)$ is defined as the output vector of the honest party and the adversary $\mathcal{A}$ from the real execution of π.

Definition 1. *A protocol π is said to securely compute f with abort in the presence of malicious adversaries if for every non-uniform probabilistic polynomial time machine $\mathcal{A}$ for the real model, there exists a non-uniform probabilistic polynomial time machine $sim_{\mathcal{A}}$ for the ideal model, such that for every $\mathcal{C} \subseteq \{1,2\}$, every balanced vector $\overline{x}$ (for every i and j, $|x_i| = x_j$), and every auxiliary input $z \in \{0,1\}^*$:*

$$\{IDEAL_{f,\mathcal{A}(\sigma),\mathcal{C}}(\overline{x},k)\}_{k\in N} \approx \{REAL_{\pi,\mathcal{A}(\sigma),\mathcal{C}}(\overline{x},k)\}_{k\in N}$$

where $\approx$ indicates computational indistinguishability.

3 Building Blocks

Our implementation of two-party computation of Squared Euclidean Distance protocols in the presence of malicious adversaries will use the following state-of-the-art cryptographic primitives:

Paillier's public key encryption scheme: Paillier investigated a novel computational problem called the composite residuosity class problem (CRS), and its applications to public key cryptography in [17]. Our implementation of Squared Euclidean Distance protocols heavily relies on Paillier's public key encryption scheme which is sketched below:

The public key is a k_1-bit RSA modulus $n = pq$, where p, q are two large safe primes. The plain-text space is Z_n and the cipher-text space is $Z^*_{n^2}$. To encrypt $a \in Z_n$, one chooses $r \in Z^*_n$ uniformly at random and computes the cipher-text as $E_{PK}(a,r) = g^a r^n \bmod n^2$, where $g = (1+n)$ has order n in $Z^*_{n^2}$. The private key is (p,q). It is straightforward to verify that given $c = (1+n)^a r^n \bmod n^2$, and trapdoor information (p,q), one can first computes $c_1 := c \bmod n$, and then compute r from the equation $r = c_1^{n^{-1} \bmod \phi(n)} \bmod n$; Finally, one can compute a from the equation $cr^{-n} \bmod n^2 = 1 + an$. The encryption function is homomorphic, i.e., $E_{PK}(a_1,r_1) \times E_{PK}(a_2,r_2) \bmod n^2 = E_{PK}(a_1 + a_2 \bmod n, r_1 \times r_2 \bmod n)$.

Fujisaki-Okamoto commitment scheme: Let s and k_2 be security parameters. The public key is a k_2-bit RSA modulus, where P, Q are two large safe primes. We assume that neither the committer C nor the receiver R knows factorization N. Let g_1 be a generator of QR_N and g_2 be an element of large order of the group generated by g_1 such that both discrete logarithm of g_1 in base g_2 and the discrete logarithm of g_2 in base g_1 are unknown by C and R. We denote $C(a,r_a) = g_1^a g_2^{r_a} \bmod N$ a commitment to a in base (g_1,g_2), where r_a is randomly selected over $\{0, 2^s N\}$. This commitment scheme first appeared in [9] and reconsidered by Damgård and Fujisaki [5] is statistically secure commitment scheme, i.e.,

- C is unable to commit itself to two values a_1, a_2 such that $a_1 \neq a_2$ in Z by the same commitment unless R can factor N or solves the discrete logarithm of g_1 in base g_2 or the the discrete logarithm of g_2 in base g_1;
- $C(a, r_a)$ statistically reveals no information to R, *i.e.*, there is a simulator which outputs simulated commitments to a which are statistically indistinguishable from true ones.

Notice that this commitment is homomorphic, i.e., $C(a+b, r_a+r_b) = C(a, r_a) \times C(b, r_b)$. This property is useful when R wants to prove that the committed value $a \in [x, y]$.

Boudot's protocol: With the help of Fujisaki-Okamoto commitment scheme, an efficient protocol allows Alice to prove to Bob that a committed number x belongs to the desired interval $[a, b]$ ($0 < a \in Z$ and $a < b \in Z$), has been proposed by Boudot [2]. The idea behind Boudot's protocol is that to achieve a proof of membership without tolerance, the size of x is first enlarged, and then Alice proves to Bob that the value $2^T x$ lies in the interval $< 2^T a - 2^T, 2^T b + 2^T >$ with tolerance (a proof with tolerance is easier than a proof without tolerance, we refer the reader to [2] for further reference), and thus $x \in [a, b]$. Boudot's proof technique is only asymptotically sound. When it is employed the concrete parameter setting must be discussed to show that a certain degree of soundness is obtained. Since Boudot's protocol is implemented based on the notion of Fujisaki-Okamoto commitment scheme, we here simply assume that the security parameters are inherently adopted from Fujisaki-Okamoto commitment scheme.

Proof of knowledge of encryptions: Given a cipher-text Enc(x) which is computed from Paillier's encryption scheme, a prover (say, Alice) wants to prove that she knows x and x lies in a given interval I to Bob. There is efficient protocol presented by Damgård and Jurik already in [7]. The basic idea is the following: given a=Enc(x, r_e), the prover provides a commitment b=Com(x, r_c) which is computed from Fujisaki-Okamoto commitment scheme, proves that the commitment contains the same number as the encryption, and then uses Boudot's protocol [2] to prove that $x \in I$. More precisely,

- let T be the maximum bit length of x. The prover chooses at random u, an integer of length $T + 2k$, where k is a security parameter. She sends $\overline{a}$=Enc(u, r'_e), $\overline{b}$=Com(u, r'_c) to the verifier;
- the verifier chooses a l-bit challenge f;
- the prover opens the encryption $a' = a\,\overline{a}^f \bmod n^2$ and the commitment $b' = b\overline{b}^f \bmod N$, to reveal in both cases the number $z = u + fx$, $R_e = r'_e + fr_e$ and $R_c = r'_c + fr_c$. The verifier checks that the openings were correct.

4 Implementation and Proof of Security

4.1 Our Implementation

Our implementation of two-party computation of Squared Euclidean Distance protocols in the presence of malicious adversaries consists of the following two stages: setup stage and secure computation stage. In the setup stage, a common reference string σ and public key encryption schemes for two participants are generated.

Description of Setup Stage

– A common reference string σ is derived from Okamoto-Fujisaki's commitment scheme (this commitment scheme first appeared in [9] and reconsidered by Damgård and Fujisaki in [5]. We stress that the use of Okamoto-Fujisaki's commitment scheme is not essential, it can be replaced by any homomorphic commitment with similar cryptographic properties. For simplicity, we will make use of this commitment throughout the paper). In our protocol, a common reference string σ is generated by an oracle and the resulting string σ is used by both participants throughout the protocol. On input security parameters k_C and s, an oracle generates the public key $N = PQ$, where P, Q are large safe prime numbers (i.e., P=$2P'$+1, Q=$2Q'$+1, P, Q, P' and Q' are large prime numbers) such that neither Alice nor Bob knows factorization N. The oracle also generates two random generator $g_1 \in_r QR_N$ and $g_2 \in_r QR_N$ such that the discrete logarithm of g_1 in base g_2 and the discrete logarithm of g_2 in base g_1 are unknown by Alice and Bob. By $C(a, r_a) = g_1^a g_2^{r_a} \bmod N$, we denote a commitment to x in the bases g_1 and g_2, where r_a is randomly selected over $\{0, 2^s N\}$, s is another security parameter. By (g_1, g_2, s, N), we denote the common reference string σ.
– On input a security parameter k_A, a k_A-bit RSA modulus $n_A = p_A q_A$ for Paillier's encryption scheme E_A is generated (by Alice) [17], where p_A, q_A are two large safe primes. The plain-text space is Z_{n_A} and the cipher-text space is $Z^*_{n_A{}^2}$. The public key of Alice pk_A is $(g_A,\, n_A)$ where g_A=$(1 + n_A)$ has order n_A in $Z^*_{n_A{}^2}$. The private key sk_A is (p_A, q_A).
– Similarly, on input a security parameter k_B, a k_B-bit RSA modulus $n_B = p_B q_B$ for Paillier is generated (by Bob), where p_B, q_B are two large safe prime numbers. The plain-text space is Z_{n_B} and the cipher-text space is $Z^*_{n_B{}^2}$. The public key of Bob pk_B is $(g_B,\, n_B)$ where g_B=$(1 + n_B)$ has order n_B in $Z^*_{n_B{}^2}$. The private key sk_B is (p_B, q_B).

We stress that the size of each public key of Paillier's encryption scheme is sufficiently large so that all computations can be performed over the integer domain. This is possible since the extension of Damgård and Jurik [6] can achieve this requirement. We thus simply assume that the public key size of the Paillier's encryption is sufficiently large throughout the paper.

High level description of our secure computation stage. For convenience the reader, we briefly describe what achieves in each of the following steps specified in the secure computation stage. Without loss of generality, we assume that Alice talks first in the following computations. At Step 1, Alice commits her input vector inp_A =$(x_1, \cdots, x_l)$, and then sends the encryption vectors $E_A(inp_A)$ (=$(E_A(x_1), \cdots, E_A(x_l))$) to Bob, finally, Alice proves to Bob that the i-th commitment and the i-th encryption hide the same value x_i which lies in the correct interval I; At Step 2, Bob commits his input vector inp_B=$(y_1, \cdots, y_l)$, and then Alice and Bob jointly compute random shares (s_A, s_B) of the scalar product computed from their input vectors such that $s_A + s_B = \Sigma_{i=1}^{l} x_i y_i$. At Step 3, Alice proves to Bob that the value $E_B(\alpha_A - 2s_A)$ is correctly computed, where $\alpha_A = \Sigma_{i=1}^{l} x_i^2$; Similarly, at Step 4 and Step 5, Bob proves to Alice that the value $E_A(\alpha_B - 2s_B)$ is correctly computed, where $\alpha_B = \Sigma_{i=1}^{l} y_i^2$; Finally, the Squared Euclidean Distance is computed from the Step 6.

Detail description of our secure computation stage. We now can describe the detailed implementation below:

Step 1: Alice commits and encrypts her input vector and proves consistency of the encrypted values

- Step 1.1: computes the commitments of her input vector $\{c_{A,i}\}$=$\{C(x_i, r_{x_i})\}$ $(1 \leq i \leq l)$, where the Okamoto-Fujisaki's commitment C is indexed by the common reference string σ;
- Step 1.2: computes the encryptions of her input vector $\{e_{A,i}\}$=$\{E_A(x_i)\}$ $(1 \leq i \leq l)$, where the Paillier's encryption scheme E_A is indexed by the public key pk_A;
- Step 1.3: sends $(c_{A,i}$, $e_{A,i})$ to Bob $(1 \leq i \leq l)$;
- Step 1.4: proves to Bob that the commitment $c_{A,i}$ and the encryption $e_{A,i}$ hide the same value x_i and $x_i \in I$ $(1 \leq i \leq l)$ by performing the following computations [7]:
  The prover Alice chooses $u_i \in_r \{0,1\}^{|I|+k_1}$, where k_1 is a security parameter, e.g., k_1 =160-bit. and then sends $E_A(u_i)$, $C(u_i)$ to the verifier Bob;
  The verifier chooses a k_1-bit challenge f_i;
  The prover opens the encryption $E_A(u_i){e_{A,i}}^{f_i}$ and the commitment $C(u_i)c_{A,i}^{f_i}$, to reveal in both cases the number $z_i = u_i + x_i f_i$. The verifier checks the correctness of the opening.
  The prover further proves to Bob that x_i lies in the correct interval I by running Boudot's protocol [2].

Step 2: Bob commits and encrypts his input vector and proves consistency of the encrypted values

- Step 2.1: verifies the correctness of the proofs; If both verifications are correct, then Bob continues the following computations, otherwise, Bob stops the computation and outputs $\perp$;
- Step 2.2: computes the commitments of his input vector $\{c_{B,i}\}$=$\{C(y_i, r_{y_i})\}$ and proves to Alice that he knows how to open the commitment $c_{B,i}$ and $y_i \in I (1 \leq i \leq l)$;
- Step 2.3: chooses a random string $s_B \in \{0,1\}^{2|I|+\lceil \log l \rceil}$, computes the commitment of $c(s_B) = C(s_B, r_{s_B})$, and proves Alice he knows how to open the commitment $c(s_B)$ and $s_B \in \{0,1\}^{2|I|+\lceil \log l \rceil}$;
- Step 2.4: computes $\prod_{i=1}^{l} E_A(x_i)^{y_i} E_A(1)^{s_B}$;

– Step 2.5: proves to Alice that each exponent of $\prod_{i=1}^{l} E_A(x_i)^{y_i} E_A(1)^{s_B}$ to the base of $E_A(x_i)$ $(1 \leq i \leq l)$ is equal to the hidden value of the commitment $c_{B,i}$ and the exponent to the base $E_A(1)$ $(1 \leq i \leq l)$ is equal to the hidden value of $c(s_B)$ by performing the following computations (the technique described below can be viewed as an extended version of Damgård and Jurik [7], and Cramer and Damgård [4]).

  Bob chooses $\hat{s_B} \in \{0,1\}^{2|I|+\lceil \log l \rceil + k_1}$, $\hat{y_1} \in_r \{0,1\}^{|I|+k_1}$, $\cdots$, $\hat{y_l} \in_r \{0,1\}^{|I|+k_1}$ uniformly at random, and then computes $C(\hat{s_B}, r_{\hat{s_B}})$, $C(\hat{y_1}, r_{\hat{y_1}})$, $\cdots$, $C(\hat{y_l}, r_{\hat{y_l}})$; $E_A(s_A) = E_A(1)^{s_B} E_A(x_1)^{y_1} \cdots E_A(x_l)^{y_l}$ and $E_A(\hat{s_A}) = E_A(1)^{\hat{s_B}}\ E_A(x_1)^{\hat{y_1}} \cdots E_A(x_l)^{\hat{y_l}}$.

  Finally, Bob sends $E_A(s_A)$, $E_A(\hat{s_A})$, together with the proof that he knows how to open the commitments $C(\hat{s_B})$, $C(\hat{y_i})$ and each $\hat{y_i}$ lies in the correct interval $\{0,1\}^{|I|+k_1}$ $(0 \leq i \leq l)$ and $\hat{s_B} \in \{0,1\}^{2|I|+\lceil \log l \rceil + k_1}$ to Alice.

  Alice verifies the correctness of the proof. If it is incorrect, then she stops the execution of the protocol; otherwise, she performs the following computations $s_A \leftarrow D_A(E_A(s_A))$, $\hat{s_A} \leftarrow D_A(E_A(\hat{s_A}))$.

  Alice then chooses a random string $f \in \{0,1\}^{k_1}$ uniformly at random and sends it to Bob; Alice and Bob then computes: $C(z_{s_B}) = C(s_B, r_{s_B})^f C(\hat{s_B}, r_{\hat{s_B}})$; $C(z_1) = C(y_1, r_{y_1})^f C(\hat{y_1}, r_{\hat{y_1}})$, $\cdots$, $C(z_l) = C(y_l, r_{y_l})^f C(\hat{y_l}, r_{\hat{y_l}})$;

  Bob opens $C(z_{s_B})$, $C(z_1)$, $\cdots$, $C(z_l)$ to Alice. Alice checks the correctness of all opening of the commitments, and also checks the validity of equation $\hat{s_A} + f s_A = z_{s_B} + x_1 z_1 + \cdots + x_l z_l$.

Step 3: Alice computes her random share for the scalar-product of two input vectors and proves to Bob her random share is correctly generated

– Step 3.1: if all proofs are correct, then Alice continues the following computations, otherwise, Alice stops the computation and outputs $\perp$;
– Step 3.2: computes the commitment $c(s_A) := C(s_A, r_{s_A})$;
– Step 3.3: Alice computes $c_{A,i^2} := C(x_i^2, r_{x_i^2})$ $(1 \leq i \leq l)$;
– Step 3.4: proves to Bob that the encrypted value of c_{A,i^2} is the square of the committed value of $c_{A,i}$ by performing the following computations [2,9]: for each $c_{A,i} = C(x_i, r_{x_i})$, Alice computes $r_i = r_{x_i} - x_i r_{x_i^2}$ (thus $c_{A,i^2} = c_{A,i}^{x_i} g_2^{r_i}$); Alice proves to Bob that c_{A,i^2} is a commitment to x_i in base $(c_{A,i}, g_2)$ (Alice has already proved to Bob that $c_{A,i}$ is a commitment to x_i in base (g_1, g_2) at the Step 1.4);
  If all proofs are correct, Bob continues the following joint computations with Alice; otherwise Bob stops and outputs $\perp$.
– Step 3.6: $\alpha_A \leftarrow \sum_{i=1}^{l} x_i^2$, and then computes the commitment $C(\alpha_A, r_{\alpha_A}) \leftarrow \prod_{i=1}^{l} c_{A,i^2}$;
– Step 3.7: proves to Bob $c(s_A)$ and $E_A(s_A)$ hide the same value;
– Step 3.8: computes $E_B(\alpha_A - 2s_A)$ and $C(\alpha_A - 2s_A) \leftarrow C(\alpha_A, r_{\alpha_A}) \times c(s_A)^{-2}$; and proves that $E_B(\alpha_A - 2s_A)$ and $C(\alpha_A - 2s_A)$ hide the same value.

Step 4: Bob computes his share of the scalar-product of two input vectors and thus obtains Δ

- Step 4.1: verifies the correctness of the proof; If it is correct, then Bob continues the following computations, otherwise, Bob stops the computation and outputs $\perp$;
- Step 4.2: $(\alpha_A - 2s_A) \leftarrow D_B(E_B(\alpha_A - 2s_A))$, and sets $\beta_A = (\alpha_A - 2s_A)$;

Step 5: Bob proves to Alice his commitments and encryptions are correctly generated that allows Alice to compute Δ

- Step 5.1: $\alpha_B \leftarrow \sum_{i=1}^{l} y_i^2$;
- Step 5.2: computes c_{B,i^2}: $=C(y_i, r_{y_i})$ $(1 \leq i \leq l)$, and proves to Alice that the committed value of c_{B,i^2} is the square of the committed value of $c_{B,i}$;
- Step 5.3: computes $C(\alpha_B, r_{\alpha_B}) \leftarrow \prod_{i=1}^{l} c_{B,i^2}$;
- Step 5.4: computes $E_A(\alpha_B + 2s_B)$ and $C(\alpha_B + 2s_B) \leftarrow C(\alpha_B, r_{\alpha_B}) \times c(s_B)^2$ (the proofs that Bob knows how to open $c(s_B)$ and s_B lies in the correct interval are presented at the Step 2.3);
- Step 5.5: proves to Alice that $E_A(\alpha_B + 2s_B)$ and $C(\alpha_B + 2s_B)$ hide the same value.

Step 6: Alice obtains Δ

- Step 6.1: verifies the correctness of the proof; If it is correct, then Alice continues the following computations, otherwise, Alice stops the computation and outputs $\perp$;
- Step 6.2: $(\alpha_B + 2s_B) \leftarrow D_A(E_A(\alpha_B + 2s_B))$, and sets $\beta_B = (\alpha_B + 2s_B)$;
- Step 6.3: outputs $\Delta \leftarrow \beta_A + \beta_B$;

This ends the description of our implementation.

4.2 The Proof of Security

Lemma 1. *For every non-uniform probabilistic polynomial time $\mathcal{A}$ for the real protocol which corrupts Alice, there exists a non-uniform probabilistic polynomial time adversary sim_A (a simulator that creates views for malicious Alice and honest Bob) such that the output vectors of the honest Bob and the adversary $\mathcal{A}$ from the real execution of the protocol defined above is computationally indistinguishable from the output of the honest Bob and the adversary sim_A from the ideal execution.*

Proof. The simulator sim_A is defined as follows: sim_A first generates a common reference string σ which will be used in the real world protocol. That is, on input security parameters k_C and s, sim_A generates the public key $N = PQ$, where P, Q are large safe prime numbers (i.e., $P{=}2P' + 1$, $Q{=}2Q' + 1$, P, Q, P' and Q' are large prime numbers). sim_A also produces a random generator $g_2 \in_r QR_N$, and then computes $g_1 \in QR_N$ from the equation $g_1 = g_2^{\lambda}$. Thus sim_B knows the discrete logarithm of g_1 in base g_2. The common reference string $\sigma = (g_1, g_2, s, N)$. The auxiliary information is (P, Q, P', Q', λ).

From the description of the real world protocol, we know what Alice proves to Bob are the following things:

– the knowledge of $x_i \in I$ $(1 \le i \le l)$ at the Step 1.4;
– the knowledge of x_i^2 $(1 \le i \le l)$ at the Step 3.4;
– the knowledge of s_A at the Step 3.7;
– and the knowledge of β_A $(=\alpha_A - 2s_A)$ at the Step 3.8.

sim_A extracts Alice's input vector inp_A=$(x_1, \cdots, x_l)$ $(x_i \in I,\ 1 \le i \le l)$ by rewinding malicious Alice at the Step 1.4. sim_A then sends inp_A to $\mathcal{F}_{SED}$ and obtains Δ from $\mathcal{F}_{SED}$. sim_A also extracts the knowledge of s_A by rewinding malicious Alice at the Step 3.7. Once sim_A obtains x_i $(1 \le i \le l)$ and s_A, it can simulate the proof of the knowledge of x_i^2 $(1 \le i \le l)$ at the Step 3.6; the proof of the knowledge of s_A at the Step 3.7; and the proof of the knowledge of β_A at the Step 3.8. By the binding property of the underlying commitment scheme, we know that the consistency of α_A and β_A is satisfied.

The rest work of sim_A is to simulate what Alice verifies in the real world protocol. sim_A now first computes $\beta_B \in \{0,1\}^{2|I|+\lceil \log l \rceil+1}$ from the equation Δ $= \beta_A + \beta_B$. sim_A then chooses $y_1 \in I, \cdots, y_l \in I$ uniformly at random such that $\frac{\alpha_B - \beta_B}{2}$ $=s_B$, where $\alpha_B = \sum_{i=1}^{l} y_i^2$ (if $\beta_B - \alpha_B$ is not an even number, then sim_A re-performs the above computations until $\alpha_B - \beta_B$ is an even number). We stress that $s_B \in \{0,1\}^{2|I|+\lceil \log l \rceil}$ can simulated trivially since sim_A knows the auxiliary information of the commitment scheme. For the fixed random string $f \in \{0,1\}^{k_1}$ and s_A, sim_A chooses $\hat{s_A} \in \{0,1\}^{2|I|+\lceil \log l \rceil+k_1+1}$, $z_1 \in \{0,1\}^{2|I|+k_1+1}$, $\cdots$, $z_l \in \{0,1\}^{2|I|+k_1+1}$ uniformly at random, and performs the following computations:

– computes $C(s_B, r_{s_B})$ and sets $C(s_B) \leftarrow C(s_B, r_{s_B})$;
– computes $C(y_1, r_{y_1})$, $\cdots$, $C(y_l, r_{y_l})$;
– computes $C(z_1, r_{z_1})$, $\cdots$, $C(z_l, r_{z_l})$, and sets $C(z_1) \leftarrow C(z_1, r_{z_1})$, $\cdots$, $C(z_l)$ $\leftarrow C(z_l, r_{z_l})$;
– computes z_{s_B} from the equation $\hat{s_A} + f s_A = z_{s_B} + x_1 z_1 + \cdots + x_l z_l$;
– computes $C(z_{s_B}, r_{z_{s_B}})$ and sets $C(z_{s_B}) \leftarrow C(z_{s_B}, r_{z_{s_B}})$;
– computes $C(\hat{s_B}, r_{\hat{s_B}})$, $C(\hat{y_1}, r_{\hat{y_1}})$, $\cdots$, $C(\hat{y_l}, r_{\hat{y_l}})$ from the following equations: $C(z_{s_B}) = C(s_B, r_{s_B})^f\ C(\hat{s_B}, r_{\hat{s_B}})$, $C(z_1) = C(y_1, r_{y_1})^f\ C(\hat{y_1}, r_{\hat{y_1}})$, $\cdots$, $C(z_l)$ $= C(y_l, r_{y_l})^f\ C(\hat{y_l}, r_{\hat{y_l}})$.

The requirement that $\hat{s_B} \in \{0,1\}^{2|I|+\lceil \log l \rceil+k_1}$ and $\hat{y_i} \in \{0,1\}^{|I|+k_1}$ $(0 \le i$ $\le l)$ can be achieved trivially since sim_A holds the auxiliary information of the underlying commitment scheme. It follows that the output vectors of the honest Bob and the adversary $\mathcal{A}$ from the real execution of the protocol defined above is computationally indistinguishable from the output of the honest Bob and the adversary sim_A from the ideal execution.

Lemma 2. *For every non-uniform probabilistic polynomial time $\mathcal{A}$ for the real protocol which corrupts Bob, there exists a non-uniform probabilistic polynomial time adversary sim_B (a simulator that creates views for honest Alice and malicious Bob) such that the output vectors of the honest Alice and the adversary $\mathcal{A}$ from the real execution of the protocol defined above is computationally indistinguishable from the output of the honest Alice and the adversary sim_B from the ideal execution.*

Proof. The simulator sim_B is defined as follows: sim_B first generates a common reference string σ which is used in the real world protocol. That is, on input security parameters k_C and s, sim_B generates the public key $N = PQ$, where P, Q are large safe prime numbers (i.e., $P{=}2P'+1$, $Q{=}2Q'+1$, P, Q, P' and Q' are large prime numbers). sim_B also produces a random generator $g_2 \in_r QR_N$, and then computes $g_1 \in QR_N$ from the equation $g_1 = g_2^{\lambda}$. Thus sim_B knows the discrete logarithm of g_1 in base g_2. The common reference string $\sigma = (g_1, g_2, s, N)$. The auxiliary information is (P, Q, P', Q', λ).

From the description of the real world protocol, we know what Bob proves to Alice are the following things:

– the knowledge of $y_i \in I$ $(1 \le i \le l)$ at the Step 2.2;
– the knowledge of s_B at the Step 2.3
– $E_A(s_A)$ $(= \prod_{i=1}^{l} E_A(x_i)^{y_i} E_A(1)^{s_B})$ is correctly calculated with respect to y_i $(1 \le i \le l)$ and s_B at the Step 2.5;
– the knowledge of α_B at the Step 5.3;
– and the knowledge of β_B $(=\alpha_B + 2s_B)$ at the Step 5.4.

sim_B obtains $y_i \in I$ $(1 \le i \le l)$ by rewinding Bob at the Step 2.2. sim_B then sends inp_B to $\mathcal{F}_{SED}$ and obtains Δ from $\mathcal{F}_{SED}$. sim_B also extracts the knowledge of s_B by rewinding Bob at the Step 2.3. Once given y_i, s_B and Δ, sim_B can compute $\alpha_B = \sum_{i=1}^{l} y_i^2$, $\beta_B = \alpha_B + 2s_B$. By the binding property of the underlying commitment scheme, we know that the consistency of α_B and β_B is satisfied. sim_B then computes β_A from the equation $\beta_A + \beta_B = \Delta$. sim_B further chooses $x_1 \in I, \cdots, x_l \in I$ uniformly at random, and then computes s_A from the equation $\alpha_A - 2s_A = \beta_A$ such that $s_A = \frac{\alpha_A - \beta_A}{2}$, where $\alpha_A = \sum_{i=1}^{l} x_i^2$. We stress that the simulation of s_A that lies in a correct interval $\{0,1\}^{2|I|+\lceil \log l \rceil+1}$ is trivial since sim_B has the auxiliary information of the commitment scheme.

For fixed f at the Step 2.5, sim_B chooses $z_1 \in \{0,1\}^{|I|+k_1+1}, \cdots, z_l \in \{0,1\}^{|I|+k_1+1}$ and $z_{s_B} \in \{0,1\}^{2|I|+\lceil \log l \rceil+k_1+1}$ uniformly at random and then computes $\hat{s_A}$ from the equation $\hat{s_A} + fs_A = z_{s_B} + x_1 z_1 + \cdots, x_l z_l$. Furthermore, sim_B computes $C(\hat{s_B}, r_{\hat{s_B}}) = C(z_{s_B})\ C(s_B, r_{s_B})^{-f}$; $C(\hat{y_1}, r_{\hat{y_1}}) = C(z_1)$ $C(y_1, r_{y_1})^{-f}$, $\cdots$, $C(\hat{y_l}, r_{\hat{y_l}}) = C(z_l)\ C(y_l, r_{y_l})^{-f}$; Again, since sim_B holds the auxiliary information of the commitment scheme, thus, the correctness of opening for each commitment computed above can be simulated trivially. By rewinding Bob, sim_B obtains two commitment vectors $(z_1, \cdots, z_l)$ and $(z_1', \cdots, z_l')$ such that $\hat{s_A} + fs_A = z_{s_B} + x_1 z_1 + \cdots + x_l z_l$, and $\hat{s_A} + f's_A = x_1 z_1' + \cdots + x_l z_l'$. As a result, we have the following equation $(f' - f)s_A = z_{s_B}' - z_{s_B} + x_1(z_1' - z_1) + \cdots + x_l(z_l' - z_l)$. By applying the binding property of the underlying commitment scheme, we know that $s_A = s_B + \sum_{i=1}^{l} x_i y_i$.

The rest work of sim_B is to simulate what Bob verifies in the real protocol. Again since sim_B has the auxiliary information of the commitment scheme, the view of Bob when he interacts with sim_B at the Step 1 and Step 3 can be simulated trivially. It follows that the output vectors of the honest Alice and the adversary $\mathcal{A}$ from the real execution of the protocol defined above is computationally indistinguishable from the output of the honest Alice and the adversary sim_B from the ideal execution.

By combining two lemmas above, we have the following main result immediately.

Theorem 1. *The two-party protocol for computing Squared Euclidean Distance described above is secure in the malicious model assuming that the underlying commitment is statistically hiding and computationally binding and the homomorphic encryption scheme is semantically secure in the common reference string model.*

4.3 Computation and Communication Complexity

We now consider the communication and communication complexity of our implementation of Squared Euclidean Distance protocols. At the first step, Alice computes l commitments and l encryptions, and then proves to Bob that the j^th commitment and the j^{th} encryption hide the same values. Finally, Alice proves to Bob each of committed value lies in I, which costs additional $O(l)$ commitments. Thus, the communication complexity is bounded by $O(lk_A) + O(lk_C) + l(k_1 + |I| + s)$, where s is a security parameter in the Fujisaki-Okamoto commitment scheme and k_1 is a security parameter (say 160 bit of challenge string) specified in our implementation. At the second step, Bob commits his input vector inp_B=$(y_1, \cdots, y_l)$, and then Alice and Bob jointly compute random shares (s_A, s_B) of the scalar product computed from their input vectors such that $s_A + s_B = \Sigma_{i=1}^l x_i y_i$. Thus, the computation complexity at this step is $O(l)$ and the communication complexity is bounded by $O(lk_A) + O(lk_B) + O(l(k_1 + |I| + s))$. The rest of computations of Alice and Bob are both bounded by $O(l)$, and thus,the total communication complexity of both parties are bounded by $O(lk_A)$ + $O(lk_B) + O(l(k_1 + |I| + s))$. If we further assume that $k_A \approx k_B \approx k_C = k$, and the cost of an encryption is approximate to that of a commitment, then the communication complexity is bounded by $O(lk_A) + O(l(|I| + s))$ =$O(l(k + |I| + s))$ which is superior to the circuit-based solution where the communication complexity is $O(|I|^2 lk)$.

5 Conclusion

We have proposed a new protocol for computing Squared Euclidean Distance that does not emulate the circuit of the function in the malicious model. We have shown that our implementation is secure against malicious adversary in the common reference string model assuming that the underlying commitment is statistically hiding and computationally binding and the homomorphic encryption scheme is semantically secure.

References

1. Barak, B., Lindell, Y.: Strict polynomial-time in simulation and extraction. In: STOC 2002, pp. 484–493 (2002)
2. Boudot, F.: Efficient Proofs that a Committed Number Lies in an Interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)

3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic faulttolerant distributed computation. In: Proc. 20th Annual ACM Symposium on Theory of Computing, pp. 1–10 (1988)
4. Cramer, R., Damgård, I.: Secret-Key Zero-Knowlegde and Non-interactive Verifiable Exponentiation. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 223–237. Springer, Heidelberg (2004)
5. Damgård, I., Fujisaki, E.: A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)
6. Damgård, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In: Public Key Cryptography 2001, pp. 119–136 (2001)
7. Damgård, I., Jurik, M.: Client/Server Tradeoffs for Online Elections. In: Proc. of Public Key Cryptography 2002, pp. 125–140. Springer, Heidelberg (2002)
8. Fagin, R., Naor, M., Winkler, P.: Comparing Information Without Leaking it. Communication of ACM 39, 77–85 (1996)
9. Fujisaki, E., Okamoto, T.: Statistically zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
10. Goldreich, O.: The Foundations of Cryptography, vol. 2. Cambridge University Press, Cambridge (2004)
11. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game: A Completeness Theorem for Protocols with Honest Majority. In: 19th STOC, pp. 218–229 (1987)
12. Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On Private Scalar Product Computation for Privacy-Preserving Data Mining. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 104–120. Springer, Heidelberg (2005)
13. Jagannathan, G., Wright, R.N.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: ACM-KDD 2005, pp. 593–599 (2005)
14. Kantarcioglu, M., Clifton, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned data. In: The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD 2002), June 2 (2002)
15. Lindell, Y.: Composition of Secure Multi-Party Protocols. LNCS, vol. 2815. Springer, Heidelberg (2003)
16. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: ACM Conference on Electronic Commerce 1999, pp. 129–139 (1999)
17. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
18. Pinkas, B.: Fair Secure Two-Party Computation. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 87–105. Springer, Heidelberg (2003)
19. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: The 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002), pp. 639–644 (2002)
20. Yao, A.C.: Protocols for Secure Computations. In: Proc. of the 23rd IEEE Symp. On Foundations of Computer Science, pp. 160–164 (1982)
21. Yao, A.C.: How to Generate and Exchange Secrets. In: Proc. of the 27th IEEE Symp. On Foundations of Computer Science, pp. 162–167 (1986)

A Discrete-Logarithm Based Non-interactive Non-malleable Commitment Scheme with an Online Knowledge Extractor[⋆]

Ning Ding and Dawu Gu

Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, 200240, China
{dingning,dwgu}@sjtu.edu.cn

Abstract. We propose a discrete-logarithm based non-interactive non-malleable commitment scheme with an online knowledge extractor in the random oracle and the public parameter model (need a third party to distribute public parameters to both sender and receiver in advance). Our scheme is statistically-secret computationally-binding. The fundamental technique we employ is the construction of non-interactive zero-knowledge proofs of knowledge with online knowledge extractors from Fiat-Shamir proofs of knowledge for relations with logarithmic challenge length presented by Fischlin in Crypto'05. Compared with previous works, our scheme is practical and the online knowledge extractor is strictly polynomial-time.

1 Introduction

Commitment schemes are a basic ingredient in many cryptographic protocols. Very informally, a commitment scheme involves two probabilistic polynomial-time players, a sender and a receiver, and consists of two phases, the commitment phase and the decommitment phase. In the commitment phase, the sender with a secret input x engages in a protocol with the receiver. In the end of this protocol, the receiver still does not know what x is, and subsequently, i.e., during the decommitment phase the sender can open only one possible value of x. In general there are two types of commitment schemes: statistically-secret and computationally-binding schemes, computationally-secret and statistically-binding schemes. The number of rounds in the commitment phase is the total number of messages exchanged in interaction (that is, both sender messages and receiver messages). A commitment scheme with one round in the commitment phase is called non-interactive.

The notion of non-malleability was first introduced in [5]. [5] points out that adversary should be not able to generate a commitment in which the committed

[⋆] This work is supported by the National Natural Science Foundation of China under Grant No. 60573031 and New Century Excellent Talent Program of Education Ministry of China under Grant NCET–05–0398.

value has any non-trivial relation with another value in a known commitment. In many applications non-malleable commitment schemes are required. [5] also shows that the existence of a knowledge extractor for a commitment scheme implies non-malleability, which provides a practical way that is also adopted by our paper to prove non-malleability of a commitment scheme.

Commonly, knowledge extractors work by repeatedly rewinding the sender to the step after having sent commitments and completing the executions with independent random challenges to produce the committed message. However, the rewinding strategy makes extractors of many known commitment schemes run in expected polynomial-time instead of strictly polynomial-time ([5][9]). The relaxed requirement (that allows for expected polynomial-time extraction) is less than satisfactory for philosophical, extraction and technical considerations (For details, see [2]). Furthermore [2] points out that expected polynomial-time extractors allow for the attack that trade-offs between running time and success probability, while strictly polynomial-time extractors rule out any such attack. The extractors that need not rewinding sender are called online (Such extractors are also called straight-line in some literatures). In this paper we will present a non-interactive non-malleable commitment scheme with an online extractor that runs in strictly polynomial-time in the random oracle and the public parameter model.

1.1 Previous Works

The notion of non-malleability was first formalized and implemented by Dolev, Dwork and Naor in [5]. Their main result is the first implementation of non-malleable commitment based on any one-way function. The drawbacks of their solution are that it requires at least logarithmic number of rounds in interaction between sender and receiver and it uses costly zero-knowledge proofs and the extractor runs in expected polynomial-time.

There are also several round-efficient non-malleable protocols known in the shared random string model. Sahai [14] constructed a single-round, i.e., non-interactive, non-malleable zero-knowledge proof system in this model. This scheme was improved by De Santis, Di Crescenzo, Ostrovski, Persiano and Sahai [6]. Di Crescenzo, Ishai and Ostrovsky [7] constructed in the shared random string model a non-interactive commitment scheme that is non-malleable in a weaker sense than [5] (non-malleable with respect to opening). Di Crescenzo, Katz, Ostrovski, and Smith [8] constructed in the shared random string model a non-interactive commitment satisfying the stronger notion of non-malleability (non-malleable with respect to commitment) defined in [5]. Canetti and Fischlin [4] constructed in the shared random string model a non-interactive universally composable commitment scheme which is a stronger notion than non-malleability. Interestingly, it is impossible to construct universally composable commitments in the plain model [4]. Barak [1] proposed the first const-round non-malleable scheme without any shared random strings through his non-black-box techniques. Barak and Lindell [2] pointed out that expected polynomial-time extractors allow for the attack that trade-offs between running time and success probability, while strictly polynomial-time extractors rule out any such attack. [2] shows that there exist constant-round zero-knowledge arguments of knowledge with strictly polynomial-time extractors.

Based on algebraic assumptions, one can build a non-malleable commitment scheme using non-interactive zero-knowledge proofs of knowledge of De Santis and Persiano [15]. That is, [15] implements non-interactive zero-knowledge proofs of knowledge assuming the existence of so-called "dense" cryptosystems. But the scheme uses inefficient zero-knowledge subprotocols.

Fischlin [11] proposed a discrete-log based non-malleable commitment scheme, which is the improvement of Pederson's non-interactive (but malleable) commitment scheme [13] and Okamoto's identification scheme [12]. Its commitment phase needs three-round interaction and the knowledge extractor is expected polynomial-time. With a stronger assumption, that of the existence of cryptographic hash functions which behave like random oracles, Bellare and Rogaway [3] showed how to implement non-malleable commitments in an efficient way. However, [3] does not supply an efficient knowledge extractor to verify proofs of knowledge.

In summary, in previous works either these schemes are too theoretical to put into practice, or knowledge extractors need rewinding and are not strictly polynomial-time. Recently, Fischlin [10] showed a construction converting Fiat-Shamir proofs of knowledge with logarithmic challenge length into non-interactive zero-knowledge proofs of knowledge with online extractors for relations in the random oracle model. The important figure of the online extractors is that the extractors do not rewind senders and run in strictly polynomial-time. Therefore they are efficient, which is the motivation of our work to be described in the following.

1.2 Our Results

We propose a discrete-logarithm based non-interactive non-malleable commitment scheme with an online knowledge extractor in the random oracle and the public parameter model. Our scheme is statistically-secret computationally-binding. Our results are obtained in three steps. The starting point of our work is Okamoto's discrete-log based identification scheme in [12], which challenge length is not logarithmic in the security parameter. Hence firstly, we reduce the challenge length in Okamoto's scheme in [12] to the logarithm and thus show that it is a Fiat-Shamir proof of knowledge for some relation with logarithmic challenge length. Secondly, we employ Fischlin's work in [10] to convert it into a non-interactive zero-knowledge proof of knowledge with an online knowledge extractor for the relation, which makes the knowledge extraction procedure more efficient. Thirdly, we propose our commitment scheme based on the non-interactive zero-knowledge proof of knowledge. Compared with previous works, our scheme is practical and the online knowledge extractor is strictly polynomial-time.

1.3 Organization

The rest of this paper is organized as follows. Section 2 contains notions and definitions of non-interactive non-malleable commitment schemes as well as Fiat-Shamir proofs of knowledge and non-interactive zero-knowledge proofs of knowledge with online extractor for relations. Section 3 presents the details of our results.

2 Preliminaries

2.1 Basic Notions

Unless stated otherwise all parties and algorithms are probabilistic polynomial-time. Throughout this paper, we use the notion of uniform algorithms.

Definition 1. *(Negligible and Noticeable Functions) A function $\varepsilon(n)$ is said to be negligible if $\varepsilon(n) < \frac{1}{p(n)}$ for every polynomial $p(n)$ and sufficiently large n; Denote by $neg(n)$ some negligible function; A function which is not negligible is called noticeable.*

Definition 2. *(Computational Indistinguishability) Two ensembles $X=\{X_n\}_{n\in N}$ and $Y = \{Y_n\}_{n\in N}$ are called computational indistinguishable if for all polynomial-time algorithm A, $|\Pr[A(X_n) = 1] - \Pr[A(Y_n) = 1]|$ is negligible.*

Definition 3. *(Statistical Differences) Given two random variables X_n and Y_n, statistical differences of X_n and Y_n is defined as*
$$\Delta(n) = \frac{1}{2}\sum_{\alpha} |\Pr(X_n = \alpha) - \Pr(Y_n = \alpha)|.$$

Definition 4. *(Statistical Indistinguishability) Two ensembles $X = \{X_n\}_{n\in N}$ and $Y = \{Y_n\}_{n\in N}$ are called statistically indistinguishable if their statistical difference is negligible.*

We introduce the discrete-logarithm assumption in the following, which states that computing logarithm in groups like $\mathbb{Z}_p^*$ or elliptic curves is intractable. We simplify and consider the discrete-log problem with respect to prime order subgroups of $\mathbb{Z}_p^*$ only. Also the assumption can be transferred to other prime order groups like elliptic curves. Suppose there is an efficient algorithm $\mathrm{IndexGen}(1^k)$ generating a random prime p, an k-bit prime q with $q|p-1$, and a generator g of a subgroup G_q of order q. Then:

Definition 5. *(Discrete Logarithm Assumption) For any probabilistic polynomial time algorithm A, the inversion probability $\Pr[A(1^k, p, q, g, g^x \bmod p) = x]$ is negligible in k, where the probability is taken over the choice of $(p, q, g) \leftarrow \mathrm{IndexGen}(1^k)$, $x \in_{\mathrm{R}} \mathbb{Z}_q$ and A's internal random coins.*

2.2 Non-interactive Non-malleable Commitment Schemes

Denote by k the security parameter. Assume that the messages of the sender and the adversary are taken from a space D. Abusing notations, we view D also as an efficiently computable distribution, and write $x \in_{\mathrm{R}} D$ for a randomly drawn message according to D.

Definition 6. *(Non-Interactive Statistically-Secret Commitment Scheme) A tuple $(\mathcal{S}, \mathcal{R}, \mathcal{T})$ of probabilistic polynomial-time algorithms $\mathcal{S}$ (Sender), $\mathcal{R}$ (Receiver), $\mathcal{T}$ (Trusted party) is called a non-interactive statistically-secret computationally -binding commitment scheme (in the public parameter model) if:*

1. *(Completeness) For all $k \in N$ and message $m_k \in D_k$, let $\mathrm{com}_k \leftarrow \mathcal{S}(m_k, \sigma)$ denote the commitment for m_k where $\sigma \leftarrow \mathcal{T}(1^k)$ is public parameters,* $\Pr[\mathcal{R}(\mathrm{com}_k, \sigma) = \mathrm{accept}] \geq 1 - neg(k)$ *(in commitment phase) and* $\Pr[\mathcal{R}(m_k, \mathrm{com}_k, \sigma) = \mathrm{accept}] = 1$ *(in decommitment phase).*
2. *(Statistically-Secret) For any sequences $(m_k)_{k \in N}$ and $(m'_k)_{k \in N}$ of messages $m_k, m'_k \in M_k$, the corresponding commitments $(\mathrm{com}_k)_k$ and $(\mathrm{com}'_k)_k$ are statistically indistinguishable; If they are identically distributed we say that the scheme provides perfectly-secret.*
3. *(Computationally-Binding) For any probabilistic polynomial-time $\mathcal{S}^*$, the probability that* $\mathrm{com}_k \leftarrow S^*(\sigma)$ $\mathcal{R}(\mathrm{com}_k, \sigma) = \mathrm{accept}$ *(in commitment phase),* $(m_k, m'_k) \leftarrow S^*(\mathrm{com}_k, \sigma)$ $R(m_k, \mathrm{com}_k, \sigma) = R(m'_k, \mathrm{com}_k, \sigma) = \mathrm{accept}$ *(in decommitment phase), is negligible (over the internal random coins of all parties).*

We follow [9] to briefly show non-malleability. Denote by R any interesting relation approximator. The definition on non-malleable commitment comes up with the well known idea of defining secure encryption, namely, we will demand that for any adversary $\mathcal{A}$ transforming the sender's commitment successfully, there should be an adversary simulator $\mathcal{A}'$ that finds a commitment to a related message at almost the same probability as $\mathcal{A}$ but without the sender's help.

Let PubPar be the public parameters that are generated by a trusted party according to a publicly known efficiently samplable distribution, Hist be a-priori information that the adversary has about m. On input PubPar the adversary $\mathcal{A}$ then picks the adversarial parameters AdvPar for D and R. The sender $\mathcal{S}$ is initialized with $m \in_{\mathrm{R}} D(\mathrm{AdvPar})$. Now $\mathcal{A}$, given $\mathrm{Hist}(m)$, mounts a Man-In-The-Middle (MIM) attack with $\mathcal{S}(m)$ and $\mathcal{R}$. In cases of interactive commitment schemes, the adversary might gain advantage from interaction. In non-interactive cases, all the adversary can do is to modify a given commitment. Denote by $p_{\mathrm{com}}(\mathcal{A}, D, R)$ the probability that, at the end of the commitment phase, the protocol execution between $\mathcal{A}$ and $\mathcal{R}$ constitutes a valid commitment for some message m^* satisfying $R(\mathrm{AdvPar}, \mathrm{Hist}(m), m, m^*)$. Let $p_{\mathrm{open}}(\mathcal{A}, D, R)$ denote the probability that $\mathcal{A}$ is able to successfully open the commitment after $\mathcal{S}$ has decommitted.

In a second experiment, a simulator $\mathcal{A}'$ tries to commit to a related message without the help of the sender. That is, $\mathcal{A}'$ gets as input random public parameters PubPar, generates adversarial parameters AdvPub' and then, given $\text{Hist}(m)$ for some $(m, \text{Hist}(m)) \in_{\text{R}} D(\text{AdvPar}')$, it commits to $\mathcal{R}$ without interaction with $\mathcal{S}(m)$. Let $p_{\text{com}}(\mathcal{A}', D, R)$ denote the probability that this is a valid commitment to some related message m' under PubPar with respect to relation $R(\text{AdvPar}', \text{Hist}(m), \cdot, \cdot)$. Let $p_{\text{open}}(\mathcal{A}', D, R)$ denote the probability that $\mathcal{A}'$ simply outputs a correct decommitment (without reference to public parameters, commitment and decommitment). So far we can bring forward the notion of non-malleability concluded in [9].

Definition 7. *(Non-Malleability) A non-interactive statistically-secret computationally -binding commitment scheme is called:*

a) non-malleable with respect to commitment if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{A}'$ such that for all samplable messages space D, all interesting relations R $|p_{\text{com}}(\mathcal{A}, D, R) - p'_{\text{open}}(\mathcal{A}', D, R)|$ is negligible.
b) non-malleable with respect to opening if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{A}'$ such that for all samplable messages space D, all interesting relations R $|p_{\text{open}}(\mathcal{A}, D, R) - p'_{\text{open}}(\mathcal{A}', D, R)|$ is negligible.
c) non-malleable with respect to DDN if for every adversary $\mathcal{A}$ there exists a simulator $\mathcal{A}'$ such that for all samplable messages space D, all interesting relations R $|p_{\text{com}}(\mathcal{A}, D, R) - p'_{\text{com}}(\mathcal{A}', D, R)|$ is negligible.

As [9] shows, non-malleability with respect to commitment implies non-malleability respect to opening and with respect to DDN.

2.3 Two Alternative Models for Zero-Knowledge Proofs of Knowledge

Definition 8. *([10]) A Fiat-Shamir proof of knowledge (with $l(k)$-bit challenges) for relation W is a pair (P, V) of probabilistic polynomial-time algorithms $P = (P_0, P_1)$, $V = (V_0, V_1)$ with the following properties.*

1.(Completeness) For any parameter k, any $(x, w) \in W_k$, any $(\text{com}, \text{ch}, \text{resp}) \leftarrow (P(x, w), V_0(x))$, it holds $V_1(x, com, ch, resp) = 1$.
2.(Commitment Entropy) For parameter k, for any $(x, w) \in W_k$, the min-entropy of com $P_0(x, w)$ is superlogarithmic in k.
3.(Public Coin) For any k, any $(x, w) \in W_k$ any $\text{com} \leftarrow P_0(x, w)$ the challenge $\text{ch} \leftarrow V_0(x, \text{com})$ is uniform on $\{0, 1\}^{l(k)}$.
4.(Unique Responses) For any probabilistic polynomial-time algorithm A, for parameter k and $(x, \text{com}, \text{ch}, \text{resp}, \text{resp}') \leftarrow A(1^k)$ we have, as a function of k, $\Pr[V_1(x, \text{com}, \text{ch}, \text{resp}) = V_1(x, \text{com}, \text{ch}, \text{resp}') = 1 \wedge \text{resp} \neq \text{resp}']$ is negligible.
5.(Special Soundness) There exists a probabilistic polynomial-time algorithm K, the extractor, such that for any k, any $(x, w) \in W_k$, any pair $(\text{com}, \text{ch}, \text{resp})$, $(\text{com}, \text{ch}', \text{resp}')$ with $V_1(x, \text{com}, \text{ch}, \text{resp}) = V_1(x, \text{com}, \text{ch}', \text{resp}') = 1$ and $\text{ch} \neq \text{ch}'$, for $w' \leftarrow K(x, \text{com}, \text{ch}, \text{resp}, \text{ch}', \text{resp}')$ it holds $(x, w') \in W_k$.

6. *(Honest-Verifier Zero-Knowledge) There exists a probabilistic polynomial-time algorithm Z, the zero-knowledge simulator, such that for any pair of probabilistic polynomial-time algorithms $D = (D_0, D_1)$ the following distributions are computationally indistinguishable:*
   - Let $(x, w, \delta) \leftarrow D_0(1^k)$, and $(\mathrm{com}, \mathrm{ch}, \mathrm{resp}) \leftarrow (P(x, w), V_0(x))$ if $(x, w) \in W_k$, and $(\mathrm{com}, \mathrm{ch}, \mathrm{resp}) \leftarrow \perp$ otherwise. Output $D_1(\mathrm{com}, \mathrm{ch}, \mathrm{resp}, \delta)$.
   - Let $(x, w, \delta) \leftarrow D_0(1^k)$, and $(\mathrm{com}, \mathrm{ch}, \mathrm{resp}) \leftarrow Z(x, \mathrm{Yes})$ if $(x, w) \in W_k$, and $(\mathrm{com}, \mathrm{ch}, \mathrm{resp}) \leftarrow Z(x, \mathrm{No})$ otherwise. Output $D_1(\mathrm{com}, \mathrm{ch}, \mathrm{resp}, \delta)$.

We obtain the stronger properties of honest-verifier perfect/statistical zero-knowledge if above two distributions are identical/statistically indistinguishable.

Definition 9. *([10]) A pair (P, V) of probabilistic polynomial-time algorithms is called a non-interactive zero-knowledge proof of knowledge for relation W with an online extractor (in the random oracle model) if the following holds.*

1. *(Completeness) For any oracle H, any $(x, w, \delta) \leftarrow D^H(1^k)$ and any $\pi \leftarrow P^H(x, w)$ we have $\Pr[V^H(x, \pi) = 1] \geq 1 - neg(k)$.*
2. *(Zero-Knowledge) There exist a pair of probabilistic polynomial-time algorithms $Z = (Z_0, Z_1)$, the zero-knowledge simulator, such that for any pair of probabilistic polynomial-time algorithms $D = (D_0, D_1)$, the following distributions are computationally indistinguishable:*
   - Let H be a random oracle, $(x, w, \delta) \leftarrow D_0^H(1^k)$, and $\pi \leftarrow P^H(x, w)$ if $(x, w) \in W_k$, and $\pi \leftarrow \perp$ otherwise. Output $D_1^H(\pi, \delta)$.
   - Let $(H_0, \sigma) \leftarrow Z_0(1^k)$, $(x, w, \delta) \leftarrow D_0^{H_0}(1^k)$, and $(H_1, \pi) \leftarrow Z_1(\sigma, x, \mathrm{Yes})$ if $(x, w) \in W_k$, and $(H_1, \pi) \leftarrow Z_1(\sigma, x, \mathrm{No})$ otherwise. Output $D_1^{H_1}(\pi, \delta)$.
   We obtain the stronger properties of perfect/statistical zero-knowledge if above two distributions are identical/statistically indistinguishable.
3. *(Online Extractor) There exists a probabilistic polynomial-time algorithm K, the online extractor, such that the following holds for any algorithm A. Let H be a random oracle, $(x, \pi) \leftarrow A^H(1^k)$ and $Q^H(A)$ be the sequence of queries of A to H and H's answers. Let $w \leftarrow K(x, \pi, Q^H(A))$. Then, as a function of k, $\Pr[(x, w) \notin W_k \wedge V^H(x, \pi) = 1]$ is negligible.*

[10] proposes a construction that converts Fiat-Shamir proofs of knowledge with logarithmic challenge length to non-interactive zero-knowledge proofs of knowledge with online knowledge extractors for relations, which does not need any rewinding and produces much shorter proofs than the tree-based solutions while having comparable extraction error and running time characteristics. The construction is showed as follows.

Construction 1: Let $(P_{\mathrm{FS}}, V_{\mathrm{FS}})$ denote an interactive Fiat-Shamir proof of knowledge with challenges of $l = l(k) = O(\log k)$ bits for relation W. Define the parameters b, u, Sum, n (as functions of k) for the number of test bits, repetitions, maximum sum and trial bits such that $bu = \omega(\log k)$, $2^{n-b} = \omega(\log k)$, $b, u, n = O(\log k)$, $Sum = O(u)$ and $b \leq n \leq l$. Define the following non-interactive proof system (P, V) for relation W in the random oracle model, where the random oracle H maps to b bits.

Prover: The prover P^H on input (x, w) first runs the prover $P_{\mathrm{FS}}(x, w)$ in u independent repetitions to obtain u commitments $\mathrm{com}_1, \cdots, \mathrm{com}_u$. Let $\overrightarrow{\mathrm{com}} = (\mathrm{com}_1, \cdots, \mathrm{com}_u)$. Then P^H does the following, either sequentially or in parallel for each repetition i. For each $\mathrm{ch}_i = 0, 1, 2, \cdots, 2^t - 1$ (viewed as t-bit strings) it lets P_{FS} compute the final responses $\mathrm{resp}_i = \mathrm{resp}_i(\mathrm{ch}_i)$ by rewinding, until it finds the first one such that $H(x, \overrightarrow{\mathrm{com}}, i, \mathrm{ch}_i, \mathrm{reps}_i) = 0^b$; if no such tuple is found then P^H picks the first one for which the hash value is minimal among all 2^n hash values. The prover finally outputs $\pi = (\mathrm{com}_i, \mathrm{ch}_i, \mathrm{resp}_i)_{i=1,2,\cdots,u}$.

Verifier: The verifier V^H on input x and $\pi = (\mathrm{com}_i, \mathrm{ch}_i, \mathrm{resp}_i)_{i=1,2,\cdots,u}$ accepts if and only if $V_{1,\mathrm{FS}}(x, \mathrm{com}_i, \mathrm{ch}_i, \mathrm{resp}_i) = 1$ for each $V_{1,\mathrm{FS}}(x, \mathrm{com}_i, \mathrm{ch}_i, \mathrm{resp}_i) = 1$ and if $\sum_{i=1}^{u} H(x, \overrightarrow{\mathrm{com}}, i, \mathrm{ch}_i, \mathrm{resp}_i) \leq Sum$.

Theorem 1. *([10]) Let $(P_{\mathrm{FS}}, V_{\mathrm{FS}})$ be an interactive Fiat-Shamir proof of knowledge for relation W. Then (P, V) in Construction 1 is a non-interactive zero-knowledge proof of knowledge for relation W (in the random oracle model) with an online extractor.*

3 Our Results

Our results are presented by following three steps. The starting point of our work is Okamoto's discrete-log based identification scheme in [12], which challenge length is not logarithmic. In Section 3.1 we reduce the challenge length in his scheme to the logarithm of the security parameter. Then we prove that this variation of Okamoto's scheme is a Fiat-Shamir proof of knowledge for some relation with logarithmic challenge length. In Section 3.2 we employ the technique in [10] to convert this variation into a non-interactive zero-knowledge proof of knowledge for the relation with an online extractor, based on which we propose our commitment scheme in Section 3.3.

3.1 The Discrete-Log Based Fiat-Shamir Proof of Knowledge for Relation W with Logarithmic Challenge Length

Let G_q be a cyclic group of prime order q and g_0, h_0 are two random generators of G_q. Assume that computing the discrete logarithm $\log_{g_0} h_0$ is intractable. Let k denote the security parameter which represents the length of description of G_q and $\mathbb{Z}_q$. Then $k = O(\log q)$. Let p be a small number such that $p = O(\log q)$. Define relation $W = \{(g_0^m h_0^r, (m, r)) : m \in \mathbb{Z}_q^*, r \in \mathbb{Z}_q\}$. Okamoto's well known discrete-log based identification scheme in [12] is a proof of knowledge for relation W in which verifier randomly chooses a challenge from $\mathbb{Z}_q$. This means the bit length of challenges viewed as strings is $O(k)$ in Okamoto's scheme. Hence in order to apply Construction 1 in Section 3.2 we let verifier randomly choose a challenge from $\mathbb{Z}_p$, i.e., reduce the challenge length to $O(\log k)$ bits. Denote by $(P_{\mathrm{FS}}, V_{\mathrm{FS}})$ the variation of Okamoto's. We describe the details of $(P_{\mathrm{FS}}, V_{\mathrm{FS}})$

in Protocol 3.1 and prove that it is a Fiat-Shamir proof of knowledge with logarithmic challenge length later.

Protocol 3.1 $(P_{\mathrm{FS}}, V_{\mathrm{FS}}) = (P_0, P_1, V_0, V_1)$**:** The Fiat-Shamir proof of knowledge for W with logarithmic challenge length.
Common input: $G_q, p, q, g_0, h_0,\ M \in G_q$.
The private input to the prover: The witness (m, r) satisfying $M = g_0^m h_0^r$.
P_0: The prover first randomly chooses $s, t \in \mathbb{Z}_q$. Compute $S = g_0^s h_0^t$ and send S to the verifier.
V_0: The verifier randomly chooses a challenge $c \in \mathbb{Z}_p$. Send c to the prover.
P_1: After receiving c, the prover computes $y = s + cm \bmod q$, $z = t + cr \bmod q$. Then send y and z to the receiver.
V_1: Upon receiving the response y and z, the verifier determines whether accepts or not. The verifier accepts iff $SM^c = g_0^y h_0^z$. Otherwise reject it.

From Protocol 3.1 we can see that it is from $\mathbb{Z}_p$, instead of $\mathbb{Z}_q$ in Okamoto's [12], that the verifier chooses a random challenge in step V0. Since $p = O(k)$, then $\mathbb{Z}_p$ can be described using $O(\log k)$ bits. This implies that the bit length of challenges, viewed as strings, is $O(\log k)$. Furthermore, we have Theorem 2.

Theorem 2. $(P_{\mathrm{FS}}, V_{\mathrm{FS}})$ *in Protocol 3.1 is a Fiat-Shamir proof of knowledge for relation W with $O(\log k)$-bit challenges. Specially,* $(P_{\mathrm{FS}}, V_{\mathrm{FS}})$ *is honest-verifier perfect zero-knowledge.*

Proof. We have shown that the challenge length is $O(\log k)$ bits. Then in the following we show that completeness, commitment entropy, public coin, unique responses, special soundness and honest-verifier perfect zero-knowledge, according to Definition 8, are satisfied.

Completeness. Completeness is obviously satisfied. Whenever $(M, (m, r)) \in W_k$, for any s, t and challenge c the prover can easily compute $y = s + mc \bmod q$ and $z = s + rc \bmod q$ if he knows (m, r). Consequently, $SM^c = g_0^s h_0^t \cdot g_0^{mc} h_0^{rc} = g_0^{s+mc} h_0^{t+rc} = g_0^y h_0^z$.

Minimal Entropy. Since both s and t are randomly chose from Z_q, S is uniform distributed in G_q. Since $q = O(2^k)$, then the entropy of uniform distribution on G_q is $O(k)$ which is superlogarithmic in k.

Public Coin. Since c is uniform chose from $\mathbb{Z}_p$, then public coin property is satisfied.

Unique Response. Suppose there exists a probabilistic polynomial-time algorithm A and $(M, S, c, y, z, y', z') = A(1^k)$ such that $\Pr[V_1(M, S, c, y, z)) = V_1(M, S, c, y', z')) = 1 \wedge (y, z) \neq (y', z')]$ is noticeable. Then we can construct a reduction B to solve the discrete-log problem. The input to B are g_0, h_0 and the task of B is to compute $\log_{g_0} h_0$. B invokes oracle A and A responses (M, S, c, y, z, y', z') with noticeable probability such that $\Pr(V_1(M, S, c, y, z)) = V_1(M, S, c, y', z')) =$

$1 \wedge (y,z) \neq (y',z')]$ holds. That is to say, $(y,z) \neq (y',z')$ and $SM^c = g_0^y h_0^z$, $SM^c = g_0^{y'} h_0^{z'}$. Then $g_0^y h_0^z = g_0^{y'} h_0^{z'}$. Denote $\log_{g_0} h_0$ by x. Then B gains $g_0^{y'+xz'} = g_0^{y+xz}$. This implies that $y' + xz' = y + xz (\bmod q)$. Therefore B is able to compute $x = (y - y')(z' - z)^{-1} \bmod q$ with the same noticeable probability, which contradicts the discrete-log assumption.

Special Soundness. We construct a probabilistic polynomial-time algorithm K to extract knowledge. For any k and $(M,(m,r)) \in W_k$, suppose there exist a pair (M,c,y,z) and (M,c',y',z') such that $V_1(M,S,c,y,z)) = V_1(M,S,c',y',z') = 1$. Let K obtain the input (M,S,c,y,z,c',y',z'). Since $c \neq c'$, then $(y,z) \neq (y',z')$. Since $y = s + cm \bmod q$ and $y' = s + c'm \bmod q$, then $y - y' = (c - c')m(\bmod q)$. Hereafter, K is able to compute $m = (y - y')(c - c')^{-1} \bmod q$ as well as r in the same way.

Honest-Verifier Perfect Zero-Knowledge. We provide a zero-knowledge simulator Sim for the case of $(M,(m,r)) \in W_k$. The case of $(M,(m,r)) \notin W_k$ is trivial. Moreover, for every $(M,(m,r)) \in W_k$ an equivalent way to prove (perfect/statistical) zero-knowledge is to show that (S,c,y,z), in a real implementation of Protocol 3.1, and (S',c',y',z'), generated by Sim on M, are (identical/statistically indistinguishable) computationally indistinguishable. In the tuple (S,c,y,z) of a real implementation of Protocol 3.1, we have that S is uniform distributed on G_q, y,z are uniform distributed on $\mathbb{Z}_q$ and c is uniform distributed on $\mathbb{Z}_p$. In order to generate (S',c',y',z') with input M such that (S,c,y,z) and (S',c',y',z') are identical distributions, Sim works as follows: First randomly generates $y',z' \in \mathbb{Z}_q$ and $c' \in \mathbb{Z}_p$. Then Sim computes $S' = {(g_0^{y'} h_0^{z'})}/{M^{c'}}$ and outputs (S',c',y',z'). Then it is easy to verify that (S,c,y,z) and (S',c',y',z') are identical distributions and that (S',c',y',z') makes the verifier to accept M. Hence honest-verifier perfect zero-knowledge is satisfied. □

3.2 The Non-interactive Zero-Knowledge Proof of Knowledge with an Online Knowledge Extractor for Relation *W*

In this subsection we bring forward the non-interactive zero-knowledge proof of knowledge with an online extractor for relation W based on $(P_{\mathrm{FS}}, V_{\mathrm{FS}})$ described in Protocol 3.1 according to Construction 1.

Assume that $k, G_q, p, q, g_0, h_0, m, r, W$ have the same meaning as Section 3.1 shows. Set parameters $b, u, Sum = O(\log p)$ such that $bu = \omega(\log p)$, ${p}/{2^b} = \omega(\log p)$ and $b \leq \log p$. Let H be the random oracle that maps to b bits. Then we propose (P,V) in Protocol 3.2 according to Construction 1.

Protocol 3.2 (P,V)**:** The non-interactive zero-knowledge proof of knowledge for W with an online extractor in the random oracle model.
$(P_{\mathrm{FS}}, V_{\mathrm{FS}}) = (P_0, P_1, V_0, V_1)$ as Protocol 3.1 shows.
Common input: $G_q, p, q, g_0, h_0, b, u, Sum$, $M \in G_q$.
Private input to the prover: (m,r).

Prover: The prover P runs P_0 in Protocol 3.1 in u independent repetitions to obtain $S_1, \cdots, S_u$. Let $\boldsymbol{S} = (S_1, \cdots, S_u)$. P^H does the following: For each $i = 1, 2, \cdots, u$ and each $c_i = 0, 1, \cdots, p-1$, it let P_1 compute the final response y_i and z_i by rewinding, until it finds the first one such that $H(M, \boldsymbol{S}, i, c_i, y_i, z_i) = 0^b$; if no such tuple is found then P^H picks the first one for which the hash value is minimal among all p hash values. The prover finally sends $\pi = (S_i, c_i, y_i, z_i)_{i=1,2,\cdots,u}$ to the verifier.

Verifier: The verifier V^H on input M and $\pi = (S_i, c_i, y_i, z_i)_{i=1,2,\cdots,u}$ accepts if and only if $V_1(M, S_i, c_i, y_i, z_i) = 1$ for each $i = 1, 2, \cdots, u$ and if $\sum_{i=1}^{u} H(M, \boldsymbol{S}, i, c_i, y_i, z_i) \leq Sum$.

Theorem 3. *(1) (P, V) in Protocol 3.2 is a non-interactive zero-knowledge proof of knowledge for W with an online knowledge extractor in the random oracle model. (2) In particular, (P, V) is statistical zero-knowledge.*

Since $(P_{\text{FS}}, V_{\text{FS}})$ is a Fiat-Shamir proof of knowledge for W with $O(\log k)$-bit challenges and Protocol 3.2 is built according to Construction 1, as a result of Theorem 1, (1) of Theorem 3 comes into existence immediately.

From Theorem 2, $(P_{\text{FS}}, V_{\text{FS}})$ is honest-verifier perfect zero-knowledge. According to [10], we have that if the Fiat-Shamir proof of knowledge is honest-verifier perfect zero-knowledge, then the converted non-interactive zero-knowledge proof of knowledge is statistical zero-knowledge. As a direct result, (P, V) is statistical zero-knowledge.

Theorem 3 shows that the knowledge extractor of (P, V) is strictly polynomial-time and does not need any rewinding. That is, for any adversary $\mathcal{A}$ and its output (M, π), if (M, π) is valid, the knowledge extractor is able to find (m, r) when input $(M, \pi, Q^H(\mathcal{A}))$ with the probability negligible close to the probability that (M, π) is valid even without rewinding $\mathcal{A}$, where $Q^H(\mathcal{A})$ denotes the sequence of queries of $\mathcal{A}$ to H and H's answers. Hence the knowledge extraction procedure is efficient.

3.3 The Non-interactive Non-malleable Commitment Scheme with an Online Knowledge Extractor

In this subsection we propose the non-interactive non-malleable commitment scheme with an online knowledge extractor $(\mathcal{S}, \mathcal{R}, \mathcal{T})$ based on (P, V) in Protocol 3.2, where $\mathcal{T}$ denotes the trusted party that generate public parameters for sender and receiver. Assume that $k, G_q, p, q, g_0, h_0, m, r, W, b, u, Sum, H$ have the same meaning as Section 3.2 shows. Then our commitment scheme is described as Protocol 3.3 shows.

Protocol 3.3 $(\mathcal{S}, \mathcal{R}, \mathcal{T})$**:** Our commitment scheme in the random oracle model.
(P, V), H as Protocol 3.2 shows.
$\sigma = (G_q, p, q, g_0, h_0, m, r, b, u, Sum)$ generated by $\mathcal{T}(1^k)$.
Input to receiver $\mathcal{R}$**:** σ.
Input to sender $\mathcal{S}$**:** σ and $m \in \mathbb{Z}_q$.

Commitment phase

Sender: $\mathcal{S}$ randomly chooses $r \in \mathbb{Z}_q$ and computes $M = g_0^m h_0^r$. Run P^H to compute $\pi = (S_i, c_i, y_i, z_i)_{i=1,2,\cdots,u}$ and send (M, π) to receiver as the commitment.

Receiver: Upon receiving (M, π), $\mathcal{R}$ runs V^H to determine whether $V^H(M)$ accepts π or not. If V^H accepts, $\mathcal{R}$ accepts the commitment. Otherwise $\mathcal{R}$ rejects (M, π) as the commitment.

Decommitment phase

Sender: $\mathcal{S}$ sends m and r to $\mathcal{R}$.

Receiver: Upon receiving m and r, $\mathcal{R}$ determines whether m and r is valid opening or not. $\mathcal{R}$ accepts iff $M = g_0^m h_0^r$. Otherwise $\mathcal{R}$ rejects m and r.

Theorem 4. *$(\mathcal{S}, \mathcal{R}, \mathcal{T})$ in Protocol 3.3 is a non-interactive non-malleable (with respect to commitment) statistically-secret and computationally-binding commitment scheme with an online knowledge extractor (in the random oracle model).*

Proof. We show that completeness, statistically-secret, computationally-binding and non-malleability, according to Definition 6, 7, are satisfied.

Completeness. Whenever $\mathcal{S}$ randomly chooses r and runs P^H to generate the commitment (M, π) for message m, then according to completeness of (P, V), we have $\Pr[V^H(M, \pi, \sigma) = 1] \geq 1 - \text{neg}(k)$. Hence in the commitment phase $\Pr[\mathcal{R}(M, \pi, \sigma) = \text{accept}] \geq 1 - \text{neg}(k)$. In decommitment phase since $M = g_0^m h_0^r$, $\Pr[\mathcal{R}(m, r, M, \pi, \sigma) = \text{accept}] = 1$ consequently.

Statistically-Secret. Denote by $((M, \pi))_k$ and $((M', \pi'))_k$ two sequence commitments for any two different sequences $(m)_k$ and $(m')_k$. Since $(r)_k$ and $(r')_k$ are identical random ensembles, then $(M)_k$ and $(M')_k$ are identical random ensembles. According to Theorem 3, (P, V) is statistical zero-knowledge for relation W. Then for any same sequence in $(M)_k$ and $(M')_k$, $(\pi)_k$ and $(\pi')_k$ are statistically indistinguishable, which is a corollary of the fact that both $(\pi)_k$ and $(\pi')_k$ are statistically indistinguishable from the output of the zero-knowledge simulator. Therefore $(\pi)_k$ and $(\pi')_k$ are statistically indistinguishable for random ensembles $(M)_k$ and $(M')_k$. Thus $((M, \pi))_k$ and $((M', \pi'))_k$ are statistically indistinguishable.

Computationally-Binding. Suppose that these exists a probabilistic polynomial-time $\mathcal{S}^*$ that can reveal different values (m, r) and (m', r') in opening for the same valid commitment (M, π) with noticeable probability such that $\mathcal{R}(m, r, M, \pi, \sigma)$= $\mathcal{R}(m', r', M, \pi, \sigma) = \text{accept}$. Then $\mathcal{S}^*$ is able to solve the discrete-log problem with the same noticeable probability, which contradicts the discrete-log assumption. In fact, since $\mathcal{R}$ accepts both (m, r) and (m', r') as correct opening, then $g_0^m h_0^r = g_0^{m'} h_0^{r'}$. Denote $\log_{g_0} h_0$ by x. Then $\mathcal{S}^*$ gains $g_0^{m'+xr'} = g_0^{m+xr}$. This implies that $m' + xr' = m + xr (\bmod q)$. Therefore $x = (m - m')(r' - r)^{-1} \bmod q$.

Non-Malleability. We show that $(\mathcal{S}, \mathcal{R}, \mathcal{T})$ has a knowledge extractor in the first. That is to say, we show that there exists a knowledge extractor K which can find the committed message. K works as following shows. For any message m, any adversary $\mathcal{A}$ randomly chooses $r \in \mathbb{Z}_q$ and computes M. $\mathcal{A}$ sends his commitment (M, π) to K. Then K calls the knowledge extractor K' of the proof of knowledge (P, V) to find m and r in the random oracle model. Finally, K outputs m. Denote by $p_{\text{com}}(\mathcal{A})$ the probability that $\mathcal{A}$ succeeds in outputting a commitment after seeing some commitment (M, π), by $p(K')$ the probability that K' finds m and r. Then according to the online extractor property of Definition 9, we have $|p(\mathcal{A}) - p(K')|$ is negligible. Let $p(K)$ denote the probability that K succeeds in finding m. Then $p(K) = p(K')$. Hence $|p(\mathcal{A}) - p(K)|$ is also negligible.

Furthermore, since the online knowledge extractor K' of (P, V) is strictly polynomial-time and therefore is more efficient than those expected polynomial-time extractors which need to rewind adversaries in order to find secrets. As an inherited character, K is also an online extractor which is strictly polynomial-time and thus more efficient.

[5] provides a general construction of a non-malleability simulator $\mathcal{A}'$ from knowledge extraction procedure. We briefly review the construction of $\mathcal{A}'$ for our case. Let D denote any sampleable distribution on $\mathbb{Z}_q$ and R denote any relation approximator. $\mathcal{A}'$ prepares the public parameters as required for the extraction procedure, invokes the adversary $\mathcal{A}$ to obtain AdvPar and sets AdvPar'=AdvPar. Then the honest sender $\mathcal{S}$ is given a secret message $m \in_{\text{R}} D(\text{AdvPar}')$ and $\mathcal{A}'$ receives Hist(m) which is forward to $\mathcal{A}$ for the black-box simulation.

For the extraction procedure, $\mathcal{A}'$ randomly chooses $m_0 \in D(\text{AdvPar}')$ and computes (M, π) as a commitment for m_0. Since the commitment is statistically-secret it holds that even if m_0 does not match the a-priori information Hist(m) $\mathcal{A}$ can not be aware of this. Then $\mathcal{A}'$ sends (M, π) to $\mathcal{A}$. If $\mathcal{A}$ can output a related commitment (M', π') to $\mathcal{A}'$, then $\mathcal{A}'$ calls the knowledge extractor K to find the committed value (K' outputs the correct decommitment.). Since $|p(\mathcal{A}, D, R) - p(K)|$ is negligible, we have that $|p_{\text{com}}(\mathcal{A}, D, R) - p_{\text{open}}(\mathcal{A}', D, R)|$ is negligible. According to [9], this implies that both $|p_{\text{open}}(\mathcal{A}, D, R) - p_{\text{open}}(\mathcal{A}', D, R)|$ and $|p_{\text{com}}(\mathcal{A}', D, R) - p_{\text{open}}(\mathcal{A}', D, R)|$ are negligible. So our scheme is non-malleable with respect to commitment, opening and DDN simultaneously. □

Acknowledgments

We thank the anonymous reviewers of Inscrypt 2007 for useful comments and suggestions.

References

1. Barak, B.: Constant-Round Coin-Tossing With a Man in the Middle or Realizing the Shared Random String Model. In: Proc. 43rd FOCS, pp. 345–355. IEEE, Los Alamitos (2002)
2. Barak, B., Lindell, Y.: Strict Polynomial-time in Simulation and Extraction. Cryptology ePrint Archive, Report 2002/043, 2000. Extended abstract appeared in STOC 2002 (2002)

3. Bellare, M., Rogaway, P.: Random Oracles are Practical: A paradigm for Designing Efficient Protocols. In: Proc. of ACM Conference on Computer and Communication Security, pp. 62–73 (1993)
4. Canetti, R., Fischlin, M.: Universally Composable Commitments. Cryptology ePrint Archive, Report 2001/055, 2001. Extended abstract appeared in Crypto 2001 (2001)
5. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. SIAM J. Comput. 30(2), 391–437 (2000) (electronic), Preliminary version in STOC 1991
6. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust Non-interactive zero-knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001)
7. Di Crescenzo, G., Ishai, Y., Ostrovsky, R.: Non-interactive and non-malleable commitment. In: Proc. 30th STOC, pp. 141–150. ACM, New York (1998)
8. Di Crescenzo, G., Katz, J., Ostrovsky, R., Smith, A.: Efficient and Non-Interactive Non-Malleable Commitment. Cryptology ePrint Archive, Report 2001/032, 2001. Preliminary versoin in Eurocrypt 2001 (2001)
9. Fischlin, M.: Trapdoor Commitment Schemes and Their Application. PhD Dissertation, Goethe-University: Germany (2001)
10. Fischlin, M.: Communication-Efficient Non-Interactive Proofs of Knowledge with Online Extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005)
11. Fischlin, M., Fischlin, R.: Efficient Non-Malleable Commitment Schemes. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 413–431. Springer, Heidelberg (2000)
12. Okamoto, T.: Provably Secure and Practical Idenitification Schemes and Corresponding Signature Schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
13. Pedersen, T.: Non-interactive and Information-Theoretical Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
14. Sahai, A.: Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In: Proc. 40th FOCS, pp. 543–553. IEEE, Los Alamitos (1999)
15. De Santis, A., Persiano, G.: Zero-Knowledge Proofs of Knowledge Without Interaction. In: Proc. of FOCS 1992, pp. 427–436 (1992)

Verifiable Multi-secret Sharing Schemes for Multiple Threshold Access Structures

Christophe Tartary[1,2], Josef Pieprzyk[3], and Huaxiong Wang[1,3]

[1] Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University
Singapore
[2] Institute for Theoretical Computer Science
Tsinghua University
Beijing, 100084
People's Republic of China
[3] Centre for Advanced Computing - Algorithms and Cryptography
Department of Computing
Macquarie University
NSW 2109 Australia
{ctartary,josef}@ics.mq.edu.au
HXWang@ntu.edu.sg

Abstract. A multi-secret sharing scheme allows several secrets to be shared amongst a group of participants. In 2005, Shao and Cao developed a verifiable multi-secret sharing scheme where each participant's share can be used several times which reduces the number of interactions between the dealer and the group members. In addition some secrets may require a higher security level than others involving the need for different threshold values. Recently Chan and Chang designed such a scheme but their construction only allows a single secret to be shared per threshold value.

In this article we combine the previous two approaches to design a multiple time verifiable multi-secret sharing scheme where several secrets can be shared for each threshold value. Since the running time is an important factor for practical applications, we will provide a complexity comparison of our combined approach with respect to the previous schemes.

Keywords: Secret Sharing Scheme, Threshold Access Structures, Share Verifiability, Chinese Remainder Theorem, Keyed One-Way Functions.

1 Introduction

In 1979, Blakley and Shamir independently invented (t, n)-threshold secret sharing schemes in order to facilitate the distributed storage of secret data in an unreliable environment [1, 18]. Such a scheme enables an authority called *dealer* to distribute a *secret* s as *shares* amongst n participants in such a way that any group of minimum size t can recover s while no groups having at most $t-1$ members can get any information about s.

Sometimes, however, several secrets have to be shared simultaneously. A basic idea consists of using a (t, n)-threshold scheme as many times as the number of secrets.

This approach, however, is memory consuming. As noticed by Chien et al. [4], multi-secret sharing schemes can be used to overcome this drawback. In such a construction, multiple secrets are protected using the same amount of data usually needed to protect a single secret. Multi-secret sharing schemes can be classified into two families: one-time schemes and multiple time schemes [12]. One-time schemes imply the dealer must redistribute new shares to every participant once some particular secrets have been reconstructed. Such a redistribution process can be very costly both in time and resources, in particular, when the group size n gets large as it may be the case in group-oriented cryptography [6].

Several constructions of multiple time schemes have been achieved [4, 25]. Nevertheless they have the drawback that a dishonest dealer who distributes incorrect shares or a malicious participant who submits an invalid share to the combiner prevents the secrets from being reconstructed. The idea of robust computational secret sharing schemes was introduced by Krawczyk [14] to deal with this problem. Several such protocols were developed. Harn designed a verifiable multi-secret sharing scheme [10] which was extended by Lin and Wu [15]. In [3], Chang et al. recently improved that construction even further by providing resistance against cheating by malicious participants and reducing the computational complexity with respect to [10, 15]. The security of that scheme relies on the intractability of both factorization and discrete logarithm problem modulo a composite number. In [25], another multi-secret sharing scheme was developed by Yang et al. As [4], its security is based on the existence of keyed one-way functions introduced by Gong in [9]. Shao and Cao recently extended Yang et al.'s scheme by providing the verification property and reducing the number of public values[19].

It may occur that the same group of n participants share several secrets related to different threshold values according to their importance. As an example, consider that an army commander requests a strike to be executed and transmits the order to a group of 10 generals. One can imagine that any pair of officers can reconstruct the coordinates of the target and then initialize the process by mobilizing the appropriate equipment (plane, submarine, missile) but only subsets of 8 out of 10 generals can get access to the bomb activation code and launch the strike. Recently Chan and Chang designed such a construction [2] but it only allows a single secret to be shared per threshold value.

In this article, we propose a generalization of [2, 19] by introducing a Verifiable Multi-Threshold Multi-secret Sharing Scheme (VMTMSS) where several secrets can be shared per threshold value. The security of our multiple time scheme is guaranteed as soon as keyed one-way functions and collision resistant one-way functions exist. In the previous situation, our VMTMSS would enable any pair of generals to have access to target location, launch time, type of weapon to be used while any subset of 8 out of 10 officers can recover the bomb code as well as the commander's digital signature [20] as the approval for the strike. This example also emphasizes the need for computational efficiency. Therefore we will also provide an analysis of the computational cost of our construction.

This paper is organized as follows. In the next section we will recall the polynomial interpolation problem as well as Garner's algorithm since they will have an important role in our construction. In Sect. 3, we will describe our multi-secret sharing scheme and prove its soundness. In Sect. 4, we will analyze the computational complexity of our approach and compare it to the cost of the two constructions from [2, 19]. The last section will summarize the benefits of our construction.

2 Preliminaries

In this part we recall two problems which will play an important role in proving the soundness and efficiency of the scheme we describe in Sect. 3.

2.1 Interpolating Points

Assume that we are given λ points $(x_1, y_1), \dots, (x_\lambda, y_\lambda)$ such that the x_i's are distinct in a field $\mathbb{K}$. The *Lagrange interpolating polynomial* $L_\lambda(X)$ is the only polynomial of degree at most $\lambda - 1$ passing through the previous λ points. Algorithm 4.6.1 from [8] computes the λ coefficients of $L_\lambda(X)$ using $\frac{5\,(\lambda-1)^2}{2}$ field operations in $\mathbb{K}$.

We now consider that we work over the finite field $\mathbb{Z}/p\mathbb{Z}$ for some prime number p. In this field an addition/subtraction requires $O(\log_2 p)$ bit operations and a multiplication needs $O(\log_2^2 p)$ bit operations. Using Algorithm 14.61 and Note 14.64 from [16], an inversion can be performed in $O(\log_2^2 p)$ bit operations as well. Therefore the λ coefficients of $L_\lambda(X)$ can be obtained using $O(\lambda^2\ \log_2^2 p)$ bit operations.

2.2 Solving the Chinese Remainder Problem

We first recall the Chinese Remainder Theorem (CRT):

Theorem 1. *Let $m_1, \dots, m_\lambda$ be λ coprime integers and denote M their product. For any λ-tuple of integers $(v_1, \dots, v_\lambda)$, there exists a unique x in $\mathbb{Z}/M\mathbb{Z}$ such that:*

$$\begin{cases} x \equiv v_1 \mod m_1 \\ \quad\vdots \qquad \vdots \\ x \equiv v_\lambda \mod m_\lambda \end{cases}$$

Solving the Chinese remainder problem is reconstructing the unique x in $\mathbb{Z}/M\mathbb{Z}$ once $v_1, \dots, v_\lambda$ and $m_1, \dots, m_\lambda$ are given. This can be achieved thanks to Garner's algorithm [16]. Based on Note 14.74, its running time is $O(\lambda \log_2^2 M)$ bit operations.

3 Our Multi-secret Sharing Scheme

We assume that we have n participants $P_1, \dots, P_n$ and ℓ distinct threshold values $t_1, \dots, t_\ell$. Consider we have ℓ distinct prime numbers $p_1, \dots, p_\ell$. For each i in $\{1, \dots, \ell\}$ we denote $S_{i\,1}, \dots, S_{i\,k_i}$ the k_i secrets of the (t_i, n)-threshold scheme. Without loss of generality we can assume that those k_i secrets belong to $\mathbb{Z}/p_i\mathbb{Z}$. We first introduce the following definition:

Definition 1. *A function $f : \mathbb{R}^+ \to \mathbb{R}^+$ is said to be* negligible *if:*

$$\forall \alpha > 0\, \exists \zeta_0 \in \mathbb{R}^+ \,:\, \forall \zeta > \zeta_0 \quad f(\zeta) < \zeta^{-\alpha}$$

We have the following definition adapted from Definition 13.2 [20].

Definition 2. *A threshold multi-secret sharing scheme for threshold value t is a method of sharing k secrets $S_1, \ldots, S_k$ among a set of n participants $\{P_1, \ldots, P_n\}$ in such a way that the following properties are satisfied:*
(i) (soundness) If at least t participants pool their shares together then they recover the whole k secrets $S_1, \ldots, S_k$.
(ii) (secrecy) If at most $t-1$ participants pool their shares together then they do not recover the whole k secrets with non-negligible probability as a function of the secret's size.

The reader may notice that Definition 13.2 is related to perfect secrecy since it is there assumed that the coalition of $t-1$ participants does not know anything about the secret value (i.e. all values are equally probable). This cannot be held here as several secrets will be shared using the same polynomial. Nevertheless we will see that $t-1$ participants cannot recover the whole k secrets with good probability. We can generalize the previous definition as follows:

Definition 3. *A multiple-threshold multi-secret sharing scheme for threshold values $t_1, \ldots, t_\ell$ is a method of sharing $k_1 + \cdots + k_\ell$ secrets $S_{1\,1}, \ldots, S_{\ell\,k_\ell}$ among a set of n participants $\{P_1, \ldots, P_n\}$ in such a way that the following properties are satisfied:*
(i) (soundness) For each $i \in \{1, \ldots, \ell\}$, if at least t_i participants pool their shares together then they recover the whole k_i secrets $S_{i\,1}, \ldots, S_{i\,k_i}$.
(ii) (secrecy) For each $i \in \{1, \ldots, \ell\}$, if at most $t_i - 1$ participants pool their shares together then they do not recover the whole k_i secrets $S_{i\,1}, \ldots, S_{i\,k_i}$ with non-negligible probability as a function of the secret's size.

A *verifiable multiple-threshold multi-secret sharing scheme* (VMTMSS) is a multiple-threshold multi-secret sharing scheme for which the validity of the share can be publicly verifiable. Let us introduce the following definition from [9]:

Definition 4. *A function $f(\cdot,\cdot)$ that maps a key and a second bit string of a fixed length is a* secure keyed one-way hash function *if it satisfies the following five properties:*

P1: *Given k and x, it is easy to compute $f(k,x)$.*
P2: *Given k and $f(k,x)$, it is hard to compute x.*
P3: *Without knowledge of k, it is hard to compute $f(k,x)$ for any x.*
P4: *Given k, it is hard to find two distinct values x and y such that $f(k,x) = f(k,y)$.*
P5: *Given (possibly many) pairs $(x, f(k,x))$, it is hard to compute k.*

Remark, however, this secure keyed one-way function is not equivalent to the two-variable one-way function defined by He and Dawson in [11] contrary to what claimed Chien et al. [4]. Indeed the collision resistance property P4 of the keyed one-way function is not a requirement for the functions created by He and Dawson (see Definition 1 in [11]).

We assume that we have ℓ such functions $f_1, \ldots, f_\ell$ whose respective domains are $D_1, \ldots, D_\ell$. Without loss of generality we can assume that the prime numbers $p_1, \ldots, p_\ell$ are chosen such that: $\forall i \in \{1, \ldots, \ell\}$ $f_i(D_i) \subset \mathbb{Z}/p_i\mathbb{Z}$. We also assume: $\forall i \in \{1, \ldots, \ell\}$ $D_i \subset \mathbb{Z}/p_i\mathbb{Z} \times \mathbb{Z}/p_i\mathbb{Z}$. We need to use a collision resistant hash function H [17]. As in [13], it will be used to check the validity of the shares.

Our approach will consist of two steps. First we will treat each (t_i, n)-threshold scheme separately. We build a polynomial $F_i(X)$ whose degree and coefficients will be determined similarly to [25]. Second we will combine the ℓ polynomials $F_1(X), \ldots, F_\ell(X)$ using the following result obtained by extending Corollary 3.2 from [2]:

Corollary 1. *(Polynomial form of CRT) Let $m_1, \ldots, m_\lambda$ be λ coprime integers and denote their product by M. For any λ-tuple of polynomials $(A_1(X), \ldots, A_\lambda(X))$ from $\mathbb{Z}/m_1\mathbb{Z}[X] \times \cdots \times \mathbb{Z}/m_\lambda\mathbb{Z}[X]$, there exists a unique polynomial $A(X)$ in $\mathbb{Z}/M\mathbb{Z}[X]$ such that:*

$$\begin{cases} A(X) \equiv A_1(X) \bmod m_1 \\ \qquad \vdots \qquad\qquad \vdots \\ A(X) \equiv A_\lambda(X) \bmod m_\lambda \end{cases} \tag{1}$$

In addition: $\deg(A(X)) = \max_{i \in \{1,\ldots,\lambda\}} (\deg(A_i(X)))$.

Proof. In [2], Chan and Chang proved the existence of such a polynomial $A(X)$. What remains to demonstrate is its uniqueness and the value of its degree.

Let $A(X)$ be a polynomial from $\mathbb{Z}/M\mathbb{Z}[X]$ solution of (1) and denote α its degree. The ring isomorphism:

$$\mathbb{Z}/M\mathbb{Z} \simeq \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_\lambda\mathbb{Z} \tag{2}$$

involves $\alpha = \max_{i \in \{1,\ldots,\lambda\}} (\deg(A_i(X)))$ since (2) implies an element μ is congruent to 0 in $\mathbb{Z}/M\mathbb{Z}$ if and only if μ is congruent to 0 in each $\mathbb{Z}/m_i\mathbb{Z}$ for $i \in \{1, \ldots, \lambda\}$.

Let $A(X)$ and $\tilde{A}(X)$ be two solutions of (1). Since their degree is α, we can write them as:

$$A(X) := \sum_{i=0}^{\alpha} a_i\, X^i \qquad \text{and} \qquad \tilde{A}(X) := \sum_{i=0}^{\alpha} \tilde{a}_i\, X^i$$

where the a_i's and $\tilde{a}_i$'s are elements of $\mathbb{Z}/M\mathbb{Z}$. Since these polynomials are solutions of (1) and due to (2), we deduce: $\forall i \in \{0, \ldots, \alpha\}\ a_i \equiv \tilde{a}_i \bmod M$. □

The previous proof involves that $A(X)$ can be computed from $A_1(X), \ldots, A_\lambda(X)$ using Garner's algorithm $\alpha + 1$ times. We will now present the details of our construction.

3.1 Scheme Construction

Our construction consists of three algorithms: SetUp, ShareConstruction and SecretReconstruction. The first two algorithms will be run by the dealer while the last one will be executed by the combiner. As in [4, 19], SetUp will only be run once while ShareConstruction will be called each time new secrets are to be shared. The private elements distributed to the n participants by the dealer when running SetUp will ensure that our VMTMSS is a multiple time scheme.

SetUp

Input: The group size n and ℓ distinct prime numbers $p_1, \ldots, p_\ell$.

```
1. For each $i \in \{1, \ldots, \ell\}$, generate $n$ distinct elements of $\mathbb{Z}/p_i\mathbb{Z}$ denoted $s_{i\,1}, \ldots, s_{i\,n}$.
2. Use Garner's algorithm as: $\forall j \in \{1, \ldots, n\}\ \mathcal{S}_j := \text{Garner}(s_{1\,j}, \ldots, s_{\ell\,j}, p_1, \ldots, p_\ell)$.
3. Distribute $\mathcal{S}_j$ to participant $P_j$ over a secure channel for each $j \in \{1, \ldots, n\}$.
```

Output: The n private values $\mathcal{S}_1, \ldots, \mathcal{S}_n$ which will be used by the participants to check the validity of their pseudo-shares.

We have the following observation concerning [4, 19]. Each of the n participants P_i receives a secret value s_i. The dealer chooses a random element r and evaluates the *pseudo-shares* $f(r, s_1), \ldots, f(r, s_n)$ where f is the keyed one-way function used in those schemes. He builds a polynomial $h(X)$ whose k lowest degree coefficients represent the k secrets to be shared. Finally he publishes $r, h(f(r, s_1)), \ldots, h(f(r, s_n))$ so that the combiner can verify the validity of shares. In order to ensure the multiple time property of their construction, a new value r is generated each time a new set of k secrets is to be shared. If r is chosen such that $f(r, s_{i_0})$ is 0 then P_{i_0} can recover one of the secrets as the constant term of the polynomial $h(X)$ from the list of public elements since: $h(0) = h(f(r, s_{i_0}))$. Even if the probability of such an event is negligible when the domain of f is large, it is still easy to deal with this problem by shifting each coefficient of the polynomial $h(X)$ by one position and setting up the new constant term as a random element. This is at the cost of publishing an extra point to reconstruct $h(X)$ since its degree has increased by 1.

We will now introduce our algorithm ShareConstruction. We first introduce the following notation:

$$\forall i \in \{1, \ldots, \ell\} \qquad \delta_i := \begin{cases} 0 & \text{if } t_i \geq k_i \\ k_i - t_i & \text{otherwise} \end{cases}$$

Notice that δ_i can be computed as soon as both t_i and k_i are known.

ShareConstruction

Input: The group size n, the prime numbers $p_1, \ldots, p_\ell$, the threshold values $t_1, \ldots, t_\ell$, the number of secrets $k_1, \ldots, k_\ell$, the corresponding secrets $S_{1\,1}, \ldots, S_{1\,k_1}, \ldots, S_{\ell\,1}, \ldots, S_{\ell\,k_\ell}$, the functions $f_1, \ldots, f_\ell$, the elements $s_{1\,1}, \ldots, s_{\ell\,n}$ from SetUp and the collision resistant hash function H.

```
1. For each $i \in \{1, \ldots, \ell\}$, pick uniformly at random an element $r_i$ from $\mathbb{Z}/p_i\mathbb{Z}$. Use Garner's algorithm as: $\mathcal{R} := \text{Garner}(r_1, \ldots, r_\ell, p_1, \ldots, p_\ell)$.
2. Do the following:
    2.1. Compute $f_i(r_i, s_{i\,j})$ for $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, n\}$.
    2.2. Compute the hashes $H(f_i(r_i, s_{i\,j}))$ for $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, n\}$ and publish them as table $T_{\mathrm{H}}$.
    2.3. Use Garner's algorithm as: $\forall j \in \{1, \ldots, n\}\ \mathcal{P}_j := \text{Garner}(f_1(r_1, s_{1\,j}), \ldots, f_\ell(r_\ell, s_{\ell\,j}), p_1, \ldots, p_\ell)$.
3. For each $i \in \{1, \ldots, \ell\}$ do the following:
    3.1. Pick uniformly at random an element $C_i$ from $\mathbb{Z}/p_i\mathbb{Z}$.
    3.2. If $t_i > k_i$ then:
```

Pick uniformly at random $u_{i\,1}, \ldots, u_{i\,\delta_i}$ from $\mathbb{Z}/p_i\mathbb{Z}$.

Build the polynomial: $F_i(X) := C_i + \sum_{j=1}^{k_i} S_{i\,j}\, X^j + \sum_{j=1}^{t_i - k_i} u_{i\,j}\, X^{j+k_i}$

Else

Build the polynomial: $F_i(X) := C_i + \sum_{j=1}^{k_i} S_{i\,j}\, X^j$

4. Denote $D := \max_{i \in \{1,\ldots,\ell\}} (\deg(F_i(X)))$. For each $i \in \{1, \ldots, \ell\}$, write $F_i(X)$ as: $F_i(X) := \sum_{j=0}^{D} F_{i\,j}\, X^j$ where: $\forall j \in \{\deg(F_i(X)) + 1, \ldots, D\}$ $F_{i\,j} = 0$. Use Garner's algorithm as: $\forall j \in \{0, \ldots, D\}$ $\mathcal{F}_j :=$ Garner$(F_{1\,j}, \ldots, F_{\ell\,j}, p_1, \ldots, p_\ell)$.

5. Build the polynomial $\mathcal{F}(X)$ as: $\mathcal{F}(X) := \sum_{j=0}^{D} \mathcal{F}_j\, X^j$ and compute $\mathcal{F}(\mathcal{P}_1), \ldots, \mathcal{F}(\mathcal{P}_n)$.

6. Do the following:

6.1. For each $i \in \{1, \ldots, \ell\}$, generate an element a_i from $\mathbb{Z}/p_i\mathbb{Z}$ distinct from $s_{i\,1}, \ldots, s_{i\,n}$.

6.2. Use Garner's algorithm as: $\mathcal{A} :=$ Garner$(f_1(r_1, a_1), \ldots, f_\ell(r_\ell, a_\ell), p_1, \ldots, p_\ell)$.

6.3. Compute $\mathcal{F}(\mathcal{A})$.

7. For each $i \in \{1, \ldots, \ell\}$ such that $\delta_i > 0$ do the following:

7.1. Generate δ_i elements $s'_{i\,1}, \ldots, s'_{i\,\delta_i}$ such that $s_{i\,1}, \ldots, s_{i\,n}, a_i, s'_{i\,1}, \ldots, s'_{i\,\delta_i}$ are $n + 1 + \delta_i$ distinct elements of $\mathbb{Z}/p_i\mathbb{Z}$.

7.2. Compute $f_i(r_i, s'_{i\,1}), \ldots, f_i(r_i, s'_{i\,\delta_i})$.

7.3. Compute $F_i(f_i(r_i, s'_{i\,1})), \ldots, F_i(f_i(r_i, s'_{i\,\delta_i}))$.

8. Publish the table T containing $\mathcal{R}, \mathcal{F}(\mathcal{P}_1), \ldots, \mathcal{F}(\mathcal{P}_n), (\mathcal{A}, \mathcal{F}(\mathcal{A}))$ as well as the couples $(f_i(r_i, s'_{i\,1}), F_i(f_i(r_i, s'_{i\,1}))), \ldots, (f_i(r_i, s'_{i\,\delta_i}), F_i(f_i(r_i, s'_{i\,\delta_i})))$ for each i such that $\delta_i > 0$.

Output: The table T_{H} which will be used to verify the pseudo-shares and the table T which will be used to reconstruct the secrets of our VMTMSS.

Notice that $(\mathcal{A}, \mathcal{F}(\mathcal{A}))$ is the extra point needed to overcome the problem from [19]. We also remark that any participant P_j can compute the pseudo-shares $f_i(r_i, s_{i\,j})$ from the public value R and his secret element $\mathcal{S}_j$ since:

$$\begin{cases} r_i = \mathcal{R} \mod p_i \\ s_{i\,j} = \mathcal{S}_j \mod p_i \end{cases}$$

Using this information any participant can verify the validity of his pseudo-shares by checking their ℓ hashes from table T_{H}. Similarly the combiner can check the validity of any pseudo-share submitted during the secret reconstruction process using T_{H} as well. Notice, however, that the prime numbers $p_1, \ldots, p_\ell$ should be large enough in order to prevent an exhaustive search to be performed by an adversary who would compute $H(\zeta)$ (where $\zeta \in \mathbb{Z}/p_i\mathbb{Z}$) until finding a match amongst the n elements $H(f_i(r_i, s_{i\,1})), \ldots, H(f_i(r_i, s_{i\,n}))$. Figure 1 represents the previous two algorithms. The construction of polynomials $F_1(X), \ldots, F_\ell(X)$ and $\mathcal{F}(X)$ is depicted on Fig. 2.

We will now design SecretReconstruction which is run be combiner to recover the secrets. We assume that $P_{j_1}, \ldots, P_{j_{t_i}}$ are the t_i participants wishing to reconstruct the k_i secrets of the (t_i, n)-threshold scheme.

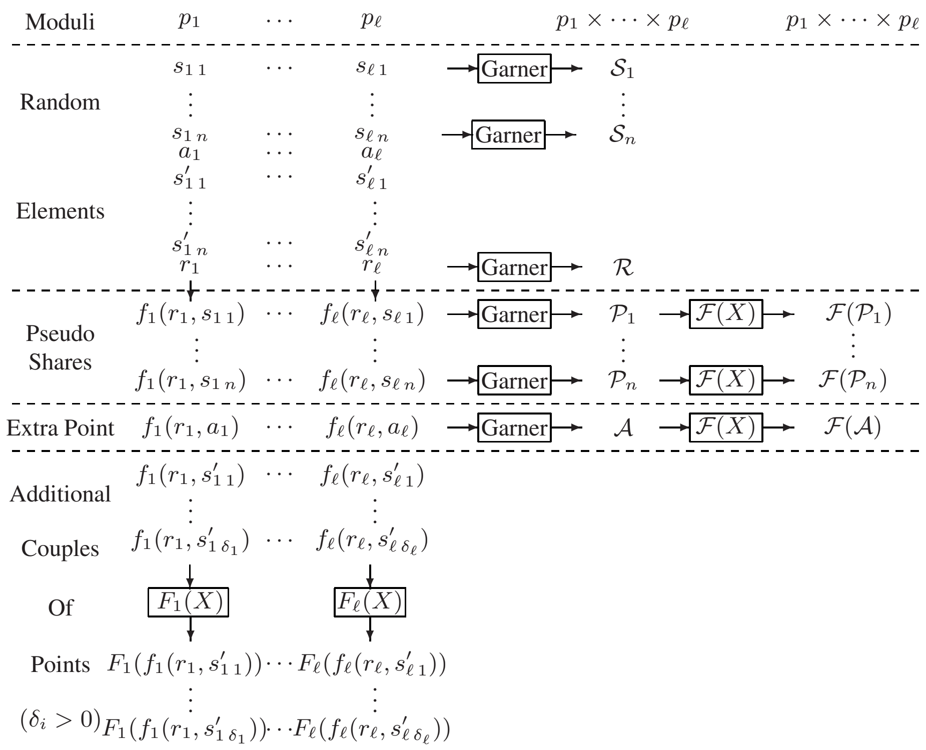

Moduli
p_1 $\cdots$ p_ℓ $p_1 \times \cdots \times p_\ell$ $p_1 \times \cdots \times p_\ell$
Random Elements
$s_{1\,1}$ $\cdots$ $s_{\ell\,1}$ Garner $\mathcal{S}_1$
$s_{1\,n}$ $\cdots$ $s_{\ell\,n}$ Garner $\mathcal{S}_n$
a_1 $\cdots$ a_ℓ
$s'_{1\,1}$ $\cdots$ $s'_{\ell\,1}$
$s'_{1\,n}$ $\cdots$ $s'_{\ell\,n}$
r_1 $\cdots$ r_ℓ Garner $\mathcal{R}$
Pseudo Shares
$f_1(r_1, s_{1\,1})$ $\cdots$ $f_\ell(r_\ell, s_{\ell\,1})$ Garner $\mathcal{P}_1$ $\mathcal{F}(X)$ $\mathcal{F}(\mathcal{P}_1)$
$f_1(r_1, s_{1\,n})$ $\cdots$ $f_\ell(r_\ell, s_{\ell\,n})$ Garner $\mathcal{P}_n$ $\mathcal{F}(X)$ $\mathcal{F}(\mathcal{P}_n)$
Extra Point $f_1(r_1, a_1)$ $\cdots$ $f_\ell(r_\ell, a_\ell)$ Garner $\mathcal{A}$ $\mathcal{F}(X)$ $\mathcal{F}(\mathcal{A})$
Additional Couples Of Points $(\delta_i > 0)$
$f_1(r_1, s'_{1\,1})$ $\cdots$ $f_\ell(r_\ell, s'_{\ell\,1})$
$f_1(r_1, s'_{1\,\delta_1})$ $\cdots$ $f_\ell(r_\ell, s'_{\ell\,\delta_\ell})$
$F_1(X)$ $F_\ell(X)$
$F_1(f_1(r_1, s'_{1\,1}))$ $\cdots$ $F_\ell(f_\ell(r_\ell, s'_{\ell\,1}))$
$F_1(f_1(r_1, s'_{1\,\delta_1}))$ $\cdots$ $F_\ell(f_\ell(r_\ell, s'_{\ell\,\delta_\ell}))$

Fig. 1. Representation of SetUp and ShareConstruction

SecretReconstruction

Input: The threshold value t_i, the number of secrets k_i, the prime numbers $p_1, \ldots, p_\ell$, the public table T as well as the pseudo-shares of the t_i participants $f_i(r_i, s_{i\,j_1}), \ldots, f_i(r_i, s_{i\,j_{t_i}})$.

1. Compute $x_{t_i+1} := \mathcal{A} \bmod p_i$ and $y_{t_i+1} := \mathcal{F}(\mathcal{A}) \bmod p_i$. For each $\lambda \in \{1, \ldots, t_i\}$, compute $y_\lambda := \mathcal{F}(\mathcal{P}_{j_\lambda}) \bmod p_i$.

2. If $\delta_i = 0$ then:

   2.1. Reconstruct the Lagrange interpolating polynomial passing through the points $(f_i(r_i, s_{i\,j_1}), y_1), \ldots, (f_i(r_i, s_{i\,j_{t_i}}), y_{t_i}), (x_{t_i+1}, y_{t_i+1})$.

   2.2. Write the polynomial obtained as: $\sum_{j=0}^{t_i} \mu_j\, X^j$ and return $\mu_1, \ldots, \mu_{k_i}$.

   Else

   2.3. Reconstruct the Lagrange interpolating polynomial passing through the points $(f_i(r_i, s_{i\,j_1}), y_1), \ldots, (f_i(r_i, s_{i\,j_{t_i}}), y_{t_i}), (x_{t_i+1}, y_{t_i+1}), (f_i(r_i, s'_{i\,1}), F_i(f_i(r_i, s'_{i\,1}))), \ldots, (f_i(r_i, s'_{i\,\delta_i}), F_i(f_i(r_i, s'_{i\,\delta_i})))$.

   2.4. Write the polynomial obtained as: $\sum_{j=0}^{k_i} \mu_j\, X^j$ and return $\mu_1, \ldots, \mu_{k_i}$.

Output: The k_i secrets $\mu_1, \ldots, \mu_{k_i}$ of the (t_i, n)-threshold scheme.

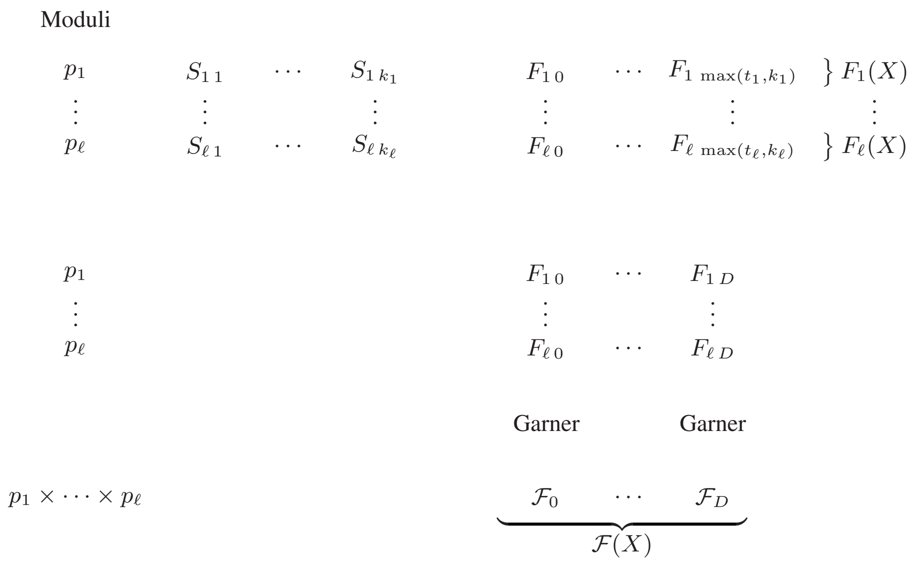

Moduli
p_1 $S_{1\,1}$ $\cdots$ $S_{1\,k_1}$ $F_{1\,0}$ $\cdots$ $F_{1\,\max(t_1,k_1)}$ $\}\,F_1(X)$
p_ℓ $S_{\ell\,1}$ $\cdots$ $S_{\ell\,k_\ell}$ $F_{\ell\,0}$ $\cdots$ $F_{\ell\,\max(t_\ell,k_\ell)}$ $\}\,F_\ell(X)$
p_1 $F_{1\,0}$ $\cdots$ $F_{1\,D}$
p_ℓ $F_{\ell\,0}$ $\cdots$ $F_{\ell\,D}$
Garner
Garner
$p_1 \times \cdots \times p_\ell$
$\underbrace{\mathcal{F}_0 \quad \cdots \quad \mathcal{F}_D}_{\mathcal{F}(X)}$

Fig. 2. Construction of Polynomials by the Dealer

3.2 Security Analysis

In this section, we have to demonstrate that our VMTMSS verifies the properties from Definition 3. In particular we have to argue that the table of hashes T_{H} and the table of extra points T do not leak too much information about the secrets. We have the following result for our construction:

Theorem 2. *The reconstruction algorithm* SecretReconstruction *is sound.*

Proof. We have to demonstrate that, for any value i in $\{1, \ldots, \ell\}$, the elements output by SecretReconstruction are the k_i secrets of the (t_i, n)-threshold scheme whatever the family of t_i participants is.

Let i be any element of $\{1, \ldots, \ell\}$. Consider $P_{j_1}, \ldots, P_{j_{t_i}}$ a family of t_i participants. Due to Steps 2, 4 and 5 of ShareConstruction2, we have the following result:

$$\forall i \in \{1, \ldots, \ell\}\, \forall \lambda \in \{1, \ldots, t_i\}\; F_i(f_i(r_i, s_{i\,j_\lambda})) = \mathcal{F}(\mathcal{P}_{j_\lambda}) \bmod p_i$$

Due to Property P4 of f_i, Step 1 of SetUp and Step 6.1 of ShareConstruction, the elements $f_i(r_i, s_{i\,j_1}), \ldots, f_i(r_i, s_{i\,j_{t_i}}), f_i(r_i, a_i)$ are distinct with overwhelming probability. Since $f_i(r_i, a_i) = \mathcal{A} \bmod p_i = x_{t_i+1}$, the $t_i + 1$ points $(f_i(r_i, s_{i\,j_1}), y_1), \ldots, (f_i(r_i, s_{i\,j_{t_i}}), y_{t_i}), (x_{t_i+1}, y_{t_i+1})$ have different abscissas in $\mathbb{Z}/p_i\mathbb{Z}$. We have two cases to consider:

First Case: $\delta_i = 0$**.** We can interpolate the previous $t_i + 1$ points as in Sect. 2.1 and denote $L_{t_i+1}(X)$ the corresponding Lagrange polynomial obtained at Step 2.1 of SecretReconstruction. It should be noticed that the polynomial $F_i(X)$ defined at Step 3.2

of ShareConstruction passes through the same points and has degree at most t_i (it is exactly t_i if the highest degree coefficient is different from 0). Due to the uniqueness of such a polynomial (see Sect. 2.1) we get: $L_{t_i+1}(X) = F_i(X)$. Thus the k_i coefficients returned at Step 2.2 of SecretReconstruction are the k_i secrets of the (t_i, n)-threshold scheme: $S_{i\,1}, \ldots, S_{i\,k_i}$.

Second Case: $\delta_i > 0$**.** Using table T, we obtain δ_i additional points: $(f_i(r_i, s'_{i\,1}), F_i(f_i(r_i, s'_{i\,1}))), \ldots, (f_i(r_i, s'_{i\,\delta_i}), F_i(f_i(r_i, s'_{i\,\delta_i})))$. This leads to a total of $t_i + 1 + \delta_i = k_i + 1$ points have different abscissas. We can interpolate those $k_i + 1$ points as in Sect. 2.1 and denote $L_{k_i+1}(X)$ the corresponding Lagrange polynomial obtained at Step 2.3 of SecretReconstruction. As $F_i(X)$ passes through the same points and has degree at most k_i (it is exactly k_i if the secret $S_{i\,k_i}$ is different from 0) we get: $L_{k_i+1}(X) = F_i(X)$. Thus the k_i coefficients returned at Step 2.4 of SecretReconstruction are the k_i secrets of the (t_i, n)-threshold scheme: $S_{i\,1}, \ldots, S_{i\,k_i}$. □

Theorem 3. *Our VMTMSS achieves secrecy.*

Proof. Let i be any integer in $\{1, \ldots, \ell\}$. Assume that $t_i - 1$ participants pool their pseudo-shares together and use public knowledge from tables T and T_{H}. The participants are denoted $P_{j_1}, \ldots, P_{j_{t_i-1}}$. Since H is a collision resistant hash function, H is a one-way function. Therefore with overwhelming probability, the only information the colluders learn from table T_{H} is the pseudo-shares of the non-colluding members are different from theirs. Nevertheless this fact was already known to each of the n participants due to Step 1 of SetUp, property P4 and (2). So table T_{H} does not give any extra-information to the colluders with overwhelming probability. We have two cases to consider.

First Case: $\delta_i = 0$**.** The colluders have to determine the $t_i + 1$ coefficients of $F_i(X)$ (Step 3.2 of ShareConstruction). Using the same technique as in the proof of Theorem 2, they can obtain t_i points $F_i(X)$ goes through from their pseudo-shares and the point $(\mathcal{A}, \mathcal{F}(\mathcal{A}))$ from T. Consider the set:

$$E := \{(f_i(r_i, s_{i\,j}), F_i(f_i(r_i, s_{i\,j}))) : j \notin \{j_1, \ldots, j_{t_i-1}\}\}$$

The elements of E represent the points owned by the non-colluding members. It should be noticed that the n values $F_i(f_i(r_i, s_{i\,1})), \ldots, F_i(f_i(r_i, s_{i\,n}))$ are known to each group participant since they can be obtained by reductions modulo p_i from elements $\mathcal{F}(\mathcal{P}_1), \ldots, \mathcal{F}(\mathcal{P}_n)$ contained in T. We will see that the probability the colluders can construct an element of E is negligible as a function of the length of p_i.

Due to Property P4 of the function f_i the colluders know, with overwhelming probability, that the abscissas of the elements of E belong to:

$$f_i(D_i) \setminus \left\{f_i(r_i, s_{i\,j_1}), \ldots, f_i(r_i, s_{i\,j_{t_i-1}}), \mathcal{A} \bmod p_i\right\}$$

We would like to draw the reader's attention to the following point. Once $F_i(f_i(r_i, s_{i\,\mu}))$ is given, there may be more than one value x such that $F_i(x) = F_i(f_i(r_i, s_{i\,\mu}))$. In the worst case we can have up to $n - t_i + 1$ such values for x which happens when all the ordinates of the elements of E are equal. Thus:

$$\text{Prob}((x, F_i(f_i(r_i, s_{i\,\mu}))) \in E, x \text{ is built by the colluders}) \leq \frac{n+1}{|f_i(D_i)| - n}$$

Second Case: $\delta_i > 0$**.** The colluders have to determine the $k_i + 1$ coefficients of $F_i(X)$ (Step 3.2 of ShareConstruction). As before, they can obtain $t_i + \delta_i$ points $F_i(X)$ goes through from their pseudo-shares and the $\delta_i + 1$ points from T. As previously we get:

$$\text{Prob}((x, F_i(f_i(r_i, s_{i\,\mu}))) \in E, x \text{ is built by the colluders}) \leq \frac{n+1}{|f_i(D_i)| - k_i}$$

Without loss of generality, we can assume that the range of f_i represents a non-negligible part of $\mathbb{Z}/p_i\mathbb{Z}$. At the same time, we can consider that the group size n and k_i is small in comparison to p_i so that there exists C_i, independent from p_i, such that, in both cases, we have:

$$\text{Prob}((x, F_i(f_i(r_i, s_{i\,\mu}))) \in E, x \text{ is built by the colluders}) \leq \frac{C_i}{p_i}$$

Therefore it is sufficient to pick the smallest of the ℓ primes to be 80 bits long to ensure computational secrecy for our scheme. $\square$

4 Complexity Survey

As claimed in Sect. 1, the computational and storage costs represent key factors to take into account when implementing a protocol as a part of a commercial application. In this part we study the cost of our construction and compare it to the schemes from [2, 19]. In this section we denote M the product of the ℓ prime numbers $p_1, \ldots, p_\ell$. We assume that picking random elements from the sets $\mathbb{Z}/p_1\mathbb{Z}, \ldots, \mathbb{Z}/p_\ell\mathbb{Z}$ has a negligible computational cost.

4.1 Cost of Our Construction

Computational Cost at the Dealer. Based on Sect. 2.2, SetUp can be executed in $O(n\,\ell \log_2^2 M)$ bit operations.

ShareConstruction performs $n + D + 3$ calls to Garner's algorithm which results in $O((n+D)\,\ell\, \log_2^2 M)$ bit operations. In addition there are $n+1$ polynomial evaluations over $\mathbb{Z}/M\mathbb{Z}$. Using Horner's rule each of them can be done via D additions and D multiplications in $\mathbb{Z}/M\mathbb{Z}$. Based on Sect. 2.1, this represents a total of $O(n\,D\, \log_2^2 M)$ bit operations. There are also δ_i polynomial evaluations over $\mathbb{Z}/p_i\mathbb{Z}$. If we denote $\Delta := \max\limits_{i \in \{1,\ldots,\ell\}} \delta_i$ then the $\delta_1 + \cdots + \delta_\ell$ polynomial evaluations cost $O\left(\Delta\, D \log_2^2 \left(\max\limits_{i\in\{1,\ldots,\ell\}} p_i\right)\right)$ bit operations. Since each prime number p_i is less than

M, the total cost of ShareConstruction is $O([D\,(\ell + n + \Delta) + n\,\ell]\,\log_2^2 M)$ bit operations. Furthermore the collision resistant hash function H is run $n\,\ell$ times while each keyed one-way function f_i is run $n + \delta_i$ times.

Computational Cost at the Combiner. Notice that the cost of SecretReconstruction depends on the threshold value t_i. We have $t_i + 2$ reductions modulo p_i of elements $\mathbb{Z}/M\mathbb{Z}$. This can be done using Euclid's divisions in $O(t_i\,(\log_2 M\,\log p_i))$ bit operations. In addition an interpolating polynomial passing through $t_i + 1 + \delta_i$ points is to be build over $\mathbb{Z}/p_i\mathbb{Z}$. We know from Sect. 2.1 this can be achieved in $O((t_i + \delta_i)^2\,\log_2^2 p_i)$ bit operation. Since $p_i \leq M$, we deduce that SecretReconstruction runs in $O((t_i + \delta_i)^2\,\log_2 M\,\log_2 p_i)$ bit operations.

Storage of Public Elements. Denote size(x) the number of bits used to represent the natural integer x. We have size$(x) = \lfloor \log_2 x \rfloor + 1$. We define $\rho := \sum_{i=1}^{\ell} \delta_i\,\mathrm{size}(p_i)$ and $\rho' := \sum_{i=1}^{\ell} \mathrm{size}(p_i)$. We also denote $\mathcal{H}$ the bitsize of a digest produced by the collision resistant hash function. First, storing T_{H} requires $n\,\ell\,\mathcal{H}$ bits. Second, T contains $n + 3$ elements from $\mathbb{Z}/M\mathbb{Z}$ and $2\,\delta_i$ elements from $\mathbb{Z}/p_i\mathbb{Z}$ for each $i \in \{1, \ldots, \ell\}$. Thus the size of T is $(n + 3)\,\mathrm{size}(M) + 2\,\rho$ bits. As a consequence the size of public elements represents a total of $n\,(\ell\,\mathcal{H} + \mathrm{size}(M)) + 3\,\mathrm{size}(M) + 2\,\rho$ bits. Notice, however, that the sender must buffer all the elements $s_{1\,1}, \ldots, s_{\ell\,n}$ from Step 1 of SetUp which represents $n\,\rho'$ bits.

4.2 Efficiency Comparison

The parameters of the schemes are depicted in Table 1. Notice that the construction by Chan and Chang does not allow flexibility in the number of secrets to be shared. Indeed when we iterate that construction λ times (with the same threshold values) then the total number of secrets has to be $\lambda\,\ell$. Therefore we restrict our comparison to the scheme by Shao and Cao as it enables to choose the number of secrets per threshold independently from the total number of thresholds. Remark that our construction can be seen as extension of Chan and Chang's approach providing flexibility. To have an accurate survey, we assume that Shao and Cao's construction is iterated ℓ times (one iteration per family of k_i secrets). The results of our comparison are summarized in Table 2.

The reader can notice that ρ' is slightly larger than size(M) so, a priori, our technique does not provide any significant size benefit from ℓ iterations of Shao and Cao's construction. As noticed in [2], however, the latter approach requires each participant to keep multiple shares which can create a share management problem. In our construc-

Table 1. Parameters of the Three VMTMSS

	Our Scheme	Chan-Chang's Scheme [2]	Shao-Cao's Scheme [19]
Thresholds	ℓ	ℓ	1
Secrets per Threshold	k_i	1	k
Size Private Values	size(M) bits	size(p) bits	size(p) bits

Table 2. Computational Complexity of the Three VMTMSS

	Our Scheme	Shao-Cao's Scheme [19]
Size Private Values	size(M) bits	ρ' bits
Set-up Phase	$n\,\ell$ random elements n calls to Garner	$n\,\ell$ random elements
Share Creation Process	δ_i pol. eval. in each $\mathbb{Z}/p_i\mathbb{Z}$ $n+1$ pol. eval. in $\mathbb{Z}/M\mathbb{Z}$ $n\,\ell$ calls to H $n+\delta_i$ calls to each f_i $n+D+3$ calls to Garner	$n+\delta_i$ pol. eval. in each $\mathbb{Z}/p_i\mathbb{Z}$ $\max(t_i,k_i)$ exp. in each $\mathbb{Z}/p_i\mathbb{Z}$ n calls to each f_i
Pseudo-Share Validity Verification	1 call to H	$\max(t_i,k_i)$ exp. in each $\mathbb{Z}/p_i\mathbb{Z}$ $\max(t_i,k_i)$ exp. in $\mathbb{Z}/\frac{p_i-1}{2}\mathbb{Z}$ $\max(t_i,k_i)$ mult. in $\mathbb{Z}/p_i\mathbb{Z}$
Secret Recovery	1 polynomial reconstruction t_i+2 reductions modulo p_i	1 polynomial reconstruction
Storage Public Elements	$n\,(\ell\,\mathcal{H}+\text{size}(M))+3\,\text{size}(M)+2\,\rho$ bits	$(n+1)\,\rho'+2\,\rho+\sum_{i=1}^{\ell} t_i\,\text{size}(p_i)$ bits
Storage Sender	$n\,\rho'$ bits	$n\,\rho'$ bits

tion, each participant holds a single "master" share which can be used to recreate the share for each (t_i,n)-scheme. We now have two points to consider.

First, the pseudo-share verification process from [19] is expensive. Indeed verifying a single pseudo-share roughly costs $2\,\max(t_i,k_i)$ exponentiations in $\mathbb{Z}/p_i\mathbb{Z}$. Even if each of them can be performed in $O(\log_2^3 p_i)$ bit operations using the fast exponentiation. algorithm [17], the coefficient $\max(t_i,k_i)$ is prohibitive for large thresholds t_i. In addition, when the communication channel is under attack of malicious users flooding the combiner with incorrect values, the coefficient $\max(t_i,k_i)$ may result in successful denial-of-service attacks as the computational resources needed to identify correct shares amongst forgeries become too large. This problem does not happen with our construction as only a single hash as to be computed to validate/discard a share. Notice that each participant first needs to perform 2 reductions modulo p_i and 1 call to f_i to construct his pseudo-share from his secret value and the public element $\mathcal{R}$. However this is at the cost of running $2\,n+D+3$ times Garner's algorithm at the dealer during the set-up and share construction phases.

Second, our pseudo-share verification process requires $n\ell$ hashes to be published as table T_{H}. If we use SHA-256 as collision resistant hash function then T_{H} is represented over $256\,n\,\ell$ bits. On the other hand, the construction by Shao and Cao is secure provided that the discrete logarithm problem over each $\mathbb{Z}/p_i\mathbb{Z}$ is intractable. For achieve security, it is suggested to use 1024-bit moduli or larger [16]. If we assume that the different thresholds are roughly equal to the same value t then the coefficient

$\sum_{i=1}^{\ell} t_i \,\mathrm{size}(p_i)$ is approximately $1024\,\ell\,t$ bits. Therefore the storage of our public elements less expensive as soon as $t \geq \frac{n}{4}$, i.e. the construction by Shao and Cao provides better space efficiency only for small threshold values.

5 Conclusion

In this paper, we generalized the approaches from [2, 19] by designing a multiple time verifiable secret sharing scheme allowing several secrets to be shared per threshold value. As in [19], our construction allows any number of secrets to be shared per threshold value. In addition, we showed that our pseudo-share verification process was much faster than in [19] while the storage requirements were smaller. We would like to point three facts. First, we assumed that the threshold values were different (see Sect. 3). Nevertheless our techniques could also be employed if some threshold t_i is used τ_i times provided that different primes $p_{i\,1}, \ldots, p_{i\,\tau_i}$ are used respectively. Second, the security of our scheme is based on the random oracle model for the collision resistant hash function H. Most hash functions used in practice are considered heuristically collision resistant. Recently several such functions were successfully attacked [21, 22, 23, 24, 26]. In order to maintain the security of our protocol, we suggest to use a hash function whose security has be proved to be linked to a computationally difficult problem such as Very Smooth Hash [5] or Gibson's discrete logarithm-based hash function [7]. Nevertheless this may result into larger digests or increased running time. Finally the main drawback of our construction is that we are only able to deal with threshold schemes and our approaches cannot be directly generalized to non-threshold access structures.

Acknowledgement

The authors are grateful to the anonymous reviewers for their comments to improve the quality of this paper. This work was supported by the Australian Research Council under ARC Discovery Projects DP0558773, DP0665035 and DP0663452. This work was supported in part by the National Natural Science Foundation of China Grant 60553001 and the National Basic Research Program of China Grant 2007CB807900, 2007CB807901. Christophe Tartary did most of this work while at Macquarie University where his research was supported by an iMURS scholarship. The research of Huaxiong Wang is partially supported by the Minitry of Education of Singapore under grant T206B2204.

References

[1] Blakley, G.R.: Safeguarding cryptographic keys. In: AFIPS 1979 National Computer Conference, pp. 313–317. AFIPS Press (1979)
[2] Chan, C.-W., Chang, C.-C.: A scheme for threshold multi-secret sharing. Applied Mathematics and Computation 166(1), 1–14 (2005)
[3] Chang, T.-Y., Hwang, M.-S., Yang, W.-P.: An improvement on the Lin-Wu (t, n) threshold verifiable multi-secret sharing scheme. Applied Mathematics and Computation 163(1), 169 (2005)

[4] Chien, H.-Y., Jan, J.-K., Tseng, Y.-M.: A practical (t, n) multi-secret sharing. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E83-A(12), 2762–2765 (2000)

[5] Contini, S., Lenstra, A.K., Steinfeld, R.: VSH: an efficient and provable collision resistant hash collision. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 165–182. Springer, Heidelberg (2006)

[6] Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 120–127. Springer, Heidelberg (1988)

[7] Gibson, J.K.: Discrete logarithm hash function that is collision free and one way. IEE Proceedings - Computers and Digital Techniques 138(6), 407–410 (1991)

[8] Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. The Johns Hopkins University Press (1996)

[9] Gong, L.: New protocols for third-party-based authentication and secure broadcast. In: 2^{nd} ACM Conference on Computer and Communications Security, pp. 176–183. ACM Press, New York (1994)

[10] Harn, L.: Efficient sharing (broadcast) of multiple secrets. IEE Proceedings - Computers and Digital Techniques 142(3), 237–240 (1995)

[11] He, J., Dawson, E.: Multisecret sharing scheme based one-way function. IEE Electronic Letters 31(2), 93–95 (1995)

[12] Jackson, W.-A., Martin, K.M., O'Keefe, C.M.: On sharing many secrets (extended abstract). In: Safavi-Naini, R., Pieprzyk, J.P. (eds.) ASIACRYPT 1994. LNCS, vol. 917, pp. 42–54. Springer, Heidelberg (1995)

[13] Karnin, E.D., Greene, J.W., Hellman, M.E.: On secret sharing systems. IEEE Transactions on Information Theory 29(1), 35–41 (1983)

[14] Krawczyk, H.: Secret sharing made short. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 136–146. Springer, Heidelberg (1994)

[15] Lin, T.-Y., Wu, T.-C.: (t, n) threshold verifiable multisecret sharing scheme based on factorisation intractability and discrete logarithm modulo a composite problems. IEE Proceedings - Computers and Digital Techniques 146(5), 264–268 (1999)

[16] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)

[17] Pieprzyk, J., Hardjono, T., Seberry, J.: Fundamentals of Computer Security. Springer, Heidelberg (2003)

[18] Shamir, A.: How to share a secret. Communication of the ACM 22(11), 612–613 (1979)

[19] Shao, J., Cao, Z.: A new efficient (t, n) verifiable multi-secret sharing (VMSS) based on YCH scheme. Applied Mathematics and Computation 168(1), 135–140 (2005)

[20] Stinson, D.R.: Cryptography: Theory and Practice, 3rd edn. Chapman & Hall/CRC (2006)

[21] Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)

[22] Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)

[23] Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

[24] Wang, X., Yu, H., Yin, Y.L.: Efficient collision search attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)

[25] Yang, C.-C., Chang, T.-Y., Hwang, M.-S.: A (t, n) multi-secret sharing scheme. Applied Mathematics and Computation 151(2), 483–490 (2004)

[26] Yu, H., Wang, G., Zhang, G., Wang, X.: The second-preimage attack on MD4. In: Desmedt, Y.G., Wang, H., Mu, Y., Li, Y. (eds.) CANS 2005. LNCS, vol. 3810, pp. 1–12. Springer, Heidelberg (2005)

Key Management Based on Hierarchical Secret Sharing in Ad-Hoc Networks

Chuangui Ma[⋆] and Rui Cheng

Department of Applied Mathematics,
Zhengzhou Science and Technology Institute,
Zhengzhou, China
chuanguima@sina.com, arrui@163.com

Abstract. A hierarchical key sharing model in ad-hoc is presented and a key management protocol based on this model is designed. Nodes were divided into two parts in the network, server nodes and ordinary nodes. The shares hold by server nodes include more information than ordinary ones. Key reconstruction is implemented by collaboration of k nodes(including at least k_1 server nodes). Key share distribution, key reconstruction, key share refreshing and recovery are discussed in this paper. The presented protocols improve the availability and enhance the security compared with the existing partially and fully distributed schemes respectively.

Keywords: ad-hoc network, verifiable secret sharing, hierarchical secret sharing, key management.

1 Introduction

Ad-hoc network can provide the users anytime and anywhere communications, which makes it become an ideal candidate for many applications both on military and civilian field. However, the use of wireless links gives chances to attacks ranging form passive eavesdropping to active message replay, impersonation and distortion. Nodes are easier to be compromised due to relatively poor physical protection. Security measures should be adopted to protect the ad-hoc communications.

Key management is a foundation for security service. Most of the security mechanisms such as digital signature, public key certification, identity authentication require some kind of cryptographic keys. As Menezes et al proposed in [1], the purpose of key management is to:

– Initialize system users within a domain.
– Generate, distribute and install keying material.
– Control the use of keying material.
– Update, revoke and destroy keying material.
– Store, recover and archive keying material.

[⋆] The work was partially supported by hi-tech research and development programm of China (2007AA01Z431).

Traditional centralized key management is not appropriate to ad- hoc network. No single node is trustworthy in an ad-hoc network because of low physical security and availability.

There are several researches on key management in ad hoc network[2]. The solution proposed by Hubaux[3] provides a public key management solution similar to PGP[4] in the sense that certificates are issued by the users themselves without the involvement of any certification authority. It is suitable for long-term networks. Basagni[5] provides a distributed key management system based on symmetric encryption. The solution provides group authentication, message integrity and confidentiality. Balfanz's scheme [6] is based on what the authors call demonstrative identification and is similar to the concept of imprinting introduced by Stanjo and Anderson[7]. It requires that the nodes be in a close proximity of each other during the initial bootstrapping.

The solution proposed by Zhou[8] distribute the key services to n special nodes called servers. Any $k+1$ of these nodes collaborate to complete the functions of CA. There are many drawbacks in this scheme. First, how can a node find enough servers in need of certification service as there are circumstances while all his neighbors are not server nodes. Secondly, Zhou did not address the problem how to maintain the availability of the service when some server nodes leave . In Luo's scheme[9] , every node hold a share of the secret key. This approach is vulnerable since adversaries can compromise arbitrary $k+1$ nodes to reveal the service key.

While distributed key management fit appropriately to the characteristic of no centralized authority in ad-hoc network, it should be improved to enhance the security of key management. Based on hierarchical threshold secret sharing, we proposed a fully distributed key management scheme for ad-hoc networks. In our scheme, nodes are partitioned into two levels: server nodes and ordinary nodes. Server nodes hold key shares of more information than ordinary nodes hold. Any l_1 server nodes and l_2 ordinary nodes ($l_1 \geq k_1$, $l_2 \geq k - l_1$, k_1, k are threshold number) cooperate to reconstruct the system's secret key. This improves the availability of the key service in ad-hoc compared with Zhou's. Nodes do not need to find $k+1$ server nodes to get service(in some cases it is not easy to communicate with so many server nodes). It only needs to find k_1 server nodes and $k - k_1$ ordinary nodes. While this scheme reduce the risk of distributing key shares to all nodes as Luo's scheme in which compromising of any k nodes will leak the system's secret key. We assume that server nodes are more reliable and securer than ordinary nodes.

Hierarchical key management scheme is beneficial in some applications. In military environment, nodes could be composed of unmanned aerial vehicles (UAVs), trucks tanks, and soldiers. Obviously, UAVs have strong communication and computation power, while soldiers have limited power. All of the fighting units on land are capable of being broken down or compromised. So they should be in different level in key management. Another example is when stockholders of a company are attending a meeting communicating through their laptops or PDAs and need to make an investment decision. Some of the attenders are employees and some of them can be department managers or members of directorate. They all hold a share of the key of the account . In a word, they ought to hold key share of different level according to their principalship and so they can play a different part in the decision.

CONTRIBUTION. We introduce hierarchical module in ad-hoc and present a hierarchical secret sharing scheme.The major properties are:

- we improve the utility of key service and do not reduce the security.
- the scheme is proactive.
- Key management is distributed to all nodes. No previously proposed sharing schemes can provide all these property together.

PAPER STRUCTURE. In the following section we compare related sharing schemes. Section 3 presents system model and notations used in this paper, Section 4 details our generic constrction.Finally we conclude our work and discuss some future work.

2 Hierarchical Secret Sharing

Hierarchy secret sharing was proposed by Tass. Our hierarchy secret sharing scheme use the concept of hierarchy access structure and a two dimensional polynomial for secret sharing and verifying.

Definition 1. *Let U be a set of n participants and assume that $U = U_1 \bigcup U_2$ and $U_1 \bigcap U_2 = \emptyset$. Let $\mathbf{k} = \{k_1, k_2\}$, $0 < k_1 < k_2$. Then the $(\mathbf{k}, n)$-hierarchical threshold problem is to assign each participant $u \in U$ a share of a given secret s such that the access structure is*

$$\Gamma = \{V \subset U : |V \bigcap (\bigcup_{j=1}^{i} U_j)| \geq k_i, for \quad i = 1, 2\} \tag{1}$$

If $\sigma(u)$ stands for the shares assigned to $u \in U$, and for any $V \subset U$,$\sigma(V) = \{\sigma(u) : u \in V\}$, then

$$H(s|\sigma(V)) = 0 \qquad \forall V \in \Gamma \, (accessibility) \tag{2}$$

while

$$H(s|\sigma(V)) = H(s) \quad \forall V \notin \Gamma \, (perfect\ security) \tag{3}$$

Hierarchical $(\mathbf{k}, n)$ secret sharing scheme: Let $\mathbb{F}_p$ be a finite field of large size and p be a prime number. Assume $\mathbf{k} = \{k_1, k_2\}$ and $k = k_2$ is the overall number of participants that are required for the recovery of the secret.

Let $V = \{v_1, \cdots, v_{|V|}\} \subset U$, assume that

$$\begin{aligned} v_1, \cdots, v_{l_1} &\in U_1 \\ v_{l_1+1}, \cdots, v_{l_2} &\in U_2 \end{aligned} \tag{4}$$

Then V is authorized if and only if $l_i \geq k_i$ for $i = 1, 2$. Let

$$r(x) = (1, x, \cdots, x^{k-1})$$

$$r^{(k_1)}(x) = \frac{d}{d^{(k_1)}}(1, x, \cdots, x^{k-1})$$

$\mathbf{a} = (a_0, \cdots, a_{k-1})^T$, then the share of each is $\sigma(u) = r(u) \cdot \mathbf{a}$ for $u(u \in U_1)$(each participant is identified by u, $u \in \mathbb{F}_p$) and $\sigma(u) = r^{(k_1)}(u) \cdot \mathbf{a}$ for $u(u \in U_2)$. The recovery of the secret is to solve the unknown vector $\mathbf{a}$ in $M_v\mathbf{a} = \sigma$, where

$$M_v = (r(v_1), \cdots, r(v_{l_1}); r^{(k_1)}(v_{l_1+1}), \cdots, r^{(k_1)}(v_{l_2}))^T \tag{5}$$

$$\sigma = (\sigma(v_1), \sigma(v_2), \cdots, \sigma(v_{l_2}))^T$$

The above Birkhoff interpolation does not have unique solution in general. The following theorem describe when it has an unique solution and how secret sharing scheme satisfy accessibility and perfect security(for details see [10]).

Theorem 1. *If for any minimal authorized subset $V \subset \Gamma$, $M_v \neq 0$ in $\mathbb{F}_p$.Then conditions 2 and 3 hold.*

Theorem 2. *Let $(\mathbf{k}, n)$ be a hierarchical secret sharing scheme. Assume that the participants in U were assigned identities in $\mathbb{F}_p$ in a monotone manner, let $N = \{maxu | u \in U\}$. Assume that $2^{-k} \cdot (k+1)^{(k+1)/2} \cdot N^{(k-1)k/2} < p$, then the hierarchical secret scheme satisfies conditions 2 and 3.*

Tassa's secret sharing scheme based on Birkoff interpolation solved the problem of hierarchical sharing, but it can't prevent cheating when dealer sends wrong shares to participants in share distribution or some participants provide wrong shares in key reconstruction. Since the wireless link is prone to be attacked, we should ensure that every node receive correct shares. This can be done by using verifiable Feldman's VSS[11] based on ECC.

3 System Models

Here we consider n nodes ad-hoc wireless network composed of n_1 server nodes and n_2 ordinary nodes. Assume that the capabilities of nodes in the network are diverse in terms of power source, CPU performance and memory size etc. Server nodes should be equipped superiorly and have strong capability. Also suppose that all nodes are capable of performing the necessary public key computation and they are equipped with PKI certificates by an off-line trusted authority. so they can authenticate each other and establish secure links. Let $u_m, u_m \in \mathbb{F}_p$ present the identity of node m. Then the set of n nodes could be considered as $U = U_1 \bigcup U_2$ where U_1 is the subset of server nodes and U_2 is the subset of ordinary nodes. In order to get a unique solution when the nodes in an authorized subsets pull out their shares, the conditions in Theorem 2.2 should be satisfied.

Our key management is designed for a long time existing network. We consider a moveable adversary who can compromise and control at most $k_1 - 1$ nodes in a certain period of time. To defend against the adversary, key shares should be proactively refreshed. Thus the operation time of system is partitioned into period. The execution time of refreshed protocols is between two sharing intervals At the end of a interval and beginning of the next interval is share refreshing time. When refreshing protocols finished, according to proactive secret sharing's definition[12] the shares hold by the adversaries would be of no use in the next period.

4 Hierarchical Key Management

Key management can be grouped as certificate related service and system maintenance service. The system maintenance service includes incorporating joining nodes into the key management, i.e. providing them with shares of the system's secret key. It also includes proactively updating the shares of the system's private key to protect it from being compromised. Here we emphasize on the steps required to setup and maintain the distributed key management. At the beginning, dealer initialize each node off-line using hierarchical secret sharing algorithm in 4.1. In the running of ad-hoc network, authorized subset plays the role of dealer. Let E be a secure elliptic curve over $\mathbb{F}_p$ and Q be a point over E. Assume ECDLP on E is hard.

4.1 Key Distribution

During the period of share refreshing, key distribution protocol is executed by a subset of server nodes independently.

Step 1: Dealer select a random polynomial $F(x, y) \in \mathbb{F}_p[X]$

$$F(x, y) = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} f_{ij} x^i y^j$$

where $f_{0,0} = s,\ f_{k-1,k-1} \neq 0$

Step 2: Dealer compute a verification matrix F

$$\mathbf{F} = (f_{ij}Q)_{k\times k} = (F_{ij})_{k\times k}$$

Step 3: Dealer sends shares to all nodes:

$$\forall u_i, dealer \rightarrow u_i, \langle dist, u_d, Cert, \mathbf{F}, \varepsilon(\mathbf{A_i}, \mathbf{B_i})\rangle_k$$

a) For $u_i \in U_1$,

$$A_i(y) = F(u_i, y) = \sum_{t=0}^{k-1} a_{it} y^t, B_i(x) = F(x, u_i) = \sum_{t=0}^{k-1} b_{it} x^t$$

b) For $u_i \in U_2$,

$$A_i(y) = F_x^{(k_1)}(u_i, y) = \sum_{t=0}^{k-1} a_{it} y^t, B_i(x) = F_y^{(k_1)}(x, u_i) = \sum_{t=0}^{k-1} b_{it} x^t$$

where $\mathbf{A_i} = (a_{i0}, a_{i1}, \ldots, a_{i,k-1})$ and $\mathbf{B_i} = (b_{i0}, b_{i1}, \ldots, b_{i,k-1})$ are the coefficient vector of $A_i(y)$ and $B_i(x)$ respectively, $s_i = A_i(0)$ is u_i's key share.

Step 4: Either server nodes or ordinary nodes u_i verify their shares by checking that if:

$$a_{ij} Q = \sum_{t=0}^{k-1} F_{tj}(u_i)^t, b_{ij} Q = \sum_{t=0}^{k-1} F_{jt}(u_i)^t$$

for $u_i \in U_1$, $j = 0, 1, \cdots, k-1$

$$a_{ij} Q = \sum_{t=k_1}^{k-1} \binom{t}{k_1} k_1! F_{tj}(u_i)^{t-k_1}, b_{ij} Q = \sum_{t=k_1}^{k-1} \binom{t}{k_1} k_1! F_{jt}(u_i)^{t-k_1}$$

for $u_i \in U_2$, $j = 0, 1, \cdots, k-1-k_1$.

Step 5 : If verification fails, u_i will ask dealer to send its share again:

$$u_i \rightarrow dealer, \langle share - request, u_i, Cert \rangle_k$$

Step 6: If u_i receives wrong share again, it will make an accusation against dealer.

4.2 Key Reconstruction

This protocol is to construct the system's key for an authorized subset.

Step 1: The server node u_i require key reconstruction will send the message:

$$u_i \rightarrow \forall u_j, \langle Reconstruct, u_i, Cert \rangle_k$$

Step 2: Every node u_j, when receiving the *Reconstruct* message, send its share to u_i:

$$\forall u_j \rightarrow u_i, \langle Reconstructed, u_j, Cert, \varepsilon(s_j) \rangle_k$$

Step 3: Upon receiving the share s_j from u_j, u_i verify if

$$s_j Q = \sum_{t=0}^{k-1} F_{t0} u_j^t$$

for $u_j \in U_1$ or

$$s_j Q = \sum_{t=k_1}^{k-1} \binom{t}{k_1} (k_1)! F_{t0} u_j^{t-k_1}$$

for $u_j \in U_2$

Step 4: Node u_i put together all these verified shares, and figure out an authorized subset in hierarchical access structure to reconstruct the secret.

Theorem 3. *Any authorized subset of nodes $V \subset U$ satisfies condition accessibility (2) and (4).*

4.3 Share Refreshing

Share refreshing is needed after a period of time to resist mobile adversary. At the end of a period, all server nodes are likely to flood a refreshing message to all nodes. Then it locate k server nodes (assume $P = \{u_1, u_2, \cdots, u_k\}$) to refresh the key share of every node. The problem is that any k of the server nodes may include compromised nodes. These nodes may not follow the protocol correctly. To overcome this circumstance these k nodes should exchange their certificates before share refreshing and check if the certificates is on the CRL. When interactive verification finished, these k nodes start share refreshing protocol.

Step 1: Each $u_i \in P$ create a two dimensional polynomial $F_i(x, y)$ satisfied $F_i(0, 0) = 0$ and apply share distribution protocol as in 4.1.

Step 2: For $u_j \in U$, when all the subshares send by $u_i \in P$ pass the verification, u_j add the new subshares to old share. Then $u_j's$ new share is:

$$A'_j(y) = A_j(y) + \sum_{i=1}^{k} A_{ij}(y), u_j$$

$$B'_j(x) = B_j(x) + \sum_{i=1}^{k} B_{ij}(x), u_j \tag{6}$$

$$\mathbf{F}' = (F_{ts} + \sum_{i=1}^{k} F_{its})_{k \times k}$$

In this section we only consider the server nodes to be responsible for share refreshing. In fact, any k nodes including server nodes and ordinary nodes can cooperate to complete it.

Theorem 4. *Participants in the authorized set can reconstruct the secret with these new shares.*

4.4 Share Recovery

Node can lost his share or ruin it due to disk crash or intrusion of bad program. A more general case is that a new node call for joining in the network. Node need to restore its shares or get its shares by asking nodes of an authorized subset for help.

Step 1: Node u_i broadcast a message to all nodes:

$$\forall u_j, u_i \rightarrow u_j$$

$$\langle Recover - request, u_i, Cert \rangle_k$$

Step 2: When node u_j receiving the message, u_j respond to node u_i's request:

a) u_i is an ordinary node:

$$u_j \rightarrow u_i, \langle Recover - response, u_j, Cert, \mathbf{F}, \varepsilon(A_j^{(k_1)}(u_i), B_j^{(k_1)}(u_i))\rangle_k \quad (7)$$

b) u_i is a server node:

$$u_j \rightarrow u_i, \langle Recover - response, u_j, Cert, \mathbf{F}, \varepsilon(A_j(u_i), B_j(u_i))\rangle_k \quad (8)$$

Step 3: Node u_i verify u_j's share according to u_i's and u_j's identity:

a) u_i is a server node and u_j is an ordinary node:

$$A_j(u_i)Q = \sum_{t=k_1}^{k-1}\sum_{s=0}^{k-1} F_{ts}\binom{t}{k_1}(k_1)!(u_j)^{t-k_1}(u_i)^s \quad (9)$$

$$B_j(u_i)Q = \sum_{t=0}^{k-1}\sum_{s=k_1}^{k-1} F_{ts}\binom{s}{k_1}(k_1)!(u_i)^t(u_j)^{s-k_1} \quad (10)$$

b) u_i is an ordinary node and u_j is a server node :

$$A_j^{(k_1)}(u_i)Q = \sum_{t=0}^{k-1}\sum_{s=k_1}^{k-1} F_{ts}\binom{s}{k_1}(k_1)!(u_j)^t(u_i)^{s-k_1} \quad (11)$$

$$B_j^{(k_1)}(u_i)Q = \sum_{t=k_1}^{k-1}\sum_{s=0}^{k-1} F_{ts}\binom{t}{k_1}(k_1)!(u_i)^{t-k_1}(u_j)^s \quad (12)$$

c) Both u_i and u_j are server nodes:

$$A_j(u_i) = \sum_{t=0}^{k-1}\sum_{s=0}^{k-1} F_{ts}(u_j)^t(u_i)^s$$

$$B_j(u_i) = \sum_{t=0}^{k-1}\sum_{s=0}^{k-1} F_{ts}(u_i)^t(u_j)^s$$

d) Both u_i and u_j are ordinary nodes:

$$A_j^{(k_1)}(u_i) = \sum_{t=k_1}^{k-1}\sum_{s=k_1}^{k-1}\binom{t}{k_1}\binom{s}{k_1}F_{ts}((k_1)!)^2(u_j)^{t-k_1}(u_i)^{s-k_1} \quad (13)$$

$$B_j^{(k_1)}(u_i) = \sum_{t=k_1}^{k-1}\sum_{s=k_1}^{k-1}\binom{t}{k_1}\binom{s}{k_1}F_{ts}((k_1)!)^2(u_i)^{t-k_1}(u_j)^{s-k_1} \quad (14)$$

Step 4: Upon receiving enough shares, u_i could recovery its share. Suppose that $V = \{u_1, \cdots, u_{l_1}, u_{l_1+1}, u_{l_2}\}$ is the subset of nodes that response to him.If V is an authorized subset,then we may obtain shares satisfying equations(15) and (18):

$$M_v \cdot \mathbf{A_i} = \sigma_{i_A} \tag{15}$$

where $\mathbf{A_i} = (a_{i0}, a_{i1}, \cdots, a_{i,k-1})$,

$$\sigma_{i_A} = (B_1(u_i), \cdots, B_{l_1}(u_i), B_{l_1+1}(u_i), \cdots, B_{l_2}(u_i))u_i \in U_1 \tag{16}$$

$$\sigma_{i_A} = (B_1^{(k_1)}(u_i), \cdots, B_{l_1}^{(k_1)}(u_i), B_{l_1+1}^{(k_1)}(u_i), \cdots, B_{l_2}^{(k_1)}(u_i))u_i \in U_2 \tag{17}$$

$$r(v) \cdot \mathbf{B_i} = \sigma_{i_B} \tag{18}$$

where $\mathbf{B_i} = (b_{i0}, b_{i1}, \cdots, b_{i,k-1})$,

$$\sigma_{i_B} = (A_1(u_i), \cdots, A_{l_1}(u_i), A_{l_1+1}(u_i), \cdots, A_{l_2}(u_i))u_i \in U_1 \tag{19}$$

$$\sigma_{i_B} = (A_1^{(k_1)}(u_i), \cdots, A_{l_1}^{(k_1)}(u_i), A_{l_1+1}^{(k_1)}(u_i), \cdots, A_{l_2}^{(k_1)}(u_i))u_i \in U_2 \tag{20}$$

Theorem 5. *Using shares received from an authorized set, u_i can recovery its polynomial $A_i(y)$ and $B_i(x)$.*

5 Conclusion

This paper investigate the problem of key management in ad-hoc network. We provided some protocols for key distribution, key share refreshing and share recovery with a traditional hierarchical sharing scheme. Since our module depends on different security level nodes,we improve the utility of key service and enhance the security of our schemes. The other aspect of key management as certificate related services including certificate renewal and revocation are not discussed here. It will be studied in our next paper.As for the feasibility for hierarchical key management scheme,it still be challenging problem.

Acnowledgments

Both authors wish to thank the anonymous reviewers for their comments.

References

1. Menezes, A., van Oorschot, P., Vantone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
2. Forine, K.: Key Management in Ad Hoc Networks, `http://www.ep.liu.se/exjobb/isy/2002/3322/`
3. Hubaux, J.P., Buttyan, L., Capkun, S.: Self-organized Public-Key Management for Mobile Ad hoc Networks. IEEE Transactions on mpbile Computing 2(1), 1–13 (2003)

4. Garfinkel, S.: PGP: Pretty Good Privacy. OReilly & Associates (1995), ISBN 1-56592-098-8 2003
5. Basagni, S., Herrin, K., Rosti, E., Bruschi, D.: Secure Pebblenets. In: Proceedings of the 2nd ACM international symposium on Mobile ad- hoc networking and computing table of contents ACM 2001, Long Beach, CA, USA, pp. 156–163 (2001)
6. Balfanz, D., Smetters, D.K., Stewart, P., Chi Wong, H.: Talking To Strangers: Authenti- cation in Ad-Hoc Wireless Networks. In: Proceedings of Network and Distributed System Security Symposium 2002, San Diego, CA (February 2002)
7. Stanjo, F., Anderson, R.: The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless Networks, `http://www.cl.cam.ac.uk/~fms27/duckling/duckling.html`
8. Zhou, L., Haas, Z.J.: Securing Ad Hoc Networks. IEEE Networks 13(6) (1999)
9. Luo, H., Lu, S.: Ubiquitous and Robust Authentication Service for Ad Hoc Wireless Networks, it Technical Report 20030, UCLA Computer Science Science Department (2000)
10. Tassa, T.: Hierarchical Threshold Secret Sharing. In: Proceedings of the Theory of Cryptography. Lecture Notes in Computer Science, vol. 2591, pp. 473–493. springer, Heidelberg (2004)
11. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: Proceedings of 28th IEEE symposium on Foundations of Computer Science, pp. 427–437 (1987)
12. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: how to cope with perpetual leakage. In: Advances in Cryptology crypto 1995. Lecture Notes in Computer Science, vol. 953, pp. 339–352. Springer, Heidelberg (1995)
13. Luo, H., Zerfos, P., Kong, J., Lu, S., Zhang, L.: Self-securing Ad Hoc Wireless Networks. In: Seventh International Symposium on IEEE ISCC 2002, pp. 567–574 (2002)

Probabilistic (*n*, *n*) Visual Secret Sharing Scheme for Grayscale Images

Daoshun Wang[1], Xiaobo Li [2], and Feng Yi [1]

[1] Tsinghua National Laboratory for Information Science and Technology (TNlist)
Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, P.R. China
daoshun@mail.tsinghua.edu.cn

[2] Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G 2E8, Canada
li@cs.ualberta.ca

Abstract. Pixel expansion has been a major issue of visual secret sharing (VSS) schemes. A number of probabilistic VSS schemes with minimum pixel expansion have been proposed for black-and-white (binary) secret images. This paper presents a probabilistic (*n*, *n*)-VSS scheme for grayscale images. Its pixel expansion is significantly smaller than that of previous methods. The construction of the shadow images (shares) is based on the Boolean XOR operation.

Keywords: Visual secret sharing; Probabilistic schemes; Pixel expansion.

1 Introduction

Visual secret sharing (VSS) schemes [1] have been proposed to encode a secret image into *n* "shadow" ("share") images to be distributed to *n* participants. The secret can be visually reconstructed only when *k* or more shares are available. No information will be revealed with any *k*-1 or fewer shares. VSS schemes were extended from binary images to color and grayscale images. Several (*n*, *n*)-VSS schemes were designed for special *n* values. Rijmen and Preneel [2] and Yang [3] focus on (2, 2)-VSS schemes and Hou [4] were mainly on (2, 2) and (4, 4) schemes. In a visual cryptography scheme, every pixel of the original secret image is expanded to m sub-pixels in a shadow image. A great effort has been directed toward reducing this pixel expansion. Verheul and Van Tilborg [5] introduced a general construction for a color (*k*, *n*)-VSS scheme. Based on the scheme in [5], Blundo et al. [6] and Yang and Laih [7] provided different construction for a color (*k*, *n*)-VSS scheme, also including color (*n*, *n*)-VSS scheme. Cimato et al. [8] presented a characterization of a *c*-color (*k*, *n*)-visual threshold scheme with optimal contrast. This scheme included a constructive proof of the optimality for a (*n*, *n*)-threshold scheme and obtained a lower bound for pixel expansion. The result is the same as the (*n*, *n*) -threshold scheme of [7] and it shows that the (*n*, *n*)-threshold scheme of [7] is optimal. Blundo et al. [9] defined and analyzed schemes for grayscale images, and provided a necessary and sufficient condition to construct such schemes for general access structures. The minimum relative differences can be obtained in any (*n*, *n*)-visual cryptography scheme for grayscale images. Iwamoto and Yamamoto in [10] presented a method that any (*n*, *n*)-VSS scheme for grayscale images can be constructed based on a polynomial representation of the basic matrices. The minimum pixel expansion of the (*n*, *n*)-VSS scheme for grayscale image in [10] is equal to the result in [9], namely $m \geq (g-1) \cdot 2^{n-1}$ where g is the number of grey levels. The deterministic VSS schemes mentioned above have achieved minimum pixel expansion m and relative difference α (=1/m), but the value of m can still be quite large, partly because m is proportional to the exponential of n. To further reduce pixel expansion, a number of probabilistic visual cryptography schemes (ProbVSS schemes) have been proposed in [11-14]. These schemes were designed for the case of $g = 2$, i.e., for black and white images. In the reconstructed secret image, the probability of white pixels in a white area is higher than that in a black area. Therefore small areas, instead of individual pixels, of the secret image can be recovered accurately. With the trade-off in resolution, probabilistic schemes can achieve zero pixel expansion (m =1), and the relative difference (the contrast parameter) is the same as the ones in the deterministic schemes. Recently, Wang et al. [15] proposed a deterministic (*n*, *n*)-secret sharing scheme for grayscale image, the scheme uses simple Boolean operations and has no pixel expansion.

In this paper, we propose a probabilistic (*n*, *n*)-visual secret sharing scheme for grayscale images. This scheme is an extension of the previously proposed deterministic (non-visual) (*n*, *n*)-secret sharing scheme in [15]. Its pixel expansion m is g-1, independent of *n*. The generation of the shadow images is based on Boolean operations OR and Exclusive-OR, and the reconstruction operation uses OR, as in other VSS schemes. The quality of the reconstructed image, measured in "Average Contrast" between consecutive grey levels, is equal to that between black and white in the probabilistic schemes proposed for binary images.

2 The Proposed Grayscale (*n*, *n*) Scheme

2.1 Quality Measures

Since the existing probabilistic schemes were only proposed for binary images, the contrast between black and white pixels was naturally chosen as an important quality measure. Our proposed scheme is for grayscale images. We use the expected contrast between two pixels with consecutive grey levels in the original image to indicate the quality. We define it as "Average Contrast" in detail below.

Let $U(i, j)$ be a pixel in the reconstructed image. This pixel corresponds to pixel s_{ij} in the original secret image. The appearance of $U(i, j)$ depends on the Hamming weight of the *m* vector: *H*(*U*). Because of the randomness of the shadow images, *H*(*U*) is a random variable. We are interested in the average Hamming weight for all pixels $U_{ij}\,|_k = U(i,j)\,|_k$ with corresponding pixels in the original secret image taking grey level *k*, namely, $s_{ij} = k,\ k \in \{0,1,\cdots,g-1\}$. With $A_{ij}^{(h)}$ representing the (*i, j*)-th pixel in the *h*-th shadow image, the reconstruction results is

$$U_{ij} = A_{ij}^{(1)} \text{ OR } A_{ij}^{(2)} \text{ OR} \ldots \text{OR } A_{ij}^{(n)} \tag{1}$$

Let $P_t = \text{Prob}(H(U_{ij}\,|_k) = t)$ be the probability of $H(U_{ij}\,|_k)$ taking value *t* with $t \in \{0,1,\cdots,g-1\}$, the expected value of $H(U_{ij}\,|_k)$ is $E(H(U_{ij}\,|_k)) = \sum_{t=0}^{g-1} t \cdot P_t$. We now define Average Grey β_k and Average Contrast α_k for the reconstructed image as

$$\beta_k = E(H(U_{ij}\,|_k)/m \text{ where } m \text{ is the pixel expansion factor,} \tag{2}$$

$$\alpha_k = \beta_k - \beta_{k-1} \tag{3}$$

2.2 Construction of the Shares

The secret image *S* is represented by an integer matrix of size $W \times H$. $S = [s_{ij}]_{W\times H}$ where $i = 1,2,\cdots,W,\ \ j = 1,2,\cdots,H,$ and $s_{ij} \in \{0,1,\cdots,g-1\}$. Each pixel of S can take any one of g different colors or grey levels.

In the construction of the shadow images, each pixel of *S* is coded as a binary string of $g-1$ bits.

For $s_{ij} = k$, its code form is $b_{g-1}^{k-1} = 0^{g-k}1^{k-1}$ that is a string of $g-k$ zeros and $k-1$ ones. The order of the bits does not matter. For example, b_{6-1}^{4-1} can be written as 00111, or 01101, or equivalently 11010.

We have $g = 2$ for a binary image, and $g = 256$ for a grayscale image with one byte per pixel. In a color image with one byte per pixel, the pixel value can be an index to a color table, thus $g = 256$.

In a color image using an RGB model, each pixel has three integers: R (red), G (green) and B (blue). If each R, G or B takes value between 0 and 255, we have $g = 256^3$.

Now, the description of the proposed scheme is given below.

Input:	The secret image *S*, $S = \{s_{ij}\}$ in the coded form $C = \{c_{ij}\}$.
Output:	The shadow images $F_1, \cdots, F_n$
Shadow Generation:	Randomly generate $n-1$ matrices $R^{(1)}, \cdots, R^{(n-1)}$ of size $W \times H$ where $R^{(h)} = \{r^{(h)}\}, \quad r^{(h)} \in \{0, \cdots, 2^{g-1} - 1\}$. $A^{(1)} = \{a_{ij}^{(1)}\} = R^{(1)}$, $A^{(h)} = \{a_{ij}^{h}\} = R^{(h-1)} \oplus R^{(h)}, \quad h = 2, \cdots, n-1,$ $A^{(n)} = \{a_{ij}^{n}\} = R^{(n-1)} \oplus S.$ Basic construction matrix is $B_{ij} = \begin{pmatrix} T(a_{ij}^{(1)}) \\ \vdots \\ T(a_{ij}^{(n)}) \end{pmatrix} = \begin{pmatrix} V_{ij}^{(1)} \\ \vdots \\ V_{ij}^{(n)} \end{pmatrix}$ where the transform *T* converts a binary string of $g-1$ bits into a row vector of $g-1$ components. The *h*-th component of the vector is the *h*-th bit in the input string. The *h*-th row of the basic matrix is used to construct the share image F_h.
Revealing:	$U = F_1 + \cdots + F_n$ where "+" is the Boolean "OR".

2.3 Proof of the Construction

Since the random matrices $R^{(1)}, \cdots, R^{(n-1)}$ are all distinct, the matrices $A_1, \cdots, A_n$ are also all distinct and all random, therefore each share does not reveal any information of *S* and security of the scheme is ensured. The quality of the scheme depends on the quality of the reconstructed image *U*. We now look at a pixel of the reconstructed image $U = F_1 + \cdots + F_n$. Theorem 1 states the average grey and average contrast of *U*.

Theorem 1. The proposed algorithm is a probabilistic (*n, n*)-VSS scheme with Pixel expansion

$$m = (g-1),$$

Average Grey

$$\beta_k = E(H(U))/m = [(1-\frac{1}{2^{n-1}})(g-k)+(k-1)]/(g-1),$$

and Average Contrast

$$\alpha_k = \beta_k - \beta_{k-1} = \frac{1}{2^{n-1}(g-1)}.$$

Proof.

Let *m* be the pixel expansion, we have $m=(g-1)$ according to the construction of the shares above.

Since $U = T(a^{(1)}) + \cdots + T(a^{(n)})$, we have

$U = T(r^{(1)}) + T(r^{(1)}) \oplus T(r^{(2)}) + \cdots T(r^{(n-2)}) \oplus T(r^{(n-1)}) + T(r^{(n-1)}) \oplus T(s)$

Substituting $T(r^{(i)})$ with V_i, we get

$$U = V_1 + V_1 \oplus V_2 + \cdots + V_{n-2} \oplus V_{n-1} + V_{n-1} \oplus V_0.$$

Here, V_0 is the coded form the original secret image *S*. That is, $V_0 = 0^{g-1}1^{k-1}$ for $s=k$. Since $V_1 + V_1 \oplus V_2 = V_1 + \overline{V_1}V_2 = V_1 + V_2$ and $V_1 + V_2 + V_2 \oplus V_3 = V_1 + V_2 + V_3$, we have

$$U = V_1 + V_2 + \cdots + V_{n-2} + V_{n-1} + (V_{n-1} \oplus V_0).$$

This can be rewritten as

$$U = W + V_{n-1} + (V_{n-1} \oplus V_0) \text{ where } W = V_1 + V_2 + \cdots + V_{n-2}.$$

We know that $V_{n-1} + (V_{n-1} \oplus V_0)$ must have at least $k-1$ bits being 1. That is $V_{n-1} + (V_{n-1} \oplus V_0)$ can be written as $x^{g-k}1^{k-1}$ where each of the $g-k$ bits, denoted by *x*, may take value 0 or 1. Therefore, $U = W + x^{g-k}1^{k-1} = y^{g-k}1^{k-1}$ also has at least $k-1$ bits being 1. The probability for each *y* bit to be 1 is $p = 1 - \frac{1}{2^{n-1}}$ since every of such bit depends on $n-1$ random matrices. The total number of 1's among these $g-k$ bits (the Hamming weight of the vector) is a random variable with a binomial distribution, and the expected value of the Hamming weight is

$$(1-\frac{1}{2^{n-1}}) \times (g-k) = p(g-k). \tag{4}$$

It follows that the expected Hamming weight of the entire $g-1$ vector is

$$E(H(U)) = (1-\frac{1}{2^{n-1}}) \times (g-k) + (k-1), \tag{5}$$

Thus the Average Grey is

$$\beta_h = E(H(U))/m = [(1-\frac{1}{2^{n-1}})(g-k)+(k-1)]/(g-1) \quad (6)$$

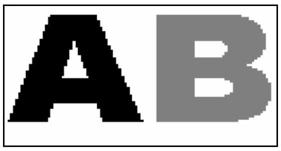
And the Average Contrast of the reconstructed image is

$$\alpha_k = \beta_k - \beta_{k-1} = \frac{1}{2^{n-1}(g-1)} \quad (7)$$

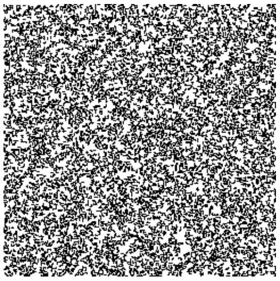
The following example will help in illustrating the technique employed in the previous theorem.

Example 1. An application of the proposed grayscale (3, 3)-VSS with $g = 3$.

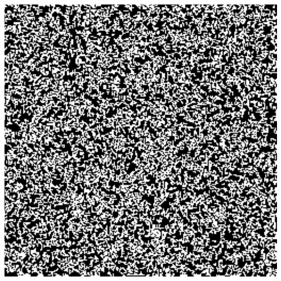
The secret image is shown in Figure 1(a). The three shadow images (shares) are in parts (b), (c), and (d). And the reconstructed image is in Fig. 1(e).



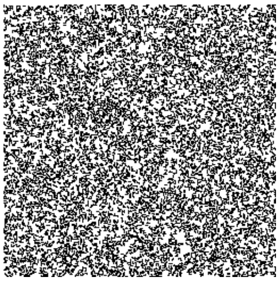
(a) Original secret image



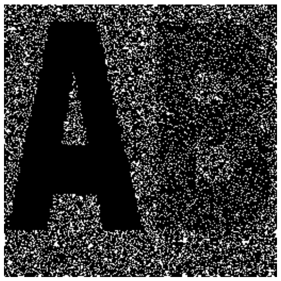
(b) Share 1



(c) Share 2



(d) Share 3



(e) Reconstructed the secret image

Fig. 1. Application example of the grayscale (3, 3)-VSS scheme

3 The Size of Recognizable Regions

With a probabilistic scheme, small regions (not individual pixels) of the secret image are correctly reconstructed. The smaller such regions can be, the better this scheme is. We now discuss the minimum size of the region that can be correctly recognized in an analysis similar to that in [12].

Consider a region of *N* pixels with the SAME grey level ($s_{ij} = k$) in the original secret image, and we are interested in the appearance of this region in the reconstructed image. The Average Grey of each of these *N* pixels is represented by X_k and X_k has a binomial distribution with μ_x and σ_x^2 . The total visual effect of the region is closely related to $Z = \sum_{i=1}^{N} X_k^{(i)}$. In the construction of the shadow images, all the pixels are treated separately from other pixels, thus the *N* pixels are independent, therefore, we have

$$E(Z) = E(\sum_{i=1}^{N} X_k^{(i)}) = \sum_{i=1}^{N} E(X_k^{(i)}) = N\mu_x = N[p(g-k)+(k-1)]$$

where $p = 1 - \frac{1}{2^{n-1}}$,

$$Var(Z) = Var(\sum_{i=1}^{N} X_k^{(i)}) = \sum_{i=1}^{N} Var(X^{(i)}k) = N\sigma_x^2 = N[p(1-p)(g-k)]$$

Using a Gaussian distribution to approximate the above binomial distribution, we can obtain the lower bound for *N*. According to Empirical Rule [12], about 99.73% of all values fall within three standard deviations of the mean. Hence, to recognize a region of grey level *k*, the region size should satisfy

$$\mu_k - 3\sigma_k > \mu_{k-1} + 3\sigma_{k-1} + N \cdot d ,$$

that is

$$N[p(g-k)+(k-1)] - 3\sqrt{Np(1-p)(g-k)} >$$
$$N[p(g-k+1)+(k-2)] + 3\sqrt{Np(1-p)(g-k+1)} + Nd$$

$$N[-p+1-d] > 3\sqrt{Np(1-p)(g-k)} + 3\sqrt{Np(1-p)(g-k+1)}$$

$$N > \frac{3\sqrt{N}\cdot\sqrt{p(1-p)}\cdot\left(\sqrt{(g-k)}+\sqrt{(g-k+1)}\right)}{1-p-d}$$

therefore

$$N > 9p(1-p)\cdot\left(\frac{\sqrt{g-k}+\sqrt{g-k+1}}{1-p-d}\right)^2 \tag{8}$$

When $g = k$, the above inequality becomes

$$N > \frac{9p(1-p)}{(1-p-d)^2} \tag{9}$$

Which indicates the minimum size of a recognizable region between grey level *g* and grey level $g-1$. When $g = 2$, the above is the minimum region size in a binary image. In the (*k*, *n*) probabilistic VSS scheme proposed in [12], the minimum region size is

$$N_{\text{Yang}} > 9\times\left(\frac{\sqrt{p_0(1-p_0)}+\sqrt{p_1(1-p_1)}}{p_0-p_1-d}\right)^2. \tag{10}$$

With $p_1 = 0$ and $p_0 = \frac{1}{2^{n-1}}$, it becomes

$$N_{\text{Yang}} > \frac{9\times p_0\times(1-p_0)}{(p_0-d)^2} \tag{11}$$

Comparing Equation (9) and (11), we have the following observations:

The minimum size of a recognizable region between grey level *g* and grey level $g-1$ of the proposed scheme is the same as that between black and white region in the (*n, n*)-Prob-VSS scheme of the (*k, n*))-Prob-VSS scheme of [12].

When our proposed scheme is applied to binary images, i.e., $g = 2$, its minimum region size is the same as that in [12].

4 Conclusions

This paper proposes a probabilistic (*n, n*) visual secret sharing scheme for grayscale images. Its pixel expansion factor is $g-1$ where *g* is the number of grey levels. This is significantly smaller than the previous result $2^{n-1}\times(g-1)$. When applied to binary images, it has the same minimum size for recognizable regions as that of the Prob-VSS scheme of [12]. We are currently investigating the approaches for a probabilistic (*k, n)*-visual secret sharing scheme for grayscale images.

Acknowledgements

This research was supported in part by the National Natural Science Foundation of the People's Republic of China under Grant No. 90304014; the Project funded by Basic

Research Foundation of School of Information Science and Technology of Tsinghua and was also supported in part by Canadian Natural Sciences and Engineering Research Council under Grant OGP9198.

References

1. Naor, M., Shamir, A.: Visual Cryptography. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 1–12. Springer, Heidelberg (1995)
2. Rijmen, V., Preneel, B.: Efficient Colour Visual Encryption or 'Shared Colors of Benetton'. In: Rump Session of EUROCRYPTO 1996 (1996), http://www.iacr.org/conferences/ec96/rump/preneel.ps
3. Yang, C.N.: A Note on Efficient Color Visual Encryption. Journal of Information Science and Engineering 18(3), 367–372 (2002)
4. Hou, Y.C.: Visual Cryptography for Color Images. Pattern Recognition 36(7), 1619–1629 (2003)
5. Verheul, E.R., Van Tilborg, H.C.A.: Constructions and Properties of k out of n Visual Secret Sharing Schemes. Designs, Codes, and Cryptography 11, 179–196 (1997)
6. Blundo, C., De Bonis, A., De Santis, A.: Improved Schemes for Visual Cryptography. Designs, Codes, and Cryptography 24, 255–278 (2001)
7. Yang, C.N., Laih, C.S.: New Colored Visual Secret Sharing Schemes. Designs, Codes, and Cryptography 20, 325–335 (2000)
8. Cimato, S., De Prisco, R., De Santis, A.: Contrast Optimal Colored Visual Cryptography Schemes. In: IEEE Information Theory Workshop, pp. 139–142. IEEE Press, New York (2003)
9. Blundo, C., De Santis, A., Naor, M.: Visual Cryptography for Grey Level images. Information Processing Letters 75, 255–259 (2000)
10. Iwamoto, M., Yamamoto, H.: The Optimal n -out of- n Visual Secret Sharing Scheme for Gray-scale Images. IEICE Transactions on Fundamentals of Electronics E85-A (10), 2238–2247 (2002)
11. Ito, R., Kuwakado, H., Tanaka, H.: Image Size Invariant Visual Cryptography. IEICE Transactions on Fundamentals of Electronics E82-A (10), 2172–2177 (1999)
12. Yang, C.N.: New Visual Secret Sharing Schemes Using Probabilistic Method. Pattern Recognition Letters 25, 481–494 (2004)
13. Cimato, S., De Prisco, R., De Santis, A.: Probabilistic Visual Cryptography Schemes. Computer Journal 49(1), 97–107 (2006)
14. Hsu, C.S., Tu, S.F., Hou, Y.C.: An Optimization Model for Visual Cryptography Schemes with Unexpanded Shares. In: Esposito, F., Raś, Z.W., Malerba, D., Semeraro, G. (eds.) ISMIS 2006. LNCS (LNAI), vol. 4203, pp. 58–67. Springer, Heidelberg (2006)
15. Wang, D.S., Zhang, L., Ma, N., Li, X.: Two Secret Sharing Schemes Based on Boolean Operations. Pattern Recognition 40, 2776–2785 (2007)

Mutually Clock-Controlled Feedback Shift Registers Provide Resistance to Algebraic Attacks

Sultan Al Hinai[1], Lynn Margaret Batten[2], and Bernard Colbert[2]

[1] Queensland University of Technology, Brisbane, Australia
s.alhinai@student.qut.edu.au
[2] Deakin University, Melbourne, Australia
lmbatten@deakin.edu.au, bernard.colbert@team.telstra.com

Abstract. Algebraic attacks have been applied to several types of clock-controlled stream ciphers. However, to date there are no such attacks in the literature on mutually clock-controlled ciphers. In this paper, we present a preliminary step in this direction by giving the first algebraic analysis of mutually clock-controlled feedback shift register stream ciphers: the bilateral stop-and-go generator, A5/1, Alpha 1 and the MICKEY cipher. We show that, if there are no regularly clocked shift registers included in the system, mutually clock-controlled feedback shift register ciphers appear to be highly resistant to algebraic attacks. As a demonstration of the weakness inherent in the presence of a regularly clocked shift register, we present a simple algebraic attack on Alpha 1 based on only 29 keystream bits.

Keywords: stream cipher, algebraic attacks, clock-control.

1 Introduction

Algebraic attacks on stream ciphers were first introduced by Courtois and Meier [14] where the keystream is used to solve a system of multivariate polynomial equations related to the initial states of the cipher. Many *regularly* clocked feedback shift register (FSR) based stream ciphers have since fallen to algebraic attacks [2, 11, 12, 13, 17, 8], whereas *irregularly* clocked stream ciphers have been more resistant. There are, to our knowledge, only three papers in the literature dealing with algebraic attacks on irregularly clocked stream ciphers [14, 1, 22]. In the current paper, we focus on an analysis of mutually clock-controlled FSR-based stream ciphers where the clocking is irregular.

The *standard* algebraic attack as presented in [14] uses the output of a cipher to generate relational equations between the initial state bits. The monomials in these equations are then treated as new variables in order to *linearize* the system. If the result is an over-defined system of linear equations (often sparse), Gaussian elimination techniques [20] are then applied to determine the initial state bits. When constructing a cipher, the hope is that the time to solve the system generated is greater than that of exhaustive search.

A variation on the above idea is to first discover a function which, when multiplied by the function describing the initial state relations, significantly reduces the degree, leading to a faster Gaussian solution. Several papers have addressed the issue of finding multipliers algorithmically [19,7,3,15] and [24], but at present, none of these guarantees an efficient determination of a low degree multiple for very large systems of equations of high degree such as those produced by irregular clocking ciphers.

Guessing some initial state bits in order to reduce the number of monomials generated is a method that has been used to successfully attack some ciphers [1, 14]. In Section 5, we apply this guessing method to Alpha 1 targeting its regularly clocked register. We use only 29 bits of keystream to break the cipher in less than brute force time indicating a surprising weakness in the cipher.

In Section 2, we define what we mean by a mutually clock-controlled feedback shift register and state a number of theoretical results which indicate that these are in general highly immune to the standard algebraic attack. Sections 3, 4, 5 and 6 respectively present an analysis of the four ciphers bilateral stop-and-go [23], A5/1 [9], Alpha 1 [16] and the MICKEY family [4,5,6]. We conclude in Section 7.

2 Algebraic Presentation

In irregularly clocked stream ciphers, one cannot tell from the output which registers clocked and which did not. However, knowledge of the architecture of the system permits us to write equations representing the clocking. The more complex the clocking mechanism is, the more complex the equations (high degree, large number of monomials) are. Mutually clocking registers is one way of adding complexity. By this we mean that (at least) two registers depend on bits of at least two registers in the cipher for their clocking determination. The four ciphers mentioned at the end of Section 1 are examples.

Definition 1. ***A Mutually Clock-Controlled FSR-based stream cipher*** *is a stream cipher consisting of two or more FSRs where at least two* ***FSRs*** *use bits from each other for clocking. We denote such a stream cipher by* ***MCFSR****.*

Assume that we have an MCFSR with just two registers R1 and R2 of size l and m respectively and that the clocking of each register depends on bits from both registers. We can express this dependence by a function $g(x_i, y_i)$ of bits x_i from R1 and y_i from R2. The function g maps to binary pairs indicating whether or not the corresponding register clocks, as indicated in Table 1. It is usually not practicable to allow the situation in which neither R1 nor R2 clocks, so this situation is often changed to both registers clocking. Any permutation of the above list can of course also be chosen.

Based on the table, one can easily incorporate the clocking mechanism into the internal state of each register. Letting x represent the first bit of $g(x_i, y_i)$

Table 1. Description of clocking

$g(x_i, y_i)$		**Register to be clocked**		
0	0	**R1**	and	**R2**
0	1	**R2**	not	**R1**
1	0	**R1**	not	**R2**
1	1	**R1**	and	**R2**

and y the second, the effect of the clocking mechanism or the internal state of registers R1 and R2 at time t can be described as follows:

$$\begin{aligned} R1_i^t = R1_i^{t-1} &\left[(x^{t-1}+1)(y^{t-1}+1) + x^{t-1}(y^{t-1}+1) + x^{t-1}y^{t-1} + 1\right] \\ +R1_{i-1}^{t-1} &\left[(x^{t-1}+1)(y^{t-1}+1) + x^{t-1}(y^{t-1}+1) + x^{t-1}y^{t-1}\right] \end{aligned} \tag{1}$$

$$\begin{aligned} R2_i^t = R2_i^{t-1} &\left[(x^{t-1}+1)(y^{t-1}+1) + (x^{t-1}+1)y^{t-1} + x^{t-1}y^{t-1} + 1\right] \\ +R2_{i-1}^{t-1} &\left[(x^{t-1}+1)(y^{t-1}+1) + (x^{t-1}+1)y^{t-1} + x^{t-1}y^{t-1}\right] \end{aligned} \tag{2}$$

Despite the fact that the clocking mechanism used is very simple, the clocking of each register is nonlinear with degree increasing quickly as time t increases.

The following theorem and its corollaries describe the strengths of this situation. For simplicity, we often reduce to the case of two registers.

Theorem 1. *An MCFSR with two registers of size l and m will generate equations of degree equal to $l+m$ and a maximum of $\sum_{j=1}^{l+m} \binom{l+m}{j}$ monomials.*

Proof: Label the bits of register 1 (*R1*), x_i and the bits of register 2 (*R2*), y_i. Assume that the clocking of Ri can be described by a nonlinear function f_i of degree d that takes as input some values of x_i and y_i and determines which register is to be clocked. Assume also that the output of the system is given by some combination of the outputs of the two registers. We can write the $j'th$ internal state of each register as

$$Rk_j^t = Rk_j^t(f(x_i, y_i)^t + 1) + Rk_{j-1}^t(f(x_i, y_i)^t) \tag{3}$$

for $k = 1, 2$. At time $t = 1$, each position Ri_j contains an expression of minimum degree d. At time $t = 2$, the degrees of the expressions in Ri_j doubles; this continues until the degree reaches the size of the internal state size of the cipher, which is $l + m$. Thus the maximum degree of the equations generated from the system is $l + m$ (which can always be achieved assuming the cipher is permitted to keep producing output) and the maximum number of monomials is

$$M = \sum_{j=1}^{l+m} \binom{l+m}{j} \tag{4}$$

The complexity of solving such a system of equations in an MCFSR using this linearization approach is therefore at least

$$\left[\sum_{j=1}^{l+m} \binom{l+m}{j}\right]^{\omega} \tag{5}$$

□

where $2.807 \leq \omega \leq 3$ [20], which is greater than the complexity of exhaustive search using currently known best algorithms for solving equations.

As noted in the introduction, guessing some initial state bits may assist an attack. In the corollary following, we easily describe the impact of such guessing.

Corollary 1. *Guessing u bits of the internal state bits of a mutually clocked controlled stream cipher reduces the overall degree of the equations by u.*

Proof: One can see from equation (5) of Theorem 1 that each guess of u bits reduces the degree of the equations by u. In addition, the complexity of algebraic attacks using the guessing approach will be

$$2^u \left[\sum_{j=1}^{(l+m-u)} \binom{(l+m-u)}{j}\right]^{\omega} \tag{6}$$

□

Corollary 2. *The existence of a regularly clocked register in an MCFSR reduces the complexity of the algebraic attack to less than exhaustive key search, assuming that the key size is the same as the total number of initial state bits.*

Proof: Let l be the total number of bits from regularly clocked registers, and k the key size. Then guessing all $k - l$ bits from the irregularly clocked registers produces only linear equations in $\binom{l}{1} = l$ monomials using the procedure of the proof of Theorem 1. The overall attack complexity is then

$$2^{k-l} l^{\omega} < 2^k \quad \text{if} \quad l \quad \text{is at least} \quad 4. \tag{7}$$

□

One might conjecture that increasing the number of clock-control bits used in an MCFSR would increase the level of complexity of the corresponding equations. The following corollary shows that this is only partly true.

Corollary 3. *Increasing the number of clock-control bits in an MCFSR may increase the degree of the equations generated during the first outputs but does not increase the maximum degree of the equations generated.*

Proof: Equation (3) shows that the maximum degree of the equations generated is independent of the number of clock-control bits. □

As mentioned in the Introduction, multiplier functions may exist which can reduce the degrees of the functions under consideration. However, the larger the system of equations generated, and the higher the degrees of these equations, the more difficult it is to find such multipliers. Thus, one of the strategies for choosing FSR-based ciphers which are immune to algebraic attack is to make sure that the number and degree of equations generated is high. MCFSRs appear to have both properties.

3 The Bilateral Stop-and-Go Generator

This section applies the analysis of Section 2 to the bilateral-stop-and-go generator. We start by a description of the cipher followed by an algebraic analysis.

3.1 Description of the Bilateral Stop and Go Generator

The bilateral Stop-and-Go generator is one of the first known examples of a mutually clock-controlled stream cipher [23]. It consists of two linear feedback shift registers (LFSRs), $R1$ and $R2$ of the same length l. Both LFSRs control their own clocking using two bits from each LFSR in the following way. Let a, b and c, d be pairs of bits from $R1$ and $R2$ respectively. If $(a, b) = (0, 1)$, then clock only $R1$. On the other hand, if $(c, d) = (0, 1) \neq (a, b)$, then clock only $R2$, otherwise clock both registers. Since there are 4 bits controlling the clocking of $R1$, it can be seen that R1 is clocked with probability equal 13/16 whereas R2 is clocked with 3/4 probability. To the best of our knowledge, there is only one correlation attack on this cipher by Golić [21] which requires l keystream bits with O($2^{l+3log_2 l}$) operations.

3.2 Algebraic Analysis of the Bilateral Stop and Go Generator

Let the output of register $R1$ be $R1_l$ and $R2$ be $R2_l$. Based on the description of the way registers $R1$ and $R2$ are clocked, we can describe the effect of the clocking mechanism on the internal state of each register described by a binary truth table of all the clocking possibilities respectively as follows

$$\begin{aligned} R1_i^t = R1_i^{t-1}((&a^{t-1}b^{t-1}c^{t-1}d^{t-1} + a^{t-1}b^{t-1}d^{t-1} + b^{t-1}c^{t-1}d^{t-1} + b^{t-1}d^{t-1} + \\ &c^{t-1}d^{t-1} + d^{t-1}) + R1_{i-1}^{t-1}((a^{t-1}b^{t-1}c^{t-1}d^{t-1} + a^{t-1}b^{t-1}d^{t-1} + \\ &b^{t-1}c^{t-1}d^{t-1} + b^{t-1}d^{t-1} + c^{t-1}d^{t-1} + d^{t-1} + 1) \end{aligned} \tag{8}$$

Thus $R1_1$ is given by

$$\begin{aligned} R1_1^t = R1_1^{t-1}((&a^{t-1}b^{t-1}c^{t-1}d^{t-1} + a^{t-1}b^{t-1}d^{t-1} + b^{t-1}c^{t-1}d^{t-1} + b^{t-1}d^{t-1} + \\ &c^{t-1}d^{t-1} + d^{t-1}) + FeedbackA((a^{t-1}b^{t-1}c^{t-1}d^{t-1} + a^{t-1}b^{t-1}d^{t-1} + \\ &b^{t-1}c^{t-1}d^{t-1} + b^{t-1}d^{t-1} + c^{t-1}d^{t-1} + d^{t-1} + 1) \end{aligned}$$

Similarly $R2_i^t$ can be represented as

$$R2_i^t = R2_i^{t-1}(a^{t-1}b^{t-1} + b^{t-1}) + R2_{i-1}^{t-1}(a^{t-1}b^{t-1} + b^{t-1} + 1) \tag{9}$$

And the $R2_1^t$ position is given by

$$R2_1^t = R2_1^{t-1}(a^{t-1}b^{t-1} + b^{t-1}) + FeedbackB(a^{t-1}b^{t-1} + b^{t-1} + 1)$$

The output of the generator at time t is:

$$z^t = R1_l^t + R2_l^t$$

For the purpose of our paper, we will substitute the values for $R1_l^t$ and $R2_l^t$ using equations 8 and 9 respectively and expand it as follows:

$$\begin{aligned} z^t = f^t = {} & a^{t-1}b^{t-1}c^{t-1}d^{t-1}R1_i^{t-1} + a^{t-1}b^{t-1}c^{t-1}d^{t-1}R1_{i-1}^{t-1} + a^{t-1}b^{t-1}d^{t-1}R1_i^{t-1} \\ & + b^{t-1}c^{t-1}d^{t-1}R1_i^{t-1} + a^{t-1}b^{t-1}d^{t-1}R1_{i-1}^{t-1} + b^{t-1}c^{t-1}d^{t-1}R1_{i-1}^{t-1} \\ & + b^{t-1}d^{t-1}R1_i^{t-1} + c^{t-1}d^{t-1}R1_i^{t-1} + b^{t-1}d^{t-1}R1_{i-1}^{t-1} + c^{t-1}d^{t-1}R1_{i-1}^{t-1} \\ & + a^{t-1}b^{t-1}R2_i^{t-1} + a^{t-1}b^{t-1}R2_{i-1}^{t-1} + d^{t-1}R1_i^{t-1} + d^{t-1}R1_{i-1}^{t-1} + b^{t-1}R2_i^{t-1} \\ & + b^{t-1}R2_{i-1}^{t-1} + R1_{i-1}^{t-1} + R2_{i-1}^{t-1}. \end{aligned} \tag{10}$$

The output function f is initially of degree 5, however, because of the nonlinear update function of each register, the degree tends to increase. According to Theorem 1, the maximum degree of the equations generated from this cipher is $d = 2l$ and the number of monomials expected to appear is given by equation 4. Clearly, trying to recover the internal state of such generators using the standard algebraic attack will usually require complexity much worse than exhaustive search on the internal state size.

3.3 Reducing the Overall Degree of the Equations

In this section, we analyse the applicability of two approaches used in reducing the overall degree of the equations on our cipher. As Corollary 1 shows, guessing any u bits for $u \leq 2l$ will result in the following attack complexity $2^u \sum_{i=1}^{2l-u} \binom{2l-u}{i}^{\omega}$. In fact, guessing the internal state bits of $R1$ has the same effect as guessing the internal state bits of $R2$ since both registers are of the same length. In order to get an algebraic attack that is less complex than exhaustive key search when some bits are guessed, the following relations must hold $2^u \sum_{i=1}^{2l-u} \binom{2l-u}{i}^{\omega} < 2^{2l}$. However, it can be seen that this relation will never hold true for any of its parameters; therefore, the guessing approach is not suitable.

In the other approach, the attacker aims to find annihilators or multiples that will reduce the overall degree of the equations. By looking at the output given by equation 10, one can see that there exist many multiples and annihilators for the output function. Table 2 below lists the number of multiples of g found on the output and the number of h such that $h = gf$.

Note that the number 0 in Table 1 indicates that there exist many degree 4 multiples of g such that $fg = 0$. If we multiply both sides of equation 10 with a^{t-1} what remains is the following expression:

$$\begin{aligned} fg &= fa^{t-1} \\ &= a^{t-1}c^{t-1}d^{t-1}R1_i^{t-1} + a^{t-1}c^{t-1}d^{t-1}R1_{i-1}^{t-1} \\ &\quad + a^{t-1}d^{t-1}R1_i^{t-1} + a^{t-1}d^{t-1}R1_{i-1}^{t-1} + a^{t-1}R1_{i-1}^{t-1} + a^{t-1}R2_{i-1}^{t-1}. \end{aligned} \tag{11}$$

Table 2. Number of multiples of g found

Degree of g	Degree of h	Total Number of g and h
1	4	4
2	3 and 4	10
3	3 and 4	42
4	0 and 4	48

This multiplication eliminates many terms of different degrees. In particular, the multiple eliminates the effect of b^{t-1}. However, the degree of such multiples keeps increasing, so reducing the possibility of an algebraic attack. Note also that we have observed that multiplying the output of the generator with $\prod(x_i+1)$, where x_i runs the contents bits through R1, then the overall degree of the equations can be reduced from $2l$ to $l+1$.

4 A5/1

4.1 Description of A5/1

A5/1 consists of three short binary LFSRs of lengths $19, 22$ and 23, denoted by $R1, R2$ and $R3$, respectively. The three LFSRs have the following primitive feedback polynomials:

$$f_1(x) = x^{19} + x^5 + x^2 + x + 1$$

$$f_2(x) = x^{22} + x + 1$$

$$f_3(x) = x^{23} + x^{15} + x^2 + x + 1$$

At time t, denote the output of register Ri as Ri^t and the output keystream of A5/1 as z^t. The output of A5/1 is given as the XOR of the output of the three LFSRs. The LFSRs are clocked in an irregular fashion, determining a type of stop/go clocking with a majority rule as follows. Each register has a certain clocking tap used to determine its clocking and the clocking of other registers. $R1_9^t$, $R2_{11}^t$ and $R3_{11}^t$ are the bits taken from $R1$, $R2$ and $R3$ as input to the majority function. The majority function is given by

$$M^t = (R1_9^t R2_{11}^t + R1_9^t R3_{11}^t + R2_{11}^t R3_{11}^t)$$

R1 gets clocked if $R1_9$ agrees with the majority function. Similarly, register $R2$ gets clocked if $R2_{11}^t$ agrees with the majority function. Finally, register $R3$ gets clocked if $R3_{11}^t$ agrees with the majority function.

4.2 Algebraic Analysis of A5/1

Based on the clocking mechanism used in A5/1, it is possible to rewrite the internal states of the three registers so they take into account the clocking mechanism in place, which as a result will produce valid relations that hold true for every clock. Each i^{th} stage in $R1$ is replaced by:

$$R1_i^t = R1_i^{t-1}(R1_9^{t-1} + M^{t-1}) + (R1_{i-1}^{t-1}(1 + R1_9^{t-1} + M^{t-1})) \tag{12}$$

and the left most position is given by

$$R1_1^t = R1_1^{t-1}(R1_9^{t-1} + M^{t-1}) + ((R1_{19}^{t-1} + R1_{18}^{t-1} + R1_{17}^{t-1} + R1_{14}^{t-1})(1 + R1_9^{t-1} + M^{t-1})).$$

Similarly for $R2$, each i^{th} stage in $R2$ is replaced by:

$$R2_i^t = R2_i^{t-1}(B_{11}^{t-1} + M^{t-1}) + (R2_{i-1}^{t-1}(1 + R2_{11}^{t-1} + M^{t-1})) \tag{13}$$

and the left most position is given by

$$R2_1^t = R2_1^{t-1}(R2_{11}^{t-1} + M^{t-1}) + ((R2_{21}^{t-1} + R2_{22}^{t-1})(1 + R2_{11}^{t-1} + M^{t-1})).$$

Finally, each i^{th} stage in $R3$ is replaced by:

$$R3_i = R3_i^{t-1}(R3_{11}^{t-1} + M^{t-1}) + (R3_{i-1}^{t-1}(1 + R3_{11}^{t-1} + M^{t-1})) \tag{14}$$

and the left most position is given by

$$R3_1^t = R3_1^{t-1}(R3_{11}^{t-1} + M^{t-1}) + ((R3_{23}^{t-1} + R3_{22}^{t-1} + R3_{21}^{t-1} + R3_8^{t-1})(1 + R3_{11}^{t-1} + M^{t-1})).$$

The output is given by

$$z = R1_{19}^t + R2_{22}^t + R3_{23}^t$$

and z^t can be described as follows

$$\begin{aligned} z^t = f^t = {} & R1_{19}^{t-1}R1_9^{t-1}R2_{11}^{t-1} + R1_9^{t-1}R1_{18}^{t-1}R2_{11}^{t-1} + R1_9^{t-1}R2_{22}^{t-1}R2_{11}^{t-1} + \\ & R1_9^{t-1}R2_{11}^{t-1}R2_{21}^{t-1} + R1_9^{t-1}R2_{11}^{t-1}R3_{23}^{t-1} + R1_{19}^{t-1}R1_9^{t-1}R3_{11}^{t-1} + \\ & R1_9^{t-1}R1_{18}^{t-1}R3_{11}^{t-1} + R1_9^{t-1}R2_{22}^{t-1}R3_{11}^{t-1} + R1_{19}^{t-1}R2_{11}^{t-1}R3_{11}^{t-1} + \\ & R1_{18}^{t-1}R2_{11}^{t-1}R3_{11}^{t-1} + R2_{22}^{t-1}R2_{11}^{t-1}R3_{11}^{t-1} + R1_9^{t-1}R2_{21}^{t-1}R3_{11}^{t-1} + \\ & R2_{11}^{t-1}R2_{21}^{t-1}R3_{11}^{t-1} + R1_9^{t-1}R3_{23}^{t-1}R3_{11}^{t-1} + R2_{11}^{t-1}R3_{23}^{t-1}R3_{11}^{t-1} + \\ & R1_9^{t-1}R2_{11}^{t-1}R3_{22}^{t-1} + R1_9^{t-1}R3_{11}^{t-1}R3_{22}^{t-1} + R2_{11}^{t-1}R3_{11}^{t-1}R3_{22}^{t-1} + \\ & R1_{19}^{t-1}R1_9^{t-1} + R1_9^{t-1}R1_{18}^{t-1} + R2_{22}^{t-1}R2_{11}^{t-1}R2_{11}^{t-1}R2_{21}^{t-1} + \\ & R3_{23}^{t-1}R3_{11}^{t-1} + R3_{11}^{t-1}R3_{22}^{t-1} + R1_{18}^{t-1} + R2_{21}^{t-1} + R3_{22}^{t-1}. \end{aligned} \tag{15}$$

4.3 Reducing the Overall Degree of the Equations

Using the guessing approach, the complexity of the approach varies depending on which registers are guessed and which registers are solved for. Table 3 lists the results for all possibilities one can take. It can be seen from Table 3, that the best result obtained is by guessing R2 and R3 and solving for the internal state of register 1. However, the complexity is still much higher than exhaustive key search.

Table 3. Algebraic attacks on A5/1 with the guessing approach

Guessed Register	**Max degree of equations**	**Total Attack Complexity**
R1	45	$\approx (2^{19}.2^{182.5}) = 2^{201.5}$
R2	42	$\approx (2^{22}.2^{169.7}) = 2^{191.7}$
R3	41	$\approx (2^{23}.2^{165.4}) = 2^{188.4}$
R1 and R2	23	$\approx (2^{41}.2^{88.7}) = 2^{129.7}$
R1 and R3	22	$\approx (2^{42}.2^{84.5}) = 2^{126.5}$
R2 and R3	19	$\approx (2^{45}.2^{71.8}) = 2^{116.8}$

As for the approach which is based on finding low degree multiples, the following was found. It is clear that the degree of the output function is initially 3. But as the internal state of each register in the ciphers is nonlinearly updated, the degree of the generated equations will increase. Based on Theorem 1, the maximum degree of the equations generated from A5/1 is $d = l + m + n = 64$ and the number of monomials expected to appear is given by equation 4. Clearly, trying to recover the internal state of such generators using the classical algebraic attack will require complexity much worse than exhaustive search on the internal state size.

We have not found any multiplier functions g of degree 1 or 2, however, we have found 9 functions of degree 3 such that $fg = 0$. But as expected, the degree of these functions increases. Interestingly though, some of these functions eliminate the effect of one or more registers.

5 Alpha 1

5.1 Description of Alpha 1

Alpha 1 is based on four binary linear LFSRs of lengths 29, 31, 33 and 35 bits, which are denoted as $R1, R2, R3$ and $R4$ respectively.

The LFSRs have the following feedback polynomials:

$$f_1(x) = x^{29} + x^{27} + x^{24} + x^8 + 1$$

$$f_2(x) = x^{31} + x^{28} + x^{23} + x^{18} + 1$$

$$f_3(x) = x^{33} + x^{28} + x^{24} + x^4 + 1$$

$$f_4(x) = x^{35} + x^{30} + x^{22} + x^{11} + x^6 + 1$$

At time t, denote the output of register Ri as Ri^t and the output keystream of Alpha 1 as z^t. The keystream bit is a function of the output bit of each of the four registers.

$$\begin{aligned} z^t &= f(R1^t, R2^t, R3^t, R4^t) \\ z^t &= R1^t + R2^t + R3^t + R4^t + (R2^t AND R3^t). \end{aligned} \tag{16}$$

Let Ri_j denote the j^{th} stage of $LFSRi$. While R1 is regularly clocked, R2, R3 and R4 are clocked in a stop/go fashion according to the following majority rule.

Alpha 1 employs two majority rule functions, $M1^t$ and $M2^t$, and uses pairs of clocking taps from R2, R3, and R4 to control the three registers clocking, where $M1^t$ and M^2t are respectively given by

$$M1^t = R2_{11}^t R3_{23}^t + R2_{11}^t R4_{12}^t + R3_{23}^t R4_{12}^t$$

and

$$M2^t = R2_{22}^t R3_{11}^t + R4_{25}^t R2_{22}^t + R3_{11}^t R4_{25}^t.$$

R2 is clocked if $R2_{11}^t$ agrees with $M1^t$ and $R2_{22}^t$ agrees with $M2^t$. R3 is clocked if $R3_{23}^t$ agrees with $M1^t$ and $R3_{11}^t$ agrees with $M2^t$. Finally, R4 is clocked if $R4_{12}^t$ agrees with $M1^t$ and $R4_{25}^t$ agrees with $M2^t$.

5.2 Algebraic Attack on Alpha 1

Before we describe our attack, we show how to obtain relationships between the internal states of the cipher with the output that takes into account the irregular clocking used. As R1 is regularly clocked, then each i^{th} stage in $R1$ is replaced by:

$$R1_i^t = R1_{i-1}^{t-1}.$$

Each i^{th} stage in $R2$ is replaced by: $R2_i^t =$

$$R2_i^{t-1}(R2_{11}^{t-1}+R2_{23}^{t-1}+M_1^{t-1}+M_2^{t-1})+R2_{i-1}^{t-1}(R2_{11}^{t-1}+R2_{23}^{t-1}+M_1^{t-1}+M_2^{t-1}+1)$$

and the left most position is given by $R2_1^t =$

$$R2_1^{t-1}(R2_{11}^{t-1} + R2_{23}^{t-1} + M_1^{t-1} + M_2^{t-1}) + ((1 + R2_{11}^{t-1} + R2_{23}^{t-1} + M_1^{t-1} + M_2^{t-1})(R2_{31}^{t-1} + R2_{28}^{t-1} + R2_{23}^{t-1} + R2_{18}^{t-1})).$$

Each i^{th} stage in $R3$ is replaced by: $R3_i^t =$

$$R3_i^{t-1}(R3_{23}^{t-1}+R3_{11}^{t-1}+M_1^{t-1}+M_2^{t-1})+R3_{i-1}^{t-1}(R3_{23}^{t-1}+R3_{11}^{t-1}+M_1^{t-1}+M_2^{t-1}+1)$$

and the left most position is given by

$$R_1^t = R3_1^{t-1}(R3_{23}^{t-1} + R3_{11}^{t-1} + M_1^{t-1} + M_2^{t-1}) + ((R3_{23}^{t-1} + R3_{11}^{t-1} + M_1^{t-1} + M_2^{t-1} + 1)(R3_{33}^{t-1} + R3_{28}^{t-1} + R3_{24}^{t-1} + R3_4^{t-1})).$$

Finally, each i^{th} stage in $R4$ is replaced by: $R4_i^t =$

$$R4_i^{t-1}(R4_{12}^{t-1}+R4_{25}^{t-1}+M_1^{t-1}+M_2^{t-1})+R4_{i-1}^{t-1}(R4_{12}^{t-1}+R4_{25}^{t-1}+M_1^{t-1}+M_2^{t-1}+1).$$

The output of the cipher is given by

$$z^t = R1_{29}^t + R2_{31}^t + R3_{33}^t + R4_{35}^t + R2_{31}^t R3_{33}^t. \quad (17)$$

5.3 Reducing the Overall Degree of the Equations

It is expected that Alpha 1 will generate equations of a maximum degree of $d = 99$ with number of monomials equal to $M = \sum_{i=1}^{99} \binom{99}{i}$. Thus applying the standard algebraic attack will have much worse complexity than the complexity of exhaustive keysearch. Note that the regularly clocked register, $R1$, will have no effect at all in determining the size of the generated equations. Table 4 lists different choices for the guessing and solving approach for Alpha 1. Note that the table only considers guessing the mutually clocked registers. It is evident from Table 4 that none of the above choices will lead to an attack that is better than exhaustive key search. However, based on Corollary 2, we see that the key size of the cipher is the same as the internal state size $k = 128$, and there is at least one register that is regularly clocked. Hence, using Corollary 2 and equation (17), guessing $R2, R3$ and $R4$ will result in linear equations in the output, containing unknowns from $R1$ only. Thus the complexity of this basic algebraic attack is $2^{31+33+35}(29)^2 = 2^{108}$, which is less than exhaustive key search. Table 5 lists the best known attacks on the Alpha 1 stream ciphers. We tested the above attack on the full version of Alpha 1 by running some experimental simulations. In the simulations we tested two things; first, whether 29 bits are sufficient to solve the system of equations and give unique solutions. Second, whether 29 bits is enough to enable us to reject wrong guesses for the bits of $R1$ since it might happen that inequivalent internal state bits might result in the same 29 keystream bits. This never happened in our simulations; we were always able to verify that the correct internal state had been chosen. In any event, 128 bits of keystream would be sufficient to guarantee the correct internal state, and this is well within the limits of the typical frame-length.

Table 4. Standard Algebraic attacks on Alpha 1 with the guessing approach

Guessed Register	**Max degree of equations**	**Total Attack Complexity**
R2	68	$\approx (2^{31}.2^{281}) = 2^{312}$
R3	66	$\approx (2^{33}.2^{272.6}) = 2^{305.6}$
R4	64	$\approx (2^{35}.2^{264}) = 2^{299}$
R2 and R3	35	$\approx (2^{64}.2^{139.8}) = 2^{203}$
R2 and R4	33	$\approx (2^{66}.2^{131}) = 2^{197}$
R3 and R4	31	$\approx (2^{68}.2^{122.7}) = 2^{190.7}$

Table 5. Best known attacks on the Alpha 1 stream cipher

Attack	**Keystream**	**Attack Complexity**
[**10**]	$35,000$ bits	2^{61}
Our attack	29 bits	2^{108}

In the experiment, we first generated and recorded 29 keystream bits by randomly choosing the values of all four registers. Then we applied the attack approach briefly outlined above for 100,000 runs, in each run randomly guessing the contents of $R2$, $R3$ and $R4$. We next generated a linear system of equations in the unknowns of R1 and solved it using the F4 algorithm of the Gröbner bases method available in Magma [18]. In each case, we obtained a unique solution for the bits of R1 and used these, together with the guessed bits for R2, R3 and R4 to reconstruct the keystream bits which were then compared with the recorded keystream. In each run, we managed with 100 percent success rate to solve the system of equations and detect if the guesses made were right or wrong. This was done Magma 2.11 on the SGI Origin 3000 using CPU at 600 MHz. The fact that this can be done with only 29 bits of keystream suggests a here-to-fore unknown weakness in the Alpha1. Appendix A provides the Magma implementation of the attack on Alpha 1.

As for the use of low degree multiples and annihilators, it is clear that even if there exist any, their degree will not be constant and will keep increasing.

6 Algebraic Analysis of MICKEY

The MICKEY family of stream ciphers consists of three ciphers, MICKEY-80 v1 uses a key size of 80-bit and internal state size of 160-bit, MICKEY-80 v2 of key size 80-bit and internal state size 200-bit and MICKEY-128 of key size 128-bit and internal state size of 320-bit. MICKEY uses an LFSR and a Nonlinear Feedback Shift Register (NLFSR). The registers are clocked in an irregular manner determined by some of the internal states bits taken from both registers. Most of the previously known clock-controlled generators used the LFSR for controlling the clocking of their registers. The authors of MICKEY believe that "‘*algebraic attacks usually become possible when the keystream is correlated to one or more linearly clocking registers, whose clocking is either entirely predictable or can be guessed*": which explains why they used both the LFSR and the NLFSR to control clocking in an irregular manner. In this section we show that even with MICKEY's novel technique for irregularly clocking the registers, we can still get an expression that deals with the irregular clocking which also relates its output with its input and holds true for every single clock t.

6.1 Description

Briefly, the MICKEY ciphers consist of two registers R and S each of the same length l. The bits of the registers at time t are labeled as $R_0^t, \ldots, R_{l-1}^t$ and $S_0^t, \ldots, S_{l-1}^t$ respectively. R is a LFSR and S is a NLFSR.

6.2 Algebraic Analysis of the MICKEY Stream Ciphers

Our analysis is based on MICKEY-80 v1 [4], which has an 80 key bit and a total internal state size of 160 bits, but can easily be adapted to the other versions.

We derive an expression that deals with the irregular clocking of R and S. The internal states of R change depending on the bit that controls the clocking of R. If the control bit C_r of register R is 0, then R simply shifts to the right and if the control bit is 1, then as well as shifting to the right, the bits are XORed back.

The control bits C_r and C_s (the control bit of S) are defined as:

$$C_r^t = S_{27}^t + R_{54}^t, \quad \text{and}$$

$$C_s^t = S_{53}^t + R_{26}^t..$$

We assume that the control bits are unknown; however, we can derive an expression that deals with both cases. For C_r

$$R_i^t = R_{i-1}^{t-1} + R_i^{t-1} C_r^{t-1}. \tag{18}$$

Depending on the control bit C_s that controls the clocking of register S, S_i^t can have different values. We can derive an expression that holds true for all clocks and takes into account all possible values for C_s^t. Four sets of fixed, known values are defined in [4]: $\{COMP0_i\}_{i=0}^{l-1}$, $\{COMP1_i\}_{i=0}^{l-1}$, $\{FB0_i\}_{i=0}^{l-1}$ and $\{FB1_i\}_{i=0}^{l-1}$ and are used to define the bits of the NLFSR as follows

$$S_i^t = \hat{S}_i^t + FB0_i \times \mathit{FeedbackBitS}^t \times (C_s^t + 1) + FB1_i \times \mathit{FeedbackBitS}^t \times C_s^t \tag{19}$$

where:

$$\hat{S}_i^t = \begin{cases} S_{i-1}^t + ((S_i^t + COMP0_i) \times (S_{i+1}^t + COMP1_i)) & \text{if } 1 < i < l \\ 0 & \text{if } i = 0 \\ S_{l-1}^t & \text{if } i = l-1. \end{cases}$$

In addition, the value of $\mathit{FeedbackBitS}^t$ is equal to S_{l-1}^t if the key initialization is ignored. The output of the cipher is given by

$$z^t = R_0^t + S_0^t \tag{20}$$

which can be rewritten using equations 18 and 19 as $z^t = f^t =$

$$R_0^t + R_1^t C_r^t + \hat{S}_0^t + FB0_1 \times \mathit{FeedbackBitS}^t \times (C_s^t + 1) + FB1_0 \times \mathit{FeedbackBitS}^t \times C_s^t \tag{21}$$

From equation 21, it can be seen that the degree of the generated equations from the MICKEY stream cipher starts at 2 and increases every clock at least by 3 until it reaches degree $2l$. Guessing l bits will result in degree l equations.

Experimental results show that for MICKEY-80 v1 several annihilators exist. Similarly, annihilators can be determined for the other two versions of MICKEY. The equations can then be linearised and solved. The authors noticed that changing the control bits has no impact on the annihilators. The complexity of algebraic attack still needs to be determined, and will be significantly reduced by the existence of the annihilators. The algebraic structure of the cipher is suggestive that other techniques may exploit this structure.

7 Conclusion

We have presented some theoretical results on the complexity of the algebraic equations generated from mutually clock-controlled feedback shift registers. Such ciphers appear to be particularly resistant to algebraic attacks and we have illustrated this on four well-known ciphers, the bilateral stop-and-go, A5/1, Alpha 1 and MICKEY. We have also presented a relatively simply attack on Alpha 1 based on only 29 bits of keystream. We recommend the use of mutually clocking cipher design in order to resist algebraic attacks. We have also exploited a result (Corollary 1) which allows us to optimize the number of bits guessed in successfully attacking Alpha 1 with only 29 bits of keystream.

References

1. Al-Hinai, S., Batten, L., Colbert, B., Wong, K.: Algebraic attacks on clock controlled stream ciphers. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 1–16. Springer, Heidelberg (2006)
2. Armknecht, F., Krause, M.: Algebraic attacks on combiners with memory. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 162–175. Springer, Heidelberg (2003)
3. Armknecht, F., Carlet, C., Gaborit, P., Kunzli, S., Meier, W., Ruatta, O.: Efficient Computation of Algebraic Immunity for Algebraic and Fast Algebraic Attacks. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 147–164. Springer, Heidelberg (2006)
4. Babbage, S., Dodd, M.: The stream cipher MICKEY (version 1). eSTREAM, ECRYPT Stream Cipher Project,2005/015 (2005)
5. Babbage, S., Dodd, M.: The stream cipher MICKEY-128 (version 1). eSTREAM, ECRYPT Stream Cipher Project,2005/016, (2005)
6. Babbage, S., Dodd, M.: The stream cipher MICKEY (version 2). eSTREAM, ECRYPT Stream Cipher Project (2006)
7. Batten, L.: Algebraic Attacks over $GF(q)$. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 84–91. Springer, Heidelberg (2004)
8. Berger, T., Minier, M.: Two Algebraic Attacks Against the F-FCSRs Using the IV Mode. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 143–154. Springer, Heidelberg (2005)
9. Briceno, M., Goldberg, I., Wagner, D.: A Pedagogical Implementation of A5/1, (May 1999), `http://www.scard.org`
10. Chen, K., Simpson, L., Henricksen, M., Millan, W., Dawson, E.: A Complete Divide and Conquer Attack on the Alpha1 Stream Cipher. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 418–431. Springer, Heidelberg (2004)
11. Cho, J., Pieprzyk, J.: Algebraic attacks on SOBER-t32 and SOBER-t16 without stuttering. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 49–64. Springer, Heidelberg (2004)
12. Courtois, N.: Cryptanalysis of Sfinks. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 261–269. Springer, Heidelberg (2006)
13. Courtois, N.: Fast algebraic attacks on stream ciphers with linear feedback. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 176–194. Springer, Heidelberg (2003)

14. Courtois, N., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 346–359. Springer, Heidelberg (2003)
15. Didier, F.: Using Wiedemanna Algorithm to Compute the Immunity Against Algebraic and Fast Algebraic Attacks. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 236–250. Springer, Heidelberg (2006)
16. Komninos, N., Honary, B., Darnell, M.: An Efficient Stream Cipher for Mobile and Wireless Devices. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 294–300. Springer, Heidelberg (2001)
17. Lee, D., Kim, J., Hong, J., Han, J., Moon, D.: Algebraic Attacks on Summation Generators. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 34–48. Springer, Heidelberg (2004)
18. `http://magma.maths.usyd.edu.au/`
19. Meier, W., Pasalic, E., Carlet, C.: Algebraic attacks and decomposition of Boolean functions. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 474–491. Springer, Heidelberg (2004)
20. Strassen, V.: Gaussian Elimination is Not Optimal. Numerische Mathematik 13, 354–356 (1969)
21. Menicocci, R., Golić, J.: Edit Probability Correlation Attack on Bilateral Stop/Go Generator. In: Walker, M. (ed.) Cryptography and Coding - 6th IMA International Conference 1999. LNCS, vol. 1746, pp. 202–212. Springer, Heidelberg (1999)
22. Wong, K., Colbert, B., Batten, L., Al-Hinai, S.: Algebraic attacks on clock controlled cascade ciphers. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 32–47. Springer, Heidelberg (2006)
23. Zeng, K., Yang, C.H., Rao, T.R.N.: Large primes in stream-cipher cryptography. In: Seberry, J., Pieprzyk, J.P. (eds.) AUSCRYPT 1990. LNCS, vol. 453, pp. 194–205. Springer, Heidelberg (1990)
24. Zhang, X., Pieprzyk, J., Zheng, Y.: On Algebraic Immunity and Annihilators. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 65–80. Springer, Heidelberg (2006)

Four Families of Binary Sequences with Low Correlation and Large Linear Complexity

Jin-Song Wang and Wen-Feng Qi[⋆]

Department of Applied Mathematics, Zhengzhou Information Engineering University, P.O.Box 1001-745,
Zhengzhou, 450002, P.R. China
jinsong.wang@126.com, wenfeng.qi@263.net

Abstract. In this paper, four new families S_1, S_2, S_3 and S_4 of binary sequences of period $2^n - 1$ with low correlation are presented, where S_1, S_3 are defined for odd n, and S_2, S_4 for even n. The family S_1 has six-valued correlations, while S_2 and S_3 have either six-valued correlations or eight-valued correlations, and S_4 has either eight-valued or ten-valued, depending on the choice of parameters.

Keywords: Gold sequence family, Gold-like sequence family, low correlation.

1 Introduction

Families of binary sequences with low correlation have important applications in CDMA communication systems and cryptography[1][2]. Sidelnikov's bound[3] is used to test the optimality of sequence families, which states that for any family of k binary sequences of period N, if $k \geq N$, then

$$C_{max} \geq (2N - 2)^{1/2},$$

where C_{max} is the maximum magnitude of correlation values except for the in-phase autocorrelation value. The Gold family[4] is the best known binary sequence family which satisfies Sidelnikov's bound. It has correlations $2^n - 1, -1, -1 \pm 2^{(n+1)/2}$, where n is odd. But the linear span of Gold sequences is too small to resist attacks based on Berlekamp-Massey algorithm. So the Gold-like families with larger linear span were constructed. The odd case of Gold-like sequence family was discovered by Boztas and Kumar[5], whose correlations are identical to those of Gold sequences. While for even n, Udaya[6] introduced families of binary sequences with correlations $2^n - 1, -1, -1 \pm 2^{n/2}, -1 \pm 2^{n/2+1}$, which corresponds to even case Gold-like sequence family. To get more sequence families with low correlation, Kim and No[7] constructed families S and U. The family S has correlations $2^n - 1, -1, -1 \pm 2^{(n+e)/2}$, while U has correlations $2^n - 1, -1, -1 \pm 2^{n/2}, -1 \pm 2^{n/2+e}$, where n and e are positive integers, $e|n$.

In this paper, we present four new families S_1, S_2, S_3 and S_4 of binary sequences with low correlation and large linear span.

[⋆] This work was supported by the National Natural Science Foundation of China (Grant 60673081) and the National "863" Project(Grant 2006AA01Z417).

2 Preliminaries

Let $GF(2^n)$ be the finite field with 2^n elements, then the trace function from $GF(2^n)$ to $GF(2)$ is defined as[8]

$$tr_1^n(x) = \sum_{i=0}^{n-1} x^{2^i}.$$

Let $S = \{(\boldsymbol{s}_i(t))_{t\geq 0} | 0 \leq i \leq M-1\}$ be a family of M binary sequences with period N, then the correlation function between two sequences $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ is

$$C_{i,j}(\tau) = \sum_{t=0}^{N-1} (-1)^{s_i(t)+s_j(t+\tau)}, \tag{1}$$

where $0 \leq i, j \leq M-1, 0 \leq \tau \leq N-1$. When $i = j$, then $C_{i,i}(\tau)$ is called the autocorrelation function. Denote

$$C_{max} = \max\{|C_{i,j}(\tau)| \mid \text{either } i \neq j, \text{ or } i = j \text{ and } \tau \neq 0\}.$$

Let α be a primitive element of $GF(2^n)$ and $f(x)$ a function from $GF(2^n)$ to $GF(2)$. Replacing x by α^t, then $f(x)$ defines a binary sequence $\boldsymbol{a} = (a(t))_{t\geq 0}$ of period dividing $N = 2^n - 1$, where

$$a(t) = f(\alpha^t), t = 0,\ 1,\ \cdots$$

Remark 1. Without extra explanation, we always assume that α is a primitive element of $GF(2^n)$ in this paper.

If $f_i(x)$ and $f_j(x)$ define sequence $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$, then $C_{i,j}(\tau)$ defined in (1) can also be represented as

$$C_{i,j}(\tau) = \sum_{x\in GF(2^n)^*} (-1)^{f_i(\delta x)+f_j(x)} \hat{=} R_{i,j}(\delta),$$

where $\delta = \alpha^\tau \in GF(2^n)^*$. Let $f(x)$ be a function from $GF(2^n)$ to $GF(2)$, the Hadamard transform of $f(x)$ is defined as:

$$\hat{f}(\lambda) = \sum_{x\in GF(2^n)} (-1)^{f(x)+tr_1^n(\lambda x)},$$

where $\lambda \in GF(2^n)$. It is easy to get

$$\hat{f}(\lambda)^2 = \sum_{w\in GF(2^n)} (-1)^{f(w)+tr_1^n(\lambda w)} \cdot (\sum_{x\in GF(2^n)} (-1)^{f(x)+f(w)+f(x+w)}). \tag{2}$$

Any choice of basis $e_1, e_2, \cdots, e_n$ for $GF(2^n)$ as a vector space over $GF(2)$ determines a one-to-one correspondence from $GF(2^n)$ to $GF(2)^n$ by $x = \sum_{i=1}^{n} x_i e_i \rightarrow$

$\overline{x} = (x_1, x_2, \cdots, x_n)$. In this sense, the Hadamard transform of $f(x)$ is equivalent to the Walsh spectrum of $f(\overline{x})$. Our analysis of correlations is based on the discussion of the corresponding Walsh spectrum.

All sequence families considered in this paper are constructed by the trace function $tr_1^n(x)$ and some function $p(x)$ from $GF(2^n)$ to $GF(2)$. Let

$$f_j(x) = \begin{cases} tr_1^n(\lambda_j x) + p(x), & 0 \le j \le 2^n - 1 \\ tr_1^n(x), & j = 2^n \end{cases},$$

Then the family S is defined as:

$$S = \{(s_j(t))_{t \ge 0} | 0 \le j \le 2^n\},$$

where $s_j(t) = f_j(\alpha^t)$, and $\{\lambda_0, \lambda_1, \cdots, \lambda_{2^n-1}\}$ is an enumeration of the elements in $GF(2^n)$. Furthermore, if $p(0) = 0$, then the correlation function between two sequences defined by $f_i(x)$ and $f_j(x)$ can be rewritten as:

$$\begin{aligned} C_{i,j}(\tau) &= \sum_{x \in GF(2^n)^*} (-1)^{f_i(\delta x) + f_j(x)} \\ &= -(-1)^{g(0)} + \sum_{x \in GF(2^n)} (-1)^{tr_1^n(\lambda x) + g(x)} = -1 + \hat{g}(\lambda), \end{aligned}$$

where $\delta = \alpha^\tau, \lambda = \delta\lambda_i + \lambda_j$ and $g(x) = p(\delta x) + p(x)$.

3 Constructions of Four Binary Sequence Families

3.1 The Sequence Families S_1 and S_2

In this subsection, we present the construction of sequence families S_1 and S_2 using two Gold-like sequences.

Lemma 1. *Let n be an odd integer, $\delta_1 \in GF(2^n) \backslash \{0, 1\}$,*

$$p(x) = \sum_{i=1}^{(n-1)/2} tr_1^n(x^{2^i+1} + (\delta_1 x)^{2^i+1}).$$

Then the distribution of Walsh spectrum of $p(x)$ is given as follows:

$$\hat{p}(\lambda) = \begin{cases} 0, & 2^{n-1} \textit{times} \\ 2^{(n+1)/2}, & 2^{n-2} + 2^{(n-3)/2} \textit{times} \\ -2^{(n+1)/2}, & 2^{n-2} - 2^{(n-3)/2} \textit{times} \end{cases}.$$

Proof. It is clear that

$$\begin{aligned} & p(x) + p(w) + p(x + w) \\ &= tr_1^n\Big(\sum_{i=1}^{(n-1)/2} (xw^{2^i} + wx^{2^i} + \delta_1 x(\delta_1 w)^{2^i} + \delta_1 w(\delta_1 x)^{2^i})\Big). \end{aligned}$$

Since $tr_1^n(wx^{2^i}) = tr_1^n((wx^{2^i})^{2^{n-i}}) = tr_1^n(w^{2^{n-i}}x)$ for $x, w \in GF(2^n)$, then we have

$$\begin{aligned}
&p(x) + p(w) + p(x+w) \\
&= tr_1^n(x \cdot \sum_{i=1}^{(n-1)/2} (w^{2^i} + w^{2^{n-i}})) + tr_1^n(\delta_1 x \cdot \sum_{i=1}^{(n-1)/2} (\delta_1 w^{2^i} + (\delta_1 x)^{2^{n-i}})) \\
&= tr_1^n(x \cdot (tr_1^n(w) + w)) + tr_1^n(\delta_1 x \cdot (tr_1^n(\delta_1 x) + \delta_1 x)) \\
&= tr_1^n(x \cdot L(w)),
\end{aligned}$$

where $L(w) = tr_1^n(w) + w + \delta_1 tr_1^n(\delta_1 w) + \delta_1^2 w$.

Now we consider the number of w in $GF(2^n)$ satisfying $L(w) = 0$. Let $w \in GF(2^n)$ be a solution to $L(w) = 0$, then w must have the form $w = (a + \delta_1 b)(1 + \delta_1)^{-2}$, where $a, b \in GF(2)$ and a, b satisfying

$$\begin{aligned}
&tr_1^n((a + \delta_1 b) \cdot (1 + \delta_1)^{-2}) \\
&= a \cdot tr_1^n((1 + \delta_1)^{-2}) + b \cdot tr_1^n(\delta_1 \cdot (1 + \delta_1)^{-2}) = a \qquad (3) \\
&tr_1^n(\delta_1 \cdot (a + \delta_1 b) \cdot (1 + \delta_1)^{-2}) \\
&= a \cdot tr_1^n(\delta_1 \cdot (1 + \delta_1)^{-2}) + b \cdot tr_1^n(\delta_1^2 \cdot (1 + \delta_1)^{-2}) = b. \qquad (4)
\end{aligned}$$

On the other hand, if a, b satisfy (3) and (4), let $w = (a + \delta_1 b) \cdot (1 + \delta_1)^{-2}$, then

$$\begin{aligned}
L(w) &= tr_1^n(w) + \delta_1 tr_1^n(\delta_1 w) + (1 + \delta_1^2)w \\
&= tr_1^n((a + \delta_1 b) \cdot (1 + \delta_1)^{-2}) + \delta_1 tr_1^n(\delta_1(a + \delta_1 b) \cdot (1 + \delta_1)^{-2}) \\
&+ (1 + \delta_1^2)(a + \delta_1 b) \cdot (1 + \delta_1)^{-2} \\
&= a + \delta_1 b + (1 + \delta_1)^2(a + \delta_1 b) \cdot (1 + \delta_1)^{-2} = 0.
\end{aligned}$$

So the number of w in $GF(2^n)$ to $L(w) = 0$ is equal to that of $(a, b) \in GF(2)^2$ satisfying (3) and (4). Next we calculate the number of those (a, b).

As $\delta_1 \cdot (1 + \delta_1)^{-2} = (1 + \delta_1)^{-1} + (1 + \delta_1)^{-2}$ and $\delta_1^2 \cdot (1 + \delta_1)^{-2} = 1 + (1 + \delta_1)^{-2}$, let $A = tr_1^n((1 + \delta_1)^{-1})$, then (3) and (4) can be represented as

$$\begin{cases} aA^2 + b(A^2 + A) = aA = a \\ a(A^2 + A) + b(1 + A^2) = b(1 + A) = b \end{cases}$$

If $A = 1$, then $(a, b) = (1, 0)$ or $(0, 0)$. Else if $A = 0$, then $(a, b) = (0, 1)$ or $(0, 0)$. That means for any $\delta_1 \in GF(2^n) \backslash \{0, 1\}$, $L(w) = 0$ has exactly two solutions. Let $KerL = \{w | L(w) = 0\}$, then

$$\sum_{x \in GF(2^n)} (-1)^{p(x)+p(w)+p(x+w)} = \sum_{x \in GF(2^n)} (-1)^{tr_1^n(xL(w))} = \begin{cases} 2^n, & \text{if } w \in KerL \\ 0, & \text{otherwise} \end{cases}.$$

So for $\lambda \in GF(2^n)$, by (2), we have

$$\begin{aligned}
\hat{p}(\lambda)^2 &= \sum_{w \in GF(2^n)} (-1)^{p(w)+tr_1^n(\lambda w)} \cdot (\sum_{x \in GF(2^n)} (-1)^{p(x)+p(w)+p(x+w)}) \\
&= 2^n \sum_{w \in KerL} (-1)^{p(w)+tr_1^n(\lambda w)}.
\end{aligned}$$

Since L is a linear transform from $GF(2^n)$ to $GF(2^n)$, $KerL$ forms a vector space over $GF(2)$. $p(w)$ is also a linear transform on $KerL$ since $p(w_1+w_2) = p(w_1)+p(w_2)$ for $w_1, w_2 \in KerL$. Furthermore, $tr_1^n(\lambda w)+p(w)$ is a linear transform on $KerL$, too. Let $\dim(KerL) = k$, then

$$\sum_{w\in KerL} (-1)^{tr_1^n(\lambda w)+p(w)} = \begin{cases} 2^k, & \text{if } tr_1^n(\lambda w)+p(w)=0 \\ 0, & \text{otherwise} \end{cases}.$$

That means $\hat{p}(\lambda)$ is equal to 0, $2^{(n+k)/2}$ and $-2^{(n+k)/2}$. Here $k = 1$, that is, $\hat{p}(\lambda)^2 = 0$ or 2^{n+1}, thus $\hat{p}(\lambda) = 0$, $2^{(n+1)/2}$ or $-2^{(n+1)/2}$. Since the Walsh spectrum $\hat{p}(\lambda)$ satisfies

$$\sum_{\lambda\in GF(2^n)} \hat{p}(\lambda) = 2^n, \sum_{\lambda\in GF(2^n)} \hat{p}^2(\lambda) = 2^{2n},$$

we can get that $\hat{p}(\lambda)$ is equal to $0, 2^{(n+1)/2}$ and $-2^{(n+1)/2}$ with the occurrences $2^{n-1}, 2^{n-2}+2^{(n-3)/2}$ and $2^{n-2}-2^{(n-3)/2}$, respectively.

Lemma 2. *Let n be an odd integer, $\delta_1, \delta_2 \in GF(2^n)\backslash\{0,1\}$ and $\delta_1 \neq \delta_2$, $q(x) = p(x)+p(\delta_2 x)$ a function from $GF(2^n)$ to $GF(2)$, where*

$$p(x) = \sum_{i=1}^{(n-1)/2} tr_1^n(x^{2^i+1} + (\delta_1 x)^{2^i+1}).$$

Set $\delta = (1+\delta_1)^{-1}\ (1+\delta_2)^{-1}$ and

$$A = tr_1^n(\delta), B = tr_1^n(\delta_1\delta), C = tr_1^n(\delta_2\delta), D = tr_1^n(\delta_1\delta_2\delta),$$
$$E = tr_1^n(\delta_1\delta_2\delta^2), F = tr_1^n(\delta_1^2\delta_2\delta^2), G = tr_1^n(\delta_1\delta_2^2\delta^2).$$

If $(A,B,C,F,G) \in \{(0,1,1,0,0),(1,0,1,0,0),(1,1,0,0,0),(1,1,1,0,0)\}$, then the distribution of $\hat{q}(\lambda)$ is given as:

$$\hat{q}(\lambda) = \begin{cases} 0, & 2^n - 2^{n-3}\,times \\ 2^{(n+3)/2}, & 2^{n-4}+2^{(n-5)/2}\,times \\ -2^{(n+3)/2}, & 2^{n-4}-2^{(n-5)/2}\,times \end{cases}.$$

Otherwise, the distribution of $\hat{q}(\lambda)$ is given as:

$$\hat{q}(\lambda) = \begin{cases} 0, & 2^{n-1}\,times \\ 2^{(n+1)/2}, & 2^{n-2}+2^{(n-3)/2}\,times \\ -2^{(n+1)/2}, & 2^{n-2}-2^{(n-3)/2}\,times \end{cases}.$$

Proof. It is clear that

$$\begin{aligned} & q(x)+q(w)+q(x+w) \\ &= \sum_{i=1}^{(n-1)/2} tr_1^n((1+\delta_1^{2^i+1})\cdot(1+\delta_2^{2^i+1})\cdot(w^{2^i+1}+x^{2^i+1}+(x+w)^{2^i+1})) \\ &= tr_1^n(x\cdot L(w)), \end{aligned}$$

where

$$L(w) = tr_1^n(w)+w+\delta_1 tr_1^n(\delta_1 w)+\delta_1^2 w+\delta_2 tr_1^n(\delta_2 w)+\delta_2^2 w+\delta_1\delta_2 tr_1^n(\delta_1\delta_2 w)+(\delta_1\delta_2)^2 w.$$

Next we need to compute the solutions to $L(w) = 0$ to decide the Walsh spectrum of $q(x)$. Similar to the analysis of Lemma 1, the number solutions to $L(w) = 0$ in $GF(2^n)$ is equal to that of parameters $(a, b, c, d) \in GF(2)^4$ satisfying

$$\begin{cases} tr_1^n((a + \delta_1 b + \delta_2 c + \delta_1\delta_2 d) \cdot \delta^2) = a \\ tr_1^n((\delta_1 a + \delta_1^2 b + \delta_1\delta_2 c + \delta_1^2\delta_2 d) \cdot \delta^2) = b \\ tr_1^n((\delta_2 a + \delta_1\delta_2 b + \delta_2^2 c + \delta_1\delta_2^2 d) \cdot \delta^2) = c \\ tr_1^n((\delta_1\delta_2 a + \delta_1^2\delta_2 b + \delta_1\delta_2^2 c + (\delta_1\delta_2)^2 d) \cdot \delta^2) = d \end{cases} \tag{5}$$

Based on the definition of $A - G$, we have

$$\begin{aligned} &tr_1^n(\delta^2) = A^2, tr_1^n(\delta_1\delta_2 \cdot \delta^2) = E, tr_1^n(\delta_1^2 \cdot \delta^2) = B^2, \\ &tr_1^n(\delta_1^2\delta_2 \cdot \delta^2) = F, tr_1^n(\delta_1\delta_2^2 \cdot \delta^2) = G, \\ &tr_1^n(\delta_1^2\delta_2^2 \cdot \delta^2) = D^2, A + B + C + D = tr_1^n(1) = 1. \end{aligned}$$

From

$$\delta_1 \cdot \delta^2 = \delta_1 \cdot \delta + \delta_1^2 \cdot \delta^2 + \delta_1\delta_2 \cdot \delta^2 + \delta_1^2\delta_2 \cdot \delta^2$$

and

$$\delta_2 \cdot \delta^2 = \delta_2 \cdot \delta + \delta_1\delta_2 \cdot \delta^2 + \delta_2^2 \cdot \delta^2 + \delta_1\delta_2^2 \cdot \delta^2$$

we have

$$tr_1^n(\delta_1 \cdot \delta^2) = B + B^2 + E + F$$

and

$$tr_1^n(\delta_2 \cdot \delta^2) = C + C^2 + E + G$$

Furthermore, since

$$\delta_1\delta_2 \cdot \delta^2 = \delta_1\delta_2 \cdot \delta + \delta_1^2\delta_2 \cdot \delta^2 + \delta_1\delta_2^2 \cdot \delta^2 + (\delta_1\delta_2)^2 \cdot \delta^2,$$

we have $D + F + G + D^2 = E$.

Thus A, B, C, F, G can be regarded as free variables. If δ_1, δ_2 satisfy $(A, B, C, F, G) \in \{(0, 1, 1, 0, 0), (1, 0, 1, 0, 0), (1, 1, 0, 0, 0), (1, 1, 1, 0, 0)\}$, then the number of solutions is 8, thus $\hat{q}(\lambda)$ is equal to $0, 2^{(n+3)/2}$ and $-2^{(n+3)/2}$ with the occurrences $2^n - 2^{n-3}, 2^{n-4} + 2^{(n-5)/2}$ and $2^{n-4} - 2^{(n-5)/2}$, respectively. Otherwise $\hat{q}(\lambda)$ is equal to $0, 2^{(n+1)/2}$ and $-2^{(n+1)/2}$ with the occurrences 2^{n-1}, $2^{n-2} + 2^{(n-3)/2}$ and $2^{n-2} - 2^{(n-3)/2}$, respectively.

Theorem 1. *Let n be an odd integer and $\delta_1 \in GF(2^n)\backslash\{0, 1\}$,*

$$b_1(x) = \sum_{i=1}^{(n-1)/2} tr_1^n(x^{2^i+1} + (\delta_1 x)^{2^i+1}),$$

the family $S_1 = \{\boldsymbol{s}_{1,j} | j = 0, 1, \cdots, 2^n\}$ is given by

$$\boldsymbol{s}_{1,j}(\alpha^t) = \begin{cases} tr_1^n(\lambda_j\alpha^t) + b_1(\alpha^t), 0 \le j \le 2^n - 1 \\ tr_1^n(\alpha^t), \qquad\qquad\qquad j = 2^n \end{cases},$$

where $\{\lambda_0, \lambda_1, \cdots, \lambda_{2^n-1}\}$ *is an enumeration of the elements in* $GF(2^n)$*. Then the distribution of correlation values of* S_1 *is given as follows:*

$$C_{i,j}(\tau) = \begin{cases} 2^n - 1, & 2^n + 1 \textit{ times} \\ -1, & 2^{3n-1} + 2^{3n-4} - 2^{3n-6} + 2^{2n} - 2^n - 2 \textit{ times} \\ -1 + 2^{(n+1)/2}, & (2^{n-2} + 2^{(n-3)/2})(2^{2n} - 2^{2n-3} - 2) \textit{ times} \\ -1 - 2^{(n+1)/2}, & (2^{n-2} - 2^{(n-3)/2})(2^{2n} - 2^{2n-3} - 2) \textit{ times} \\ -1 + 2^{(n+3)/2}, & (2^{n-4} + 2^{(n-5)/2})2^{2n-3} \textit{ times} \\ -1 - 2^{(n+3)/2}, & (2^{n-4} - 2^{(n-5)/2})2^{2n-3} \textit{ times} \end{cases}.$$

Proof. Let $\delta = \alpha^\tau$, then $C_{i,j}(\tau) = R_{i,j}(\delta)$. This proof can be divided into five cases.

Case 1): $\delta = 1$, $i = j$:

It is a trivial case and thus

$$R_{i,j}(\delta) = 2^n - 1,\ 2^n + 1 \text{ times.}$$

Case 2): $\delta \neq 1$, $i = j = 2^n$:

The sequence $\boldsymbol{s}_{1,2^n}$ is an m-sequence and

$$R_{i,j}(\delta) = -1,\ 2^n - 2 \text{ times.}$$

Case 3): $\delta = 1, 0 \leq i, j \leq 2^n - 1$ and $i \neq j$:

From the linearity of the trace function, we have

$$R_{i,j}(\delta) = \sum_{x \in GF(2^n)^*} (-1)^{s_{1,i}(\delta x) + s_{1,j}(x)} = (-1)^{tr_1^n[(\lambda_i + \lambda_j)x]} = -1,$$

$2^n(2^n - 1)$ times.

Case 4): $i = 2^n, j \neq 2^n$(or $i \neq 2^n, j = 2^n$):

For a fixed δ, we have

$$\begin{aligned} R_{2^n,j}(\delta) &= \sum_{x \in GF(2^n)^*} (-1)^{tr_1^n[(\delta + \lambda_j)x] + b_1(x)} \\ &= -1 + \sum_{x \in GF(2^n)} (-1)^{tr_1^n(\lambda x) + b_1(x)} = -1 + \hat{b}_1(\lambda), \end{aligned}$$

where $\lambda = \delta + \lambda_j$. The distribution of Walsh spectrum of $b_1(x)$ has already presented in Lemma 1. As δ varies over $GF(2^n)\backslash\{0\}$, the distribution is:

$$R_{2^n,j}(\delta) = \begin{cases} -1, & 2^{n-1}(2^n - 1)\text{times} \\ -1 + 2^{(n+1)/2}, & (2^{n-2} + 2^{(n-3)/2})(2^n - 1)\text{times} \\ -1 - 2^{(n+1)/2}, & (2^{n-2} - 2^{(n-3)/2})(2^n - 1)\text{times} \end{cases}.$$

The case $i \neq 2^n$ and $j = 2^n$ has the same distribution.

Case 5): $\delta \in GF(2^n)\backslash\{0, 1\}$, $0 \leq i, j \leq 2^n - 1$:

In this case, from (2) we have

$$R_{i,j}(\delta) = \sum_{x \in GF(2^n)^*} (-1)^{tr_1^n[(\delta\lambda_i + \lambda_j)x] + b_1(\delta x) + b_1(x)} = -1 + \hat{q}(\lambda),$$

where $q(x) = b_1(\delta x) + b_1(x), \lambda = \delta\lambda_i + \lambda_j$. For a fixed δ and λ_i, the distribution of Walsh spectrum of $q(x)$ is already given in Lemma 2. As $\delta \in GF(2^n)\backslash\{0,1\}$, let $y = (1+\delta)^{-1}$, then $y \in GF(2^n)\backslash\{0,1\}$. Replacing $(1+\delta)^{-1}$ in the variables $A\ -G$ of Lemma 2 by y, we have

$$A = tr_1^n((1+\delta_1)^{-1}\cdot y), B = A + tr_1^n(y), C = A + tr_1^n((1+\delta_1)^{-1}),$$

$$F = A + tr_1^n((1+\delta_1)^{-2}\cdot y), G = A + tr_1^n((1+\delta_1)^{-1}\cdot y^2).$$

Define $Y = \{(0,1,1,0,0),(1,0,1,0,0),(1,1,0,0,0),(1,1,1,0,0)\}$. As we know from Lemma 2, the value of $\hat{q}(\lambda)$ relies on whether $(A,B,C,F,G) \in Y$ or not. Notice that if $y = 0$, then $(A,B,C,F,G) = (0,0,tr_1^n(1+\delta_1)^{-1},0,0)$ and if $y = 1$, then $(A,B,C,F,G) = (tr_1^n(1+\delta_1)^{-1},\ 1{+}tr_1^n(1+\delta_1)^{-1},0,0,0)$. Both cases do not satisfy $(A,B,C,F,G) \in Y$. Now we consider the following two sub-cases of the number of y in $GF(2^n)\backslash\{0,1\}$ such that $(A,B,C,F,G) \in Y$.

(i) $tr_1^n((1+\delta_1)^{-1}) = 0$:
If $A = 0$, since $F = 0$ and $G = 0$, we have:

$$tr_1^n((1+\delta_1)^{-2}\cdot y) = 0,\ tr_1^n((1+\delta_1)^{-1}\cdot y^2) = 0,$$

Therefore

$$B = A + tr_1^n(y) = tr_1^n(y),$$

$$C = A + tr_1^n((1+\delta_1)^{-1}) = tr_1^n((1+\delta_1)^{-1}) = 0.$$

Then there exists no such y satisfying $(A,B,C,F,G) \in Y$.
If $A = 1$, since $F = 0$ and $G = 0$, we have:

$$tr_1^n((1+\delta_1)^{-2}\cdot y) = 1,\ tr_1^n((1+\delta_1)^{-1}\cdot y^2) = 1,$$

Therefore

$$B = A + tr_1^n(y) = 1 + tr_1^n(y),$$

$$C = A + tr_1^n((1+\delta_1)^{-1}) = 1 + tr_1^n((1+\delta_1)^{-1}) = 1.$$

As $(A,B,C,F,G) \in Y$, we can get $(A,B,C,F,G) = (1,0,1,0,0)$ or $(1,1,1,0,$ $0)$. Thus y must satisfy $tr_1^n((1+\delta_1)^{-2}\cdot y) = 1$, $tr_1^n((1+\delta_1)^{-1}\cdot y^2) = 1$ and $tr_1^n((1+\delta_1)^{-1}\cdot y) = 1$. As $\delta_1 \in GF(2^n)\backslash\{0,1\}$, then $(1+\delta_1)^{-1}$, $(1+\delta_1)^{-2}$, and $(1+\delta_1)^{-4}$ are different from each other. Therefore the number of y in $GF(2^n)$ such that $(A,B,C,F,G) \in Y$ is 2^{n-3}.

(ii) $tr_1^n((1+\delta_1)^{-1}) = 1$:
Similar to the analysis of (i) we have that if $(A,B,C,F,G) \in Y$, then $(A,B,C,$ $F,G) = (0,1,1,0,0)$ or $(1,1,0,0,0)$. For $(A,B,C,F,G) = (0,1,1,0,0)$, y must satisfy $tr_1^n(y) = 1$, $tr_1^n((1+\delta_1)^{-1}\cdot y) = 0$, $tr_1^n((1+\delta_1)^{-2}\cdot y) = 0$, and $tr_1^n((1+\delta_1)^{-1}\cdot y^2) = 0$. As $\delta_1 \in GF(2^n)\backslash\{0,1\}$, then $1, (1+\delta_1)^{-1}$, $(1+\delta_1)^{-2}$ and $(1+\delta_1)^{-4}$ are different from each other. Therefore the number of y in $GF(2^n)$ such that

$(A, B, C, F, G) = (0, 1, 1, 0, 0)$ is 2^{n-4}. Similarly, the number of y in $GF(2^n)$ such that $(A, B, C, F, G) = (1, 1, 0, 0, 0)$ is also 2^{n-4}.

Thus no matter what $tr_1^n((1 + \delta_1)^{-1})$ equals, the number of y in $GF(2^n)$ such that $(A, B, C, F, G) \in Y$ is 2^{n-3}. When $y = 0, 1$, $(A, B, C, F, G) \notin Y$, so the number of $y \in GF(2^n)\backslash\{0, 1\}$ such that $(A, B, C, F, G) \in Y$ is 2^{n-3}, and the number such that $(A, B, C, F, G) \notin Y$ is $2^n - 2^{n-3} - 2$. Therefore, the distribution of correlation can be given as follows:

$$R_{i,\ j}(\delta) = \begin{cases} -1, & 2^{2n-1}(2^n - 2^{n-3} - 2) + (2^n - 2^{n-3})2^{2n-3}\text{times} \\ -1 + 2^{(n+1)/2}, & (2^{n-2} + 2^{(n-3)/2})2^n(2^n - 2^{n-3} - 2)\text{times} \\ -1 - 2^{(n+1)/2}, & (2^{n-2} - 2^{(n-3)/2})2^n(2^n - 2^{n-3} - 2)\text{times} \\ -1 + 2^{(n+3)/2}, & (2^{n-4} + 2^{(n-5)/2})2^{2n-3} \text{ times} \\ -1 - 2^{(n+3)/2}, & (2^{n-4} - 2^{(n-5)/2})2^{2n-3} \text{ times} \end{cases}.$$

Combining the above five cases, the distribution of the correlation values for the sequence family S_1 can be obtained.

We only need to change $b(A+1) = b$ in Lemma 1 into $bA = b$ and $A+B+C+D=1$ in Lemma 2 into $A+B+C+D = 0$ to get the proofs of Lemma 3 and Lemma 4.

Lemma 3. *Let n be an even integer, $\delta_1 \in GF(2^n)\backslash\{0, 1\}$, and $p(x)$ a function from $GF(2^n)$ to $GF(2)$, where*

$$p(x) = tr_1^{n/2}(x^{2^{n/2}+1} + (\delta_1 x)^{2^{n/2}+1}) + \sum_{i=1}^{n/2-1} tr_1^n(x^{2^i+1} + (\delta_1 x)^{2^i+1}).$$

If $tr_1^n((1 + \delta_1)^{-1}) = 1$, then the distribution of $\hat{p}(\lambda)$ is given as:

$$\hat{p}(\lambda) = \begin{cases} 0, & 2^n - 2^{n-2}\textit{times} \\ 2^{n/2+1}, & 2^{n-3} + 2^{n/2-2}\textit{times} \\ -2^{n/2+1}, & 2^{n-3} - 2^{n/2-2}\textit{times} \end{cases}.$$

If $tr_1^n((1 + \delta_1)^{-1}) = 0$, then the distribution of $\hat{p}(\lambda)$ is given as:

$$\hat{p}(\lambda) = \begin{cases} 2^{n/2}, & 2^{n-1} + 2^{n/2-1}\textit{times} \\ -2^{n/2}, & 2^{n-1} - 2^{n/2-1}\textit{times} \end{cases}.$$

Lemma 4. *Let n be an even integer, $\delta_1, \delta_2 \in GF(2^n)\backslash\{0, 1\}$ and $\delta_1 \neq \delta_2, q(x) = p(x) + p(\delta_2 x)$ a function from $GF(2^n)$ to $GF(2)$, where*

$$p(x) = tr_1^{n/2}(x^{2^{n/2}+1} + (\delta_1 x)^{2^{n/2}+1}) + \sum_{i=1}^{n/2-1} tr_1^n(x^{2^i+1} + (\delta_1 x)^{2^i+1}).$$

Set $\delta = (1 + \delta_1)^{-1}\cdot(1 + \delta_2)^{-1}$ and

$$A = tr_1^n(\delta), B = tr_1^n(\delta_1 \cdot \delta), C = tr_1^n(\delta_2 \cdot \delta), D = tr_1^n(\delta_1\delta_2 \cdot \delta),$$

$$E = tr_1^n(\delta_1\delta_2 \cdot \delta^2), F = tr_1^n(\delta_1^2\delta_2 \cdot \delta^2), G = tr_1^n(\delta_1\delta_2^2 \cdot \delta^2).$$

Then

(i) if $(A, B, C, F, G) = (1, 1, 1, 0, 0)$, then the distribution of $\hat{q}(\lambda)$ is given as:

$$\hat{q}(\lambda) = \begin{cases} 0, & 2^n - 2^{n-4} times \\ 2^{n/2+2}, & 2^{n-5} + 2^{n/2-3} times \\ -2^{n/2+2}, & 2^{n-5} - 2^{n/2-3} times \end{cases},$$

(ii) if $(A, B, C, F, G) \in \{(0,1,0,0,0), (0,1,0,0,1), (1,0,1,0,0), (1,0,1,0,1), (1,1,1,0,1), (1,1,1,1,0), (1,1,1,1,1), (0,0,1,0,0), (0,0,1,1,0), (0,1,1,0,0), (0,1,1,1,1), (1,0,0,0,0), (1,0,0,1,1), (1,1,0,0,0), (1,1,0,1,0)\}$, then the distribution of $\hat{q}(\lambda)$ is given as:

$$\hat{q}(\lambda) = \begin{cases} 0, & 2^n - 2^{n-2} times \\ 2^{n/2+1}, & 2^{n-3} + 2^{n/2-2} times \\ -2^{n/2+1}, & 2^{n-3} - 2^{n/2-2} times \end{cases},$$

(iii) otherwise, the distribution of $\hat{q}(\lambda)$ is given as:

$$\hat{q}(\lambda) = \begin{cases} 2^{n/2}, & 2^{n-1} + 2^{n/2-1} times \\ -2^{n/2}, & 2^{n-1} - 2^{n/2-1} times \end{cases}.$$

In fact, we can distinguish the cases of $tr_1^n((1+\delta_1)^{-1}) = 1$ or 0 in (ii) and (iii). From Lemma 3 and Lemma 4, we have the distribution of correlation values of S_2.

Theorem 2. *Let n be an even integer, $\delta_1 \in GF(2^n) \backslash \{0, 1\}$,*

$$b_2(x) = tr_1^{n/2}(x^{2^{n/2}+1} + (\delta_1 x)^{2^{n/2}+1}) + \sum_{i=1}^{n/2-1} tr_1^n(x^{2^i+1} + (\delta_1 x)^{2^i+1}),$$

the family $S_2 = \{\boldsymbol{s}_{2,j} | j = 0, 1, \cdots, 2^n\}$ is given by

$$\boldsymbol{s}_{2,j}(\alpha^t) = \begin{cases} tr_1^n(\lambda_j \alpha^t) + b_2(\alpha^t), & 0 \le j \le 2^n - 1 \\ tr_1^n(\alpha^t), & j = 2^n \end{cases},$$

where $\{\lambda_0, \ \lambda_1, \ \cdots, \ \lambda_{2^n-1}\}$ is an enumeration of the elements in $GF(2^n)$.

If $tr_1^n((1+\delta_1)^{-1}) = 0$, then the distribution of correlation values of S_2 is given as follows:

$$C_{i,\ j}(\tau) = \begin{cases} 2^n - 1, & 2^n + 1 times \\ -1, & 2^{3n-1} - 2^{3n-3} + 2^{3n-6} - 2^{3n-8} + 2^{2n} - 2 times \\ -1 + 2^{n/2}, & (2^{n-1} + 2^{n/2-1})(2^{2n-1} - 2) times \\ -1 - 2^{n/2}, & (2^{n-1} - 2^{n/2-1})(2^{2n-1} - 2) times \\ -1 + 2^{n/2+1}, & (2^{n-3} + 2^{n/2-2})(2^{2n-1} - 2^{2n-4}) times \\ -1 - 2^{n/2+1}, & (2^{n-3} - 2^{n/2-2})(2^{2n-1} - 2^{2n-4}) times \\ -1 + 2^{n/2+2}, & (2^{n-5} + 2^{n/2-3}) 2^{2n-4} times \\ -1 - 2^{n/2+2}, & (2^{n-5} - 2^{n/2-3}) 2^{2n-4} times \end{cases}.$$

If $tr_1^n((1+\delta_1)^{-1}) = 1$, *then the distribution of correlation values of* S_2 *is given as follows:*

$$C_{i,\ j}(\tau) = \begin{cases} 2^n - 1, & 2^n + 1 times \\ -1, & 2^{3n-1} - 2^{3n-3} + 2^{2n} - 2^{n+1} + 2^{n-1} - 2 times \\ -1 + 2^{n/2}, & (2^{n-1} + 2^{n/2-1})2^{2n-1} times \\ -1 - 2^{n/2}, & (2^{n-1} - 2^{n/2-1})2^{2n-1} times \\ -1 + 2^{n/2+1}, & (2^{n-3} + 2^{n/2-2})(2^{2n-1} - 2) times \\ -1 - 2^{n/2+1}, & (2^{n-3} - 2^{n/2-2})(2^{2n-1} - 2) times \end{cases}.$$

3.2 The Sequence Families S_3 and S_4

In this subsection, we present the construction of sequence families S_3 and S_4 using three Gold-like sequences. First we give two lemmas to calculate the correlation values of S_3.

Lemma 5. *Let* n *be an odd integer,* $\delta_1, \delta_2, \delta_1 + \delta_2 \in GF(2^n)\backslash\{0,1\}$, $p(x)$ *a function from* $GF(2^n)$ *to* $GF(2)$, *where*

$$p(x) = \sum_{i=1}^{(n-1)/2} tr_1^n(x^{2^i+1} + (\delta_1 x)^{2^i+1} + (\delta_2 x)^{2^i+1}).$$

Then the distribution of $\hat{p}(\lambda)$ *is given as:*

$$\hat{p}(\lambda) = \begin{cases} 0, & 2^{n-1} times \\ 2^{(n+1)/2}, & 2^{n-2} + 2^{(n-3)/2} times \\ -2^{(n+1)/2}, & 2^{n-2} - 2^{(n-3)/2} times \end{cases}.$$

Proof. It is clear that

$$\begin{aligned} & p(x) + p(w) + p(x+w) \\ = & \sum_{i=1}^{(n-1)/2} tr_1^n((1 + \delta_1^{2^i+1} + \delta_2^{2^i+1}) \cdot (w^{2^i+1} + x^{2^i+1} + (x+w)^{2^i+1})) \\ = & tr_1^n(x \cdot L(w)), \end{aligned}$$

where $L(w) = tr_1^n(w) + w + \delta_1 tr_1^n(\delta_1 w) + \delta_1^2 w + \delta_2 tr_1^n(\delta_2 w) + \delta_2^2 w$.

As in the proof of Lemma 1, we only need to compute the solutions to $L(w) = 0$ to decide the Walsh spectrum of $p(x)$. Similar to the proof of Lemma 1, we know that the number of solutions to $L(w) = 0$ is equal to that of parameter $(a, b, c) \in GF(2)^3$ satisfying

$$\begin{cases} tr_1^n((a + \delta_1 b + \delta_2 c) \cdot \delta^2) = a \\ tr_1^n((\delta_1 a + \delta_1^2 b + \delta_1\delta_2 c) \cdot \delta^2) = b \\ tr_1^n((\delta_2 a + \delta_1\delta_2 b + \delta_2^2 c) \cdot \delta^2) = c \end{cases} \tag{6}$$

Let $tr_1^n(\delta) = A$, $tr_1^n(\delta_1 \cdot \delta) = B, tr_1^n(\delta_2 \cdot \delta) = C$, $tr_1^n(\delta_1\delta_2 \cdot \delta^2) = D$. Since $\delta_1 \cdot \delta^2 = \delta_1 \cdot \delta + \delta_1\delta_2 \cdot \delta^2 + \delta_1^2 \cdot \delta^2$ and $\delta_2 \cdot \delta^2 = \delta_2 \cdot \delta + \delta_1\delta_2 \cdot \delta^2 + \delta_2^2 \cdot \delta^2$, (6) can be rewritten as:

$$\begin{cases} aA + bD + cD = a \\ aD + bB + cD = b \\ aD + bD + cC = c \end{cases}$$

As n is an odd integer, we have $A+B+C = tr_1^n(1)$=1. Solve the above equations, (6) always has two solutions. Thus $\hat{p}(\lambda)$ is equal to 0, $2^{(n+1)/2}$ and $-2^{(n+1)/2}$ with the occurrences 2^{n-1}, $2^{n-2} + 2^{(n-3)/2}$ and $2^{n-2} - 2^{(n-3)/2}$, respectively.

Let Ω be the set of $(\delta_1, \delta_2) \in GF(2^n) \times GF(2^n)$ which satisfy the following conditions:

(1)$\delta_1, \delta_2 \notin \{0, 1\}$,
(2)$\delta_1 \notin \{\delta_2,\ 1+\delta_2, \delta_2^{-1}, \delta_2^{1/2}, \delta_2^2\}$,
(3) $(1 + \delta_1 + \delta_2)^2 \notin \{\delta_1\delta_2^{-1}, \delta_1^{-1}\delta_2, \delta_1, \delta_2, \delta_1^{-1}\delta_2^2, \delta_1^2\delta_2, \delta_1^4\delta_2^{-2}, \delta_1^3\delta_2^{-1}, \delta_1^3, \delta_1^4\delta_2^{-1}\}$.

Remark 2. As a matter of fact, given $\delta_1 \in GF(2^n)\backslash\{0, 1\}$, the number of δ_2 that $(\delta_1, \delta_2) \notin \Omega$ is at most 30. When $n \geq 12$, the proportion of Ω in $GF(2^n)\times GF(2^n)$ is greater than 99.2%. Thus we can suppose $(\delta_1, \delta_2) \in \Omega$ in S_3 and S_4 when δ_1, δ_2 are chosen randomly.

Lemma 6. *Let n be an odd integer, $\delta_1, \delta_2, \delta_3 \in GF(2^n)\backslash\{0, 1\}$ and $(\delta_1, \delta_2) \in \Omega$, then $1+\delta_1+\delta_2 \neq 0$. Suppose that $q(x) = p(x)+p(\delta_3 x)$ is a function from $GF(2^n)$ to $GF(2)$, where*

$$p(x) = \sum_{i=1}^{(n-1)/2} tr_1^n(x^{2^i+1} + (\delta_1 x)^{2^i+1} + (\delta_2 x)^{2^i+1}).$$

Let $\delta = (1 + \delta_1 + \delta_2)^{-1}, \omega = tr_1^n(\delta), \beta = tr_1^n(\delta_1 \cdot \delta), \gamma = tr_1^n(\delta_2 \cdot \delta), \theta = tr_1^n(\delta_1\delta_2 \cdot \delta^2)$, then $\omega + \beta + \gamma = 1$. For any $\delta_3 \in GF(2^n)\backslash\{0, 1\}$, if $(\omega, \beta, \gamma, \theta) \in \{(1, 1, 1, 0), (0, 0, 1, 1), (0, 1, 0, 1), (1, 0, 0, 1)\}$, then $\hat{q}(\lambda)$ is equal to 0, $\pm 2^{(n+1)/2}$, $\pm 2^{(n+3)/2}$. If $(\omega, \beta, \gamma, \theta) \in \{(1, 1, 1, 1), (0, 0, 1, 0), (0, 1, 0, 0), (1, 0, 0, 0)\}$, then $\hat{q}(\lambda)$ is equal to 0, $\pm 2^{(n+1)/2}$, $\pm 2^{(n+3)/2}$, $\pm 2^{(n+5)/2}$.

The proof of Lemma 6 can be got similar to the proof of Lemma 2, here we omit the proof process for the limitation of space.

Theorem 3. *Let n be an odd integer, $(\delta_1, \delta_2) \in \Omega$, then $1+\delta_1+\delta_2 \neq 0$. Suppose that*

$$b_3(x) = \sum_{i=1}^{(n-1)/2} tr_1^n(x^{2^i+1} + (\delta_1 x)^{2^i+1} + (\delta_2 x)^{2^i+1}),$$

the family S_3=$\{\boldsymbol{s}_{3,j} | j = 0, 1, \cdots, 2^n\}$ is given by

$$\boldsymbol{s}_{3,j}(\alpha^t) = \begin{cases} tr_1^n(\lambda_j \alpha^t) + b_3(\alpha^t), 0 \leq j \leq 2^n - 1 \\ tr_1^n(\alpha^t), \qquad\qquad\qquad j = 2^n \end{cases},$$

where $\{\lambda_0, \lambda_1, \cdots, \lambda_{2^n-1}\}$ is an enumeration of the elements in $GF(2^n)$. Let $\delta = (1 + \delta_1 + \delta_2)^{-1}, \omega = tr_1^n(\delta), \beta = tr_1^n(\delta_1 \cdot \delta), \gamma = tr_1^n(\delta_2 \cdot \delta), \theta = tr_1^n(\delta_1\delta_2 \cdot \delta^2)$, then $\omega + \beta + \gamma = 1$. If $(\omega, \beta, \gamma, \theta) \in \{(1, 1, 1, 0), (0, 0, 1, 1), (0, 1, 0, 1), (1, 0, 0, 1)\}$, *then S_3 has correlations $2^n-1, -1, -1\pm 2^{(n+1)/2}, -1\pm 2^{(n+3)/2}$. If $(\omega,\beta,\gamma,\theta)\in\{(1,1,1,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)\}$, then S_3 has correlations $2^n-1, -1$, $-1\pm 2^{(n+1)/2}$, $-1\pm 2^{(n+3)/2}$, $-1\pm 2^{(n+5)/2}$. For example, if $(\omega,\beta,\gamma,\theta)=(0,0,1,1)$, then the distribution of correlation values of S_3 is given as follows:*

$$C_{i,j}(\tau)=\begin{cases} 2^n-1, & 2^n+1 \text{ times}\\ -1, & 2^{3n-1}+2^{3n-4}-2^{3n-6}+2^{2n}-2^n-2 \text{ times}\\ -1+2^{(n+1)/2}, & (2^{n-2}+2^{(n-3)/2})(2^{2n}-2^{2n-3}-2) \text{ times}\\ -1-2^{(n+1)/2}, & (2^{n-2}-2^{(n-3)/2})(2^{2n}-2^{2n-3}-2) \text{ times}\\ -1+2^{(n+3)/2}, & (2^{n-4}+2^{(n-5)/2})2^{2n-3} \text{ times}\\ -1-2^{(n+3)/2}, & (2^{n-4}-2^{(n-5)/2})2^{2n-3} \text{ times}\end{cases}.$$

The proof of Theorem 3 is similar to that of Theorem 1, the only difference is the calculation of the distribution of correlation values in Case 5. Here we omit the proof.

If we change $A+B+C=1$ in Lemma 5 into $A+B+C=0$ in Lemma 6 and $\omega+\beta+\gamma=1$ into $\omega+\beta+\gamma=0$, then the proofs of Lemma 7 and Lemma 8 follow correspondingly.

Lemma 7. *Let n be an even integer, $\delta_1,\delta_2,\delta_1+\delta_2\in GF(2^n)\backslash\{0,1\}$, $p(x)$ a function from $GF(2^n)$ to $GF(2)$, where*

$$\begin{aligned}p(x)&=tr_1^{n/2}(x^{2^{n/2}+1}+(\delta_1x)^{2^{n/2}+1}+(\delta_2x)^{2^{n/2}+1})\\&+\sum_{i=1}^{n/2-1}tr_1^n(x^{2^i+1}+(\delta_1x)^{2^i+1}+(\delta_2x)^{2^i+1}).\end{aligned}$$

Let $\delta=(1+\delta_1+\delta_2)^{-1}$, if δ_1 and δ_2 satisfy $tr_1^n(\delta)=1$, $tr_1^n(\delta_1\cdot\delta)=1$, and $tr_1^n(\delta_2\cdot\delta)=0$, then the distribution of $\hat{p}(\lambda)$ is given as:

$$\hat{p}(\lambda)=\begin{cases}0, & 2^n-2^{n-2} \text{times}\\ 2^{n/2+1}, & 2^{n-3}+2^{n/2-2} \text{times}\\ -2^{n/2+1}, & 2^{n-3}-2^{n/2-2} \text{ times}\end{cases}.$$

Otherwise, the distribution of $\hat{p}(\lambda)$ is given as:

$$\hat{p}(\lambda)=\begin{cases}2^{n/2}, & 2^{n-1}+2^{n/2-1} \text{times}\\ -2^{n/2}, & 2^{n-1}-2^{n/2-1} \text{times}\end{cases}.$$

Lemma 8. *Let n be an even integer, $\delta_1,\delta_2,\delta_3\in GF(2^n)\backslash\{0,1\}$ and $(\delta_1,\delta_2)\in\Omega$, then $1+\delta_1+\delta_2\neq 0$. Suppose that $q(x)=p(x)+p(\delta_3x)$ is a function from $GF(2^n)$ to $GF(2)$, where*

$$\begin{aligned}p(x)&=tr_1^{n/2}(x^{2^{n/2}+1}+(\delta_1x)^{2^{n/2}+1}+(\delta_2x)^{2^{n/2}+1})\\&+\sum_{i=1}^{n/2-1}tr_1^n(x^{2^i+1}+(\delta_1x)^{2^i+1}+(\delta_2x)^{2^i+1}).\end{aligned}$$

Let $\delta=(1+\delta_1+\delta_2)^{-1}, \omega=tr_1^n(\delta), \beta=tr_1^n(\delta_1\cdot\delta), \gamma=tr_1^n(\delta_2\cdot\delta), \theta=tr_1^n(\delta_1\delta_2\cdot\delta^2)$, then $\omega+\beta+\gamma=0$. For any $\delta_3\in GF(2^n)\backslash\{0,1\}$, if $(\omega,\beta,\gamma,\theta)=(0,0,0,0)$, *then* $\hat{q}(\lambda)$ *is equal to 0,* $\pm 2^{n/2}, \pm 2^{n/2+1}, \pm 2^{n/2+2}$*, or* $\pm 2^{n/2+3}$*. If* $(\omega,\beta,\gamma,\theta) \in \{(0,0,0,1),(0,1,1,0),(0,1,1,1),(1,0,1,0),(1,0,1,1),(1,1,0,0),(1,1,0,1)\}$*, then* $\hat{q}(\lambda)$ *is equal to* $0, \pm 2^{n/2}, \pm 2^{n/2+1}$*, or* $\pm 2^{n/2+2}$*.*

Theorem 4. *Let* n *be an even integer,* $(\delta_1,\delta_2) \in \Omega$*, then* $1+\delta_1+\delta_2 \neq 0$*. Suppose that*

$$b_4(x) = tr_1^{n/2}(x^{2^{n/2}+1} + (\delta_1 x)^{2^{n/2}+1} + (\delta_2 x)^{2^{n/2}+1}) + \sum_{i=1}^{n/2-1} tr_1^n(x^{2^i+1} + (\delta_1 x)^{2^i+1} + (\delta_2 x)^{2^i+1}),$$

the family $S_4 = \{\boldsymbol{s}_{4,j} | j = 0,1,\cdots,2^n\}$ *is given by*

$$\boldsymbol{s}_{4,j}(\alpha^t) = \begin{cases} tr_1^n(\lambda_j \alpha^t) + b_4(\alpha^t), & 0 \leq j \leq 2^n - 1 \\ tr_1^n(\alpha^t), & j = 2^n \end{cases},$$

where $\{\lambda_0, \lambda_1, \cdots, \lambda_{2n-1}\}$ *is an enumeration of the elements in* $GF(2^n)$*. Let* $\delta = (1+\delta_1+\delta_2)^{-1}, \omega = tr_1^n(\delta), \beta = tr_1^n(\delta_1 \cdot \delta), \gamma = tr_1^n(\delta_2 \cdot \delta), \theta = tr_1^n(\delta_1 \delta_2 \cdot \delta^2)$*, then* $\omega + \beta + \gamma = 0$*. If* $(\omega,\beta,\gamma,\theta) = (0,0,0,0)$*, then* S_4 *has correlations* $2^n-1, -1, -1 \pm 2^{n/2}, -1 \pm 2^{n/2+1}, -1 \pm 2^{n/2+2}, -1 \pm 2^{n/2+3}$*. If* $(\omega,\beta,\gamma,\theta) \in \{(0,0,0,1),(0,1,1,0),(0,1,1,1),(1,0,1,0),(1,0,1,1),(1,1,0,0),(1,1,0,1)\}$*, then* S_4 *has correlations* $2^n-1, -1, -1 \pm 2^{n/2}, -1 \pm 2^{n/2+1}, -1 \pm 2^{n/2+2}$*. For example, if* $(\omega,\beta,\gamma,\theta) = (0,0,0,1)$*, then* S_4 *has the distribution of correlation values as follows:*

$$C_{i,j}(\tau) = \begin{cases} 2^n - 1, & 2^n + 1 times \\ -1, & 2^{3n-1} - 2^{3n-4} - 2^{3n-7} - 2^{3n-9} - 2^{3n-10} - 2^{2n-1} - 2 times \\ -1 + 2^{n/2}, & (2^{n-1} + 2^{n/2-1})(2^{2n-2} + 2^{2n-3} + 2^{2n-4} + 2^{n+1} - 2) times \\ -1 - 2^{n/2}, & (2^{n-1} - 2^{n/2-1})(2^{2n-2} + 2^{2n-3} + 2^{2n-4} + 2^{n+1} - 2) times \\ -1 + 2^{n/2+1}, & (2^{n-3} + 2^{n/2-2})2^n(2^{n-1} + 2^{n-5} + 2^{n-6} - 2) times \\ -1 - 2^{n/2+1}, & (2^{n-3} - 2^{n/2-2})2^n(2^{n-1} + 2^{n-5} + 2^{n-6} - 2) times \\ -1 + 2^{n/2+2}, & (2^{n-5} + 2^{n/2-3})2^{2n-6} times \\ -1 - 2^{n/2+2}, & (2^{n-5} - 2^{n/2-3})2^{2n-6} times \end{cases}.$$

Remark 3. If $(\omega,\beta,\gamma,\theta)$ equals to other values, we can similarly determine the distribution of correlation values of S_3 and S_4.

4 Conclusions

In this paper, we propose four new binary sequence families S_1, S_2, S_3 andS_4 with low correlation, the parameter C_{max} in S_1, S_2, S_3 andS_4 is of the order $\sqrt{N}$(where N is the period of the sequences). S_1 has six-valued correlations, while S_2 and S_3 have either six-valued correlations or eight-valued correlations, andS_4 has either eight-valued or ten-valued, depending on the choice of parameters. Compared with Gold sequence family, the linear span of the sequences in S_1, S_2, S_3 and S_4 is much higher (at least $n \cdot (n-1)/2$ except for one m-sequence). Finally we conclude the paper by a comparison of the known binary sequence families.

Table 1. Comparison of the known binary sequence families

Family	Period	Size of Family	C_{max}	Linear Complexity
Gold	$2^n - 1$(odd n)	$2^n + 1$	$1 + 2^{(n+1)/2}$	$2n$
Gold	$2^n - 1$(even n)	$2^n + 1$	$1 + 2^{(n+2)/2}$	$2n$
Gold-like	$2^n - 1$(odd n)	$2^n + 1$	$1 + 2^{(n+1)/2}$	$n(n-1)/2$
Gold-like	$2^n - 1$(even n)	$2^n + 1$	$1 + 2^{(n+2)/2}$	$n(n-1)/2$
S	$2^n - 1$(odd n)	$2^n + 1$	$1 + 2^{(n+e)/2}$	$n(n-e)/2e$
U	$2^n - 1$(even n)	$2^n + 1$	$1 + 2^{(n+2e)/2}$	$n(n-e)/2e$
S_1	$2^n - 1$(odd n)	$2^n + 1$	$1 + 2^{(n+3)/2}$	$n(n-1)/2$
S_2	$2^n - 1$(even n)	$2^n + 1$	$1 + 2^{(n+2)/2}$ or $1 + 2^{(n+4)/2}$	$n(n-1)/2$
S_3	$2^n - 1$(odd n)	$2^n + 1$	$1 + 2^{(n+3)/2}$ or $1 + 2^{(n+5)/2}$	$n(n-1)/2$
S_4	$2^n - 1$(even n)	$2^n + 1$	$1 + 2^{(n+4)/2}$ or $1 + 2^{(n+6)/2}$	$n(n-1)/2$

Acknowledgments. The authors would like to express their deep gratitude to the anonymous referees for their valuable comments on this paper.

References

1. Ziemer, R., Peterson, R.: Digital communication and spread spectrum communication systems. McMilian, New York (1985)
2. Gong, G.: New designs for signal sets with low cross correlation, balance property, and large linear span: $GF(p)$ case. IEEE Trans. Inform. Theory 48, 2847–2867 (2002)
3. Sidelnikov, V.M.: On mutual correlation of sequences. Sov. Math. Doklady 12, 197–201 (1971)
4. Gold, R.: Maximal recursive sequences with 3-valued cross correlation function. IEEE Trans. Inform. Theory 14, 154–156 (1968)
5. Boztas, S., Kumar, P.V.: Binary sequences with Gold-like correlation but larger linear span. IEEE Trans. Inform. Theory 40, 532–537 (1994)
6. Udaya, P.: Polyphase and frequency hopping sequences obtained from finite rings, Ph. D. dissertation, Dept. Elec. Eng. Indian Inst. Technol., Kanpur (1992)
7. Kim, S.H., No, J.S.: New families of binary sequences with low correlation. IEEE Trans. Inform. Theory 49, 3059–3065 (2003)
8. Lidl, R., Niederreiter, H.: Finite fields, Encycl. Math. Appl., vol. 20. Addision Wesley, Reading (1983)
9. Golomb, S.W.: Shift register sequences, San Francisco, CA, Holden-Day (1967); revised edition: Aegean Park Press, Laguna Hills, CA (1982)
10. Wang, J.S., Qi, W.F.: Trace presentation of Bent sequence families. Journal of Communications 27, 8–13 (2006)
11. Wang, J.S., Qi, W.F.: Analysis of design interleaved ZCZ sequence family. In: Gong, G., Helleseth, T., Song, H.-Y., Yang, K. (eds.) SETA 2006. LNCS, vol. 4086, pp. 129–140. Springer, Heidelberg (2006)
12. Wang, J.S., Qi, W.F.: A class of binary ZCZ sequence families constructed by extending period twice. Journal of Electronics(China) 24, 301–304 (2007)

Pseudo-Randomness of Discrete-Log Sequences from Elliptic Curves*

Zhixiong Chen[1,2], Ning Zhang[3], and Guozhen Xiao[3]

[1] Department of Mathematics, Putian Univ., Putian, Fujian 351100, China
[2] Key Lab of Network Security and Cryptology, Fujian Normal University, Fuzhou, Fujian 350007, China
ptczx@126.com
[3] Ministry of Education Key Lab of Computer Networks and Information Security, Xidian University, Xi'an, 710071, China
znlady@163.com

Abstract. We investigate an upper bound on the discrepancy and a lower bound on the linear complexity of a class of sequences, derived from elliptic curves by using discrete logarithm in this paper. The results indicate that these sequences may have 'nice' pseudo-random properties. The important tool in the proof is certain character sums estimate along elliptic curves. In addition, we apply linear recurrence relation over elliptic curves to output binary sequences with very interesting pseudo-random behavior.

Keywords: pseudo-random sequences, elliptic curves, discrete logarithm, character sums, discrepancy, linear complexity.

1 Introduction

There is a vast literature devoted to generating pseudo-random sequences using arithmetic of finite fields and residue rings. Recent developments point towards an interest in the elliptic curve analogues of pseudo-random number generators, which are reasonably new sources of pseudo-random numbers. Such generators include the elliptic curve linear congruential generators [1,8,11,12,18], the elliptic curve power generators [14] and the elliptic curve Naor-Reingold generators [22,25]. A survey about these elliptic curve generators, including some unsolved questions, is given in [24]. An alternative method, based on different ideas, is proposed to generate sequences by using endomorphisms of elliptic curves in [15]. For other number generators related to elliptic curves, the reader is referred to [4,9,13,16].

* This work was supported in part by the National Natural Science Foundation of China (No. 60473028). The research of the first author was also supported in part by the Science and Technology Foundation of Putian City (No. 2005S04), the open funds of Key Lab of Fujian Province University Network Security and Cryptology (No.07B005) and the Natural Science Foundation of Fujian Province of China (No.2007F3086).

As noted in [24], there are at least two main reasons motivating the constructions related to elliptic curves:

(1). Many standard pseudo-random number generators based on finite fields and residue rings have proved to be insecure or at least requiring great care in their use.
(2). Many cryptographic protocols explicitly require to generate random points on a given elliptic curve (in elliptic curve cryptosystems).

As the study shows that, the elliptic curve analogues of pseudo-random number generators possess strongly cryptographic properties and hence provide potential applications for generating pseudo-random numbers and session keys in encryption phases in elliptic curve cryptosystems.

In particular, the notion of index (discrete logarithm) is used to construct sequences recently. Sárközy constructed finite binary sequences by using the notion of index over finite fields in [21], Gyarmati [10] extended this construction to construct a large family of binary sequences. It was shown that these sequences have very interesting pseudo-random behavior. Following the path of [21] and [10], we constructed a family of binary sequences from elliptic curves in [3]. In [1], Beelen and Doumen constructed a large family of sequences from elliptic curves using additive characters and multiplicative characters and investigated the pseudo-random properties of balance, autocorrelation and cross-correlation of the resulting sequences.

In the present paper, we continue the investigation of the sequences derived from elliptic curves by using the notion of index. The properties of the (multi-dimensional) distribution and the linear complexity are considered. This article is organized as follows. In Section 2, some basic facts on elliptic curves and sequences over rings are introduced. Some cryptographic properties of sequences and linear recursive sequences from elliptic curves are considered in Sections 3 and 4 respectively. A conclusion is drawn in Section 5.

2 Preliminaries

Let p be a prime. We denote by $\mathbb{F}_p$ the finite field of p elements, by $\mathbb{F}_p^*$ the multiplicative group of $\mathbb{F}_p$ and by $\overline{\mathbb{F}}_p$ the algebraic closure of $\mathbb{F}_p$.

Let g be a fixed primitive element of $\mathbb{F}_p^*$, i.e., the multiplicative order of g is $p-1$. Then for each $x \in \mathbb{F}_p^*$, let $\operatorname{ind}(x)$ denote the index (discrete logarithm) of x (to the base g) so that

$$x = g^{\operatorname{ind}(x)}.$$

We add the condition

$$0 \le \operatorname{ind}(x) \le p-2$$

to make the value of index unique. A (multiplicative) character of $\mathbb{F}_p^*$ [23,17] is defined by

$$\chi_a(x) := e_{p-1}(a \cdot \operatorname{ind}(x)) = \exp\left(\frac{2\pi i a \cdot \operatorname{ind}(x)}{p-1}\right)$$

where $a \in \mathbb{Z}_{p-1}$. For convenience, we extend χ to $\mathbb{F}_p$ by adding $\chi(0) = 0$.

For any positive integer n, we identify $\mathbb{Z}_n$, the residue ring modulo n, with the set $\{0, 1, \cdots, n-1\}$. Put

$$e_n(z) = \exp(2\pi i z/n).$$

2.1 Elliptic Curves and Exponential Sums

Let $\mathcal{E}$ be an elliptic curve over $\mathbb{F}_p$, given by an affine Weierstrass equation of the form

$$y^2 + (a_1 x + a_3)y = x^3 + a_2 x^2 + a_4 x + a_6,$$

with coefficients $a_i \in \mathbb{F}_p$ and nonzero discriminant, see [7] for details. It is known that the set $\mathcal{E}(\mathbb{F}_p)$ of $\mathbb{F}_p$-rational points of $\mathcal{E}$ forms an Abelian group under an appropriate composition rule denoted by $\oplus$ and with the point at infinity $\mathcal{O}$ as the neutral element. We recall that

$$|\#\mathcal{E}(\mathbb{F}_p) - p - 1| \le 2p^{1/2}, \tag{1}$$

where $\#\mathcal{E}(\mathbb{F}_p)$ is the number of $\mathbb{F}_p$-rational points, including the point at infinity $\mathcal{O}$.

Let $\mathbb{F}_p(\mathcal{E})$ be the function field of $\mathcal{E}$ defined over $\mathbb{F}_p$. For any $f \in \mathbb{F}_p(\mathcal{E})$ and $R \in \mathcal{E}(\overline{\mathbb{F}}_p)$, R is called a *zero* (resp. a *pole*) of f if $f(R) = 0$ (resp. $f(R) = \infty$). Any rational function has only a finite number of zeros and poles. The *divisor* of a rational function f is written as

$$\mathrm{Div}(f) = \sum_{R \in \mathcal{E}(\overline{\mathbb{F}}_p)} \mathrm{ord}_R(f)[R],$$

where each integer $\mathrm{ord}_R(f)$ is the order of f at R and $\mathrm{ord}_R(f) = 0$ for all but finitely many $R \in \mathcal{E}(\overline{\mathbb{F}}_p)$. Note that $\mathrm{ord}_R(f) > 0$ if R is a zero of f and $\mathrm{ord}_R(f) < 0$ if R is a pole of f.
We also write

$$\mathrm{Supp}(f) = \{R \in \mathcal{E}(\overline{\mathbb{F}}_p) | f(R) = 0 \text{ or } f(R) = \infty\},$$

which is called the *support* of $\mathrm{Div}(f)$. In particular, $\#\mathrm{Supp}(f)$, the cardinality of $\mathrm{Supp}(f)$, is 2 or 3 if $f = x$ and $\#\mathrm{Supp}(f) \le 4$ if $f = y$.

The translation map by $W \in \mathcal{E}(\mathbb{F}_p)$ on $\mathcal{E}(\mathbb{F}_p)$ is defined as

$$\begin{aligned} \tau_W : \mathcal{E}(\mathbb{F}_p) &\to \ \mathcal{E}(\mathbb{F}_p) \\ P \quad &\mapsto P \oplus W. \end{aligned}$$

It is obvious that $(f \circ \tau_W)(P) = f(\tau_W(P)) = f(P \oplus W)$. We denote by $\ominus$ the inverse operation of $\oplus$ in the rational points group of $\mathcal{E}$. From [7, Lemmas 3.14 and 3.16 and Theorem 3.17], we have the following statement.

Lemma 1. *Let $f \in \mathbb{F}_p(\mathcal{E})$ be a nonconstant rational function. If $R \in \mathrm{Supp}(f)$ and the order of f at R is ρ, then $R \ominus W$ belongs to the support of $\mathrm{Div}(f \circ \tau_W)$ with the same order ρ.*

The character sums along elliptic curves with respect to multiplicative characters is defined as follows:

$$S(\chi, f) = \sum\nolimits^{*}_{R \in \mathcal{E}(\mathbb{F}_p)} \chi(f(R)),$$

where χ is a multiplicative character of $\mathbb{F}_p^*$, $f \in \mathbb{F}_p(\mathcal{E})$ is a rational function and $\sum^*$ indicates that the poles of f are excluded from the summation. The upper bound of $S(\chi, f)$ has been estimated in [1,19,20]. Below we present a special case of the character sums (see [20, Proposition 4.5 and Corollary 4.6] or [1, Proposition 3]), which is necessary in the context.

Lemma 2. *Let $f(x,y) \in \mathbb{F}_p(\mathcal{E})$ be a nonconstant rational function with $f(x,y) \neq z^l(x,y)$ for all $z(x,y) \in \overline{\mathbb{F}}_p(\mathcal{E})$ and all factors $l > 1$ of $p-1$. For any non-principal multiplicative character χ, the following upper bound holds:*

$$|S(\chi, f)| < \#\mathrm{Supp}(f)\sqrt{p},$$

where $\#\mathrm{Supp}(f)$ is the cardinality of the support of $\mathrm{Div}(f)$*.*

2.2 Sequences over Rings

There are many important criteria which a 'good' pseudo-random sequence should satisfy. In the present paper, we concentrate only on its *discrepancy* and *linear complexity.*

For an r-dimensional sequence of N points

$$(\gamma_{1n}, \cdots, \gamma_{rn}), \quad n = 1, \cdots, N \tag{2}$$

of the half-open interval $[0,1)^r$, the *discrepancy* of this sequence, denoted by $\mathcal{D}(N)$, is defined by

$$\mathcal{D}(N) = \sup_{J \subseteq [0,1)^r} \left| \frac{A(J,N)}{N} - |J| \right|,$$

where $A(J,N)$ is the number of points of the sequence above which hit the box $J = [\alpha_1, \beta_1) \times \cdots \times [\alpha_r, \beta_r) \subseteq [0,1)^r$, the volume $|J|$ of an interval J is given by $\prod_{i=1}^{r}(\beta_i - \alpha_i)$ and the supremum is taken over all such boxes, see [6].

It is easy to see that the discrepancy is a quantitative measure of uniformity of distribution of sequences, and thus 'good' pseudo-random sequences should have a small discrepancy.

For an integer vector $a = (a_1, \cdots, a_r) \in \mathbb{Z}^r$, we put

$$|a| = \max_{i=1,\cdots,r} |a_i|, \quad r(a) = \prod_{i=1}^{r} \max\{|a_i|, 1\}.$$

The *Erdös-Turán-Koksma inequality* (see [6, Theorem 1.21]), relating the discrepancy and character sums, is presented in the following form.

Lemma 3. *With notations as above. $\mathcal{D}(N)$ is the discrepancy of the r-dimensional sequence of N points (2). Then there exists a constant $C_r > 0$ depending only on the dimension r such that, for any integer $L \geq 1$, the bound*

$$\mathcal{D}(N) < C_r \left(\frac{1}{L} + \frac{1}{N} \sum_{0<|a|\leq L} \frac{1}{r(a)} \left| \sum_{n=1}^{N} \exp\left(2\pi i \sum_{j=1}^{r} a_j \gamma_{jn} \right) \right| \right)$$

holds, where $|a|, r(a)$ are defined as above and the summation is taken over all integer vectors $a = (a_1, \cdots, a_r) \in \mathbb{Z}^r$ with $0 < |a| \leq L$.

We recall that for any $x > 1$,

$$\sum_{n\leq x} \frac{1}{n} \leq 1 + \log(x).$$

From Lemma 3, one can estimate the discrepancy $\mathcal{D}(N)$ by bounding the corresponding character sums

$$\sum_{n=1}^{N} \exp\left(2\pi i \sum_{j=1}^{r} a_j \gamma_{jn} \right).$$

3 Generating Sequences over Elliptic Curves

Let $\mathcal{E}(\mathbb{F}_p)$ be a cyclic group of order T and G its generator. Let $f \in \mathbb{F}_p(\mathcal{E})$ be a rational function. The frequent case is to select $f = x$ or $f = y$ or $f = x + y$, and so on. We should note that, in the sequel, all rational functions are not of the form $z^l(x, y)$ for all $z(x, y) \in \overline{\mathbb{F}}_p(\mathcal{E})$ and all factors $l > 1$ of $p - 1$. For any $i : 1 \leq i \leq T$, if $f(iG) \neq 0$ and ∞, then there exists a unique $u_i \in [0, p - 2]$ such that

$$f(iG) = g^{u_i},$$

where g is a fixed primitive element of $\mathbb{F}_p^*$. If iG is a zero or a pole of f, we set $u_i = 0$ as its output. Hence we obtain a sequence $\mathcal{U} = (u_i)_{i>0}$ of elements of $\{0, 1, \cdots, p - 2\}$. Obviously this sequence is purely periodic with period T. In fact, for $1 \leq i \leq T$,

$$u_i = \begin{cases} 0, & \text{if } f(iG) = 0 \text{ or } \infty; \\ \operatorname{ind}(f(iG)), & \text{otherwise.} \end{cases} \tag{3}$$

The distribution in parts of period of $\mathcal{U} = (u_i)_{i>0}$ will be considered in Subsection 3.1. We note that the period T of $\mathcal{U}$ is bounded by the size of $\mathcal{E}(\mathbb{F}_p)$ and hence by $O(p)$ from (1). Since $\mathcal{E}(\mathbb{F}_p)$ is a cyclic group and G is its generator, $T \sim p$. We also remark that the resulting sequences $\mathcal{U}$ are very common enough. From [26,27], about 75% of the majority of (isomorphism classes of) elliptic curves have a cyclic point group.

One can consider the following more general case. Let

$$f_1, f_2, \cdots, f_r$$

be rational functions in $\mathbb{F}_p(\mathcal{E})$. Set

$$\mathcal{V}_i = (v_{i1}, v_{i2}, \cdots, v_{iT}), \ \ i = 1, 2, \cdots, r,$$

where $v_{ij} = 0$ if jG is a zero or a pole of f_i, and $v_{ij} = \mathrm{ind}(f_i(jG))$ for all $i \in [1, r]$ and $j \in [1, T]$ otherwise. We obtain an interleaved sequence

$$\mathcal{V} = \begin{pmatrix} \mathcal{V}_1 \\ \mathcal{V}_2 \\ \cdots \\ \mathcal{V}_r \end{pmatrix} = \begin{pmatrix} v_{11}, v_{12}, \cdots, v_{1T} \\ v_{21}, v_{22}, \cdots, v_{2T} \\ \cdots \ \cdots \ \cdots \ \cdots \\ v_{r1}, v_{r2}, \cdots, v_{rT} \end{pmatrix}. \tag{4}$$

The elements of $\mathcal{V}$ should be read column by column and indeed $\mathcal{V}_i$ is an $r-$decimation of $\mathcal{V}$ for $i = 1, 2, \cdots, r$. In Subsections 3.2 and 3.3 we will consider the multi-dimensional distribution and the linear complexity of $\mathcal{V}$ respectively.

3.1 Distribution over Parts of the Period

Define

$$\begin{aligned} S_{\mathcal{E}}(N) &= \sum_{n=1}^{N} e_{p-1}(au_n) = \sum_{n=1}^{N} e_{p-1}(a \cdot \mathrm{ind}(f(nG))) \\ &= \sum_{n=1}^{N} \chi_a(f(nG)) + O(1), \end{aligned}$$

where $a \in \mathbb{Z}_{p-1}$ and χ_a is a multiplicative character of $\mathbb{F}_p^*$, in the last sums the poles and the zeros of f are excluded.

Theorem 1. *With notations as above. Let $u_1, u_2, \cdots,$ be defined in (3). The following bound*

$$\max_{0 \neq a \in \mathbb{Z}_{p-1}} |S_{\mathcal{E}}(N)| \ll N^{1/2} p^{1/4}$$

holds for $1 \le N \le T$. The implied constant depends only on $\#\mathrm{Supp}(f)$.

Proof. For any $j \ge 0$, we have

$$\left| S_{\mathcal{E}}(N) - \sum_{n=1}^{N} e_{p-1}(au_{n+j}) \right| \le 2j.$$

Let $K > 0$. From

$$\left| \sum_{j=0}^{K-1} \left(S_{\mathcal{E}}(N) - \sum_{n=1}^{N} e_{p-1}(au_{n+j}) \right) \right| \ge K \left| S_{\mathcal{E}}(N) \right| - \left| \sum_{j=0}^{K-1} \sum_{n=1}^{N} e_{p-1}(au_{n+j}) \right|$$

and

$$\left|\sum_{j=0}^{K-1}\left(S_{\mathcal{E}}(N)-\sum_{n=1}^{N}e_{p-1}(au_{n+j})\right)\right| \le \sum_{j=0}^{K-1}2j < K^2,$$

we derive

$$K\left|S_{\mathcal{E}}(N)\right| \le U + K^2,$$

where $U = \left|\sum_{j=0}^{K-1}\sum_{n=1}^{N} e_{p-1}(au_{n+j})\right| \le \sum_{n=1}^{N}\left|\sum_{j=0}^{K-1} e_{p-1}(au_{n+j})\right|$.

We apply the Cauchy-Schwarz inequality and extend the summation to all value $n \in \{1, 2, \cdots, T\}$, getting

$$\begin{aligned}
U^2 &\le N\sum_{n=1}^{N}\left|\sum_{j=0}^{K-1} e_{p-1}(au_{n+j})\right|^2 \\
&\le N\sum_{n=1}^{T}\left|\sum_{j=0}^{K-1} e_{p-1}(au_{n+j})\right|^2 \\
&= N\sum_{n=1}^{T}\sum_{i,j=0}^{K-1} e_{p-1}(au_{n+i})\cdot e_{p-1}(-au_{n+j}) \\
&= N\sum_{i,j=0}^{K-1}\sum_{n=1}^{T}\chi_a\left(f((n+i)G))\cdot\chi_a(f((n+j)G)^{-1}\right)+O(1) \\
&= N\sum_{i,j=0}^{K-1}\sum_{n=1}^{T}\chi_a((f\circ\tau_{iG}\cdot(f\circ\tau_{jG})^{-1})(nG))+O(1) \\
&= N\sum_{i,j=0}^{K-1}\sum_{Q\in\langle G\rangle}\chi_a((f\circ\tau_{iG}\cdot(f\circ\tau_{jG})^{-1})(Q))+O(1).
\end{aligned}$$

(We note that in the sums above the poles and the zeros of f are excluded.)

If $i = j$, then the inner sum is trivially equal to T and there are K such sums. In case this does not happen, the inner sum above is a character sum with a rational function $F := f\circ\tau_{iG}\cdot(f\circ\tau_{jG})^{-1}$. #Supp($F$), the cardinality of the support of Div(F), is at most 2#Supp(f) by Lemma 1. By Lemma 2 each of these terms contributes at most $O(p^{1/2})$. Therefore

$$U^2 \ll NKT + NK(K-1)p^{1/2},$$

and so

$$\left|S_{\mathcal{E}}(N)\right| \ll N^{1/2}(T^{1/2}K^{-1/2}+p^{1/4})+K.$$

If $K \ge T$, the bound is trivial. So we assume $K < T$ and put $K = \lceil Tp^{-1/2}\rceil$. We have

$$\left|S_{\mathcal{E}}(N)\right| \ll N^{1/2}p^{1/4}+Tp^{-1/2}.$$

We suppose that $N \ge p^{1/2}$, otherwise the bound is trivial. From (1), the first term always dominates the second term. we derive the desired result. □

The following result follows directly from Theorem 1 and Lemma 3.

Theorem 2. *The discrepancy $\mathcal{D}_{\mathcal{E}}(N)$ of the first N terms $u_1, \cdots, u_N$ defined in (3) satisfies*

$$\mathcal{D}_{\mathcal{E}}(N) \ll N^{-1/2} p^{1/4} \log(p)$$

for $1 \leq N \leq T$. The implied constant depends on $\#\mathrm{Supp}(f)$.

3.2 Multi-dimensional Distribution over the Full Period

In this subsection, we will consider the uniformity of distribution of $rs-$tuples

$$\begin{array}{ccc} (\mathrm{ind}(f_1(nG)), & \cdots, & \mathrm{ind}(f_r(nG)), \\ \mathrm{ind}(f_1((n+1)G)), & \cdots, & \mathrm{ind}(f_r((n+1)G)), \\ \cdots, & \cdots, & \cdots, \\ \mathrm{ind}(f_1((n+s-1)G)), & \cdots, & \mathrm{ind}(f_r((n+s-1)G)), \end{array} \tag{5}$$

$1 \leq n \leq T$, in an $rs-$dimensional cube $\{0, 1, \cdots, p-2\}^{rs}$ over the full period (read row by row). Using the similar idea of Theorem 1, one can obtain the discrepancy of (5) for $1 \leq n \leq N$, where $N \leq T$. Below for convenience, we only consider $N = T$ case.

Let $\mathcal{A}$ be a nonzero $r \times s$ matrix

$$\mathcal{A} = (a_{ij})_{r \times s}$$

over $\mathbb{Z}_{p-1}$. Define

$$h(nG) = \sum_{i=1}^{r} \sum_{j=1}^{s} a_{ij} \mathrm{ind}(f_i((n+j-1)G)),$$

we estimate the exponential sums

$$S_{\mathcal{E}}(r, s) = \sum_{n=1}^{T} e_{p-1}(h(nG)) \tag{6}$$

and the discrepancy $\mathcal{D}_{\mathcal{E}}(r, s)$ of the sequence (5) by Lemma 3.

Theorem 3. *Let $\underline{0}$ be the $r \times s$ zero matrix. The bound*

$$\max_{\mathcal{A} \neq \underline{0}} |S_{\mathcal{E}}(r, s)| \leq s p^{1/2} \sum_{i=1}^{r} \#\mathrm{Supp}(f_i)$$

holds and the discrepancy $\mathcal{D}_{\mathcal{E}}(r, s)$ of the $rs-$dimensional sequence (5) satisfies

$$\mathcal{D}_{\mathcal{E}}(r, s) \ll T^{-1} p^{1/2} \log^{rs}(p),$$

where the implied constant depends on s and $\sum_{i=1}^{r} \#\mathrm{Supp}(f_i)$.

Proof.

$$\begin{aligned}S_{\mathcal{E}}(r,s) &= \sum_{n=1}^{T} e_{p-1}(h(nG)) \\ &= \sum_{n=1}^{T} e_{p-1}\left(\sum_{i=1}^{r}\sum_{j=1}^{s} a_{ij}\cdot \mathrm{ind}(f_i((n+j-1)G))\right) \\ &= \sum_{n=1}^{T}\prod_{i=1}^{r}\prod_{j=1}^{s} e_{p-1}\left(a_{ij}\cdot \mathrm{ind}(f_i((n+j-1)G))\right)\end{aligned}$$

So we have

$$\begin{aligned}|S_{\mathcal{E}}(r,s)| &= \left|\sum_{n=1}^{T}\prod_{i=1}^{r}\prod_{j=1}^{s} e_{p-1}\left(a_{ij}\cdot \mathrm{ind}(f_i((n+j-1)G))\right)\right| \\ &= \left|\sum_{n=1}^{T}\prod_{i=1}^{r}\prod_{j=1}^{s} \chi_{a_{ij}}(f_i((n+j-1)G))\right| + O(1) \\ &= \left|\sum_{n=1}^{T}\prod_{i=1}^{r}\prod_{j=1}^{s} \chi_{a_{ij}}(f_i\circ\tau_{(j-1)G}(nG))\right| + O(1)\end{aligned}$$

Let ψ be a generator of the group of all multiplicative characters of $\mathbb{F}_p^*$. Then for each $\chi_{a_{ij}}$, there exists an integer $b_{ij}\in[1,p-1]$ such that $\chi_{a_{ij}}=\psi^{b_{ij}}$. We derive

$$\begin{aligned}|S_{\mathcal{E}}(r,s)| &= \left|\sum_{n=1}^{T}\prod_{i=1}^{r}\prod_{j=1}^{s} \psi\left((f_i\circ\tau_{(j-1)G}(nG))^{b_{ij}}\right)\right| + O(1) \\ &= \left|\sum_{n=1}^{T}\psi\left(\prod_{i=1}^{r}\prod_{j=1}^{s}(f_i\circ\tau_{(j-1)G})^{b_{ij}}(nG)\right)\right| + O(1)\end{aligned}$$

By Lemma 1 we have $\#\mathrm{Supp}(f_i\circ\tau_{(j-1)G}) = \#\mathrm{Supp}(f_i)$ for $i\in[1,r]$ and $j\in[1,T]$, hence

$$\#\mathrm{Supp}(\prod_{i=1}^{r}\prod_{j=1}^{s}(f_i\circ\tau_{(j-1)G})^{b_{ij}}) \le s\sum_{i=1}^{r}\#\mathrm{Supp}(f_i).$$

We derive the desired results by Lemmas 2 and 3. □

3.3 Lower Bound on Linear Complexity

Linear complexity is an interesting characteristics of a sequence for applications in cryptography. A low linear complexity of a sequence has turned out to be undesirable for applications. Let

$$(\gamma_{1n},\cdots,\gamma_{rn}),\ \ n=1,\cdots,N$$

be an $r-$dimensional sequence with N terms over a ring $\mathcal{R}$. The *linear complexity* $L(N)$ (of an $r-$dimensional sequence) is defined as the smallest s for which the following relations hold

$$\sum_{i=1}^{r}\sum_{j=1}^{s} a_{ij}\gamma_{i(n+j)} = 0, \quad 0 \le n \le N-s$$

with some nonzero fixed matrix $\mathcal{A} = (a_{ij})$ of order $r \times s$ over $\mathcal{R}$, see [12,5].

Let v_{ij} be defined as (4), that is

$$v_{ij} = \begin{cases} 0, & \text{if } \ f_i(jG) = 0 \ \text{ or } \ \infty; \\ \mathrm{ind}(f_i(jG)), & \text{otherwise}; \end{cases}$$

for any $j > 0$ and $i \in [1, r]$.

Theorem 4. *The linear complexity L of $r-$dimensional sequence*

$$(v_{1n}, v_{2n}, \cdots, v_{rn}), \ \ n = 1, 2, \cdots$$

satisfies

$$L \ge \frac{T}{(p^{1/2}+1)\sum\limits_{i=1}^{r} \#\mathrm{Supp}(f_i)},$$

where T is the order of G.

Proof. Assume $L = s(\le T)$ and

$$\sum_{j=1}^{s}\sum_{i=1}^{r} a_{ij} v_{i(n+j)} \equiv 0 \pmod{p-1}, \quad n \ge 0$$

with a nonzero matrix $\mathcal{A} = (a_{ij})_{r\times s}$ over $\mathbb{Z}_{p-1}$. Then there are at least

$$T - s\sum_{i=1}^{r} \#\mathrm{Supp}(f_i)$$

distinct points nG in $\langle G\rangle$ such that $f_i((n+j)G) \ne 0$ and ∞ for all $1 \le i \le r$ and $1 \le j \le s$. Hence for such points nG, we have

$$\sum_{j=1}^{s}\sum_{i=1}^{r} a_{ij} \cdot \mathrm{ind}(f_i((n+j)G)) \equiv 0 \pmod{p-1}.$$

Furthermore,

$$1 = e_{p-1}(0) = e_{p-1}\left(\sum_{i=1}^{r}\sum_{j=1}^{s} a_{ij} \cdot \mathrm{ind}(f_i((n+j)G)\right).$$

So

$$T - s\sum_{i=1}^{r} \#\mathrm{Supp}(f_i) \le \left| \sum_{n=1}^{T} e_{p-1}\left(\sum_{i=1}^{r}\sum_{j=1}^{s} a_{ij}\cdot \mathrm{ind}(f_i((n+j-1)G) \right) \right|$$

$$= \left| \sum_{n=1}^{T} \prod_{i=1}^{r} \prod_{j=1}^{s} e_{p-1}\left(a_{ij}\cdot \mathrm{ind}(f_i((n+j-1)G))\right) \right|.$$

Now by the proof of Theorem 3, we derive

$$T \le s(p^{1/2}+1)\sum_{i=1}^{r} \#\mathrm{Supp}(f_i),$$

which completes the proof of Theorem 4. □

Since $\mathcal{E}(\mathbb{F}_p)$ is a cyclic group and $T \sim p$, the bound in Theorem 4 is of the order of magnitude $O(p^{1/2})$.

Corollary 1. *The lower bound on the linear complexity of the sequence $\mathcal{U} = (u_i)_{i>0}$ defined in (3) is*

$$\frac{T}{(p^{1/2}+1)\#\mathrm{Supp}(f)}.$$

4 Generating Linear Recursive Sequences over Elliptic Curves

As we have mentioned above, the period of the sequence $\mathcal{U} = (u_i)_{i>0}$ in (3) is bounded by $O(p)$, the size of the finite field $\mathbb{F}_p$ over which the elliptic curve is defined. In [1,9], a method for generating sequences was proposed by applying linear recurrence relations on elliptic curves, which may produce rational point sequences with long periods.

Let $\mathcal{E}(\mathbb{F}_p)$ be of order T, where T is a prime (only in this section). A linear recursive sequence $\mathcal{P} = (P_i)_{i\ge 0}$ over elliptic curves satisfies the recurrence relation

$$P_{n+i} = c_{n-1}P_{n-1+i} \oplus \cdots \oplus c_1P_{1+i} \oplus c_0P_i, \quad i \ge 0 \tag{7}$$

with initial values $P_0 = s_0G, P_1 = s_1G, \cdots, P_{n-1} = s_{n-1}G$ in $\mathcal{E}(\mathbb{F}_p)$, where $s_i \in \mathbb{F}_T$ for $i = 0, 1, \cdots, n-1$. Let

$$m(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_1x - c_0 \in \mathbb{F}_T[x].$$

We call that $m(x)$ is a characteristic polynomial of $\mathcal{P}$. Let $P_i = s_iG, i \ge 0$. From (7), it is easy to see the sequence (s_i) satisfies

$$s_{n+i} = c_{n-1}s_{n-1+i} + \cdots + c_1s_{1+i} + c_0s_i, \quad i \ge 0$$

with initial values $s_0, s_1, \cdots, s_{n-1} \in \mathbb{F}_T$, i.e., $m(x)$ is also a characteristic polynomial of (s_n).

From now on we always suppose that $m(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_1x - c_0$ is a primitive polynomial in $\mathbb{F}_T[x]$. In this case, the sequence (s_i) (hence $\mathcal{P}$) is periodic with least period $T^n - 1$. Let $d = (T^n - 1)/(T - 1)$. We denote by

$$\mathcal{P}_j = (P_{j+dk})_{k\geq 0}$$

the $d-$decimation of $\mathcal{P}$, where $j = 0, 1, \cdots, d-1$. Clearly $T-1$ is a period (needn't be the least) of each $\mathcal{P}_j$. We arrange the first period of $\mathcal{P}$ as an interleaved sequence:

$$\mathcal{P} = \begin{pmatrix} \mathcal{P}_0 \\ \mathcal{P}_1 \\ \cdots \\ \mathcal{P}_{d-1} \end{pmatrix} = \begin{pmatrix} s_0G, & s_dG, & \cdots, & s_{(T-2)d}G \\ s_1G, & s_{1+d}G, & \cdots, & s_{1+(T-2)d}G \\ \cdots, & \cdots, & \cdots, & \cdots \\ s_{d-1}G, & s_{2d-1}G, & \cdots, & s_{(T-1)d-1}G \end{pmatrix}_{d\times(T-1)} \quad (8)$$

From [2], we have the following two facts:

Fact I. For the matrix in (8), there are exactly $d - T^{n-1}$ rows with each entries $\mathcal{O}$, the point at infinity of $\mathcal{E}$.

Fact II. The entries in each row of the rest T^{n-1} rows exactly run through all points $G, 2G, \cdots, (T-1)G$. And these rows are shift equivalent.

Applying the method employed to construct $\mathcal{U}$ in (3), we compute the discrete logarithm of the rational function value of each point of $\mathcal{P}$ and obtain the sequence $\mathcal{W} = (w_i)$:

$$w_i = \begin{cases} 0, & \text{if } f(s_iG) = 0 \text{ or } \infty; \\ \mathrm{ind}(f(s_iG)), & \text{otherwise}; \end{cases} \quad (9)$$

where $i = 0, 1, \cdots, T^n - 2$ and $f \in \mathbb{F}_p(\mathcal{E})$ is a rational function. Combining with two facts above, we derive the following statement from Lemma 2.

Theorem 5. *The discrepancy $\mathcal{D}_{\mathcal{W}}(T^n - 1)$ of $\mathcal{W}$ in (9) is bounded by*

$$\mathcal{D}_{\mathcal{W}}(T^n - 1) \ll T^{-1}p^{1/2}\log(p),$$

where the implied constant depends on $\#\mathrm{Supp}(f)$.

Proof. We have

$$\begin{aligned} \left|\sum_{n=0}^{T^n-2} e_{p-1}(aw_n)\right| &= \left|\sum_{n=0}^{T^n-2} \chi_a(f(s_nG))\right| + O(1) \\ &\ll T^{n-1}\left|\sum_{Q\in\langle G\rangle} \chi_a(f(Q))\right| \\ &\ll T^{n-1}\#\mathrm{Supp}(f)p^{1/2}, \end{aligned}$$

which completes the proof by Lemma 3. □

In computer science, the most common case is to use binary sequences. So we will consider binary sequences from elliptic curves.

Let

$$\rho : \{0, 1, \cdots, p-2\} \to \mathbb{F}_2$$

be a map. We will obtain a binary sequence $\mathcal{X} = (x_i)_{i \geq 0}$ from $\mathcal{W}$ defined by (9):

$$x_i = \rho(w_i), \ \ i \geq 0. \tag{10}$$

From (8) and (9), we set

$$\mathcal{X}_j = (x_{j+dk})_{k \geq 0}, \quad j = 0, 1, \cdots, d-1$$

where $d = (T^n - 1)/(T - 1)$. In fact, all $\mathcal{X}_j$'s are $d-$decimation of $\mathcal{X}$ and all nonconstant $\mathcal{X}_j$'s are shift equivalent. From [2] or [9], we have the following two statements on the binary sequence $\mathcal{X}$.

Proposition 1. *The period of $\mathcal{X}$,* per($\mathcal{X}$)*, is*

$$\text{per}(\mathcal{X}) = d \cdot \text{per}(\mathcal{X}_j),$$

where $\mathcal{X}_j$ is a nonconstant sequence and $\text{per}(\mathcal{X}_j) | T - 1$.

Proposition 2. *The linear complexity of $\mathcal{X}$, $L(\mathcal{X})$, is*

$$L(\mathcal{X}) = d \cdot L(\mathcal{X}_j),$$

where $\mathcal{X}_j$ is a nonconstant sequence.

For any nonconstant $\mathcal{X}_j$, we have $L(\mathcal{X}_j) \geq 1$, so the linear complexity of $\mathcal{X}$ is at least $d = (T^n - 1)/(T - 1)$, which is very close the period $T^n - 1$. It is interesting to determine the linear complexity of $\mathcal{X}_j$.

5 Conclusion

We have presented general results about the (multi-dimensional) distribution of a family of sequences derived from elliptic curves by using discrete logarithm, which indicate the resulting sequences are asymptotically uniformly distributed. A lower bound on the linear complexity of multi-dimensional sequences is also presented. We have also investigated sequences derived from elliptic curves by applying linear recurrence relations, the periods of which can be made very long. In Section 4, the sequence is produced using a primitive polynomial $m(x)$. It is interesting to consider the case that $m(x)$ is not a primitive polynomial and it seems more complicated.

We should note that computing a discrete logarithm in a finite field $\mathbb{F}_p$ is not an easy task (the best known algorithm runs in time sub-exponential in $\log p$), hence these random generators seem to be extremely slow. But, for example, one can output the least significant bit (not the whole bits) of discrete logarithms by only computing the Legendre symbol in polynomial time.

Acknowledgment

The authors wish to thank the anonymous referees for their detailed and very helpful comments and suggestions that improved this article.

References

1. Beelen, P.H.T., Doumen, J.M.: Pseudorandom Sequences from Elliptic Curves. In: Finite Fields with Applications to Coding Theory, Cryptography and Related Areas, pp. 37–52. Springer, Berlin, Heidelberg, New York (2002)
2. Chan, A.H., Games, R.A.: On the Linear Span of Binary Sequences Obtained from $q-$ary $m-$sequences, q Odd. IEEE Trans. on Information Theory 36(3), 548–552 (1990)
3. Chen, Z., Li, S., Xiao, G.: Construction of Pseudo-Random Binary Sequences from Elliptic Curves by Using Discrete Logarithm. In: Gong, G., Helleseth, T., Song, H.-Y., Yang, K. (eds.) SETA 2006. LNCS, vol. 4086, pp. 285–294. Springer, Heidelberg (2006)
4. Chen, Z., Xiao, G.: 'Good' Pseudo-Random Binary Sequences from Elliptic Curves. Cryptology ePrint Archive, Report 2007/275 (2007), `http://eprint.iacr.org/`
5. Cusick, T.W., Ding, C., Renvall, A.: Stream Ciphers and Number Theory. Elsevier, Amsterdam (2003)
6. Drmota, M., Tichy, R.F.: Sequences, Discrepancies and Applications. Lecture Notes in Mathematics, vol. 1651. Springer, Berlin, Heidelberg, New York (1997)
7. Enge, A.: Elliptic Curves and Their Applications to Cryptography: an Introduction. Kluwer Academic Publishers, Dordrecht (1999)
8. Gong, G., Berson, T., Stinson, D.: Elliptic Curve Pseudorandom Sequence Generator. Technical Reports, No. CORR1998-53 (1998), `http://www.cacr.math.uwaterloo.ca`
9. Gong, G., Lam, C.Y.: Linear Recursive Sequences over Elliptic Curves. In: Proceedings of Sequences and Their Applications-SETA 2001. DMTCS series, pp. 182–196. Springer, Berlin, Heidelberg, New York (2001)
10. Gyarmati, K.: On a Family of Pseudorandom Binary Sequences. Periodica Mathematica Hungarica 49(2), 45–63 (2004)
11. Hallgren, S.: Linear Congruential Generators over Elliptic Curves. Technical Report, No. CS-94-143, Cornegie Mellon University (1994)
12. Hess, F., Shparlinski, I.E.: On the Linear Complexity and Multidimensional Distribution of Congruential Generators over Elliptic Curves. Designs, Codes and Cryptography 35(1), 111–117 (2005)
13. Lam, C.Y., Gong, G.: Randomness of Elliptic Curve Sequences. Technical Reports, No. CORR 2002-18 (2002), `http://www.cacr.math.Uwaterloo.ca`
14. Lange, T., Shparlinski, I.E.: Certain Exponential Sums and Random Walks on Elliptic Curves. Canad. J. Math. 57(2), 338–350 (2005)
15. Lange, T., Shparlinski, I.E.: Distribution of Some Sequences of Points on Elliptic Curves. J. Math. Cryptology 1, 1–11 (2007)
16. Lee, L., Wong, K.: An Elliptic Curve Random Number Generator. In: Communications and Multimedia Security Issues of the New Century. In: Fifth Joint Working Conference on Communications and Multimedia Security-CMS 2001, pp. 127–133 (2001)
17. Lidl, R., Niederreiter, H.: Finite fields. Addison-Wesley, Reading (1983)

18. El Mahassni, E., Shparlinski, I.E.: On the Uniformity of Distribution of Congruential Generators over Elliptic Curves. In: Proc. Intern. Conf. on Sequences and Their Applications-SETA 2001, pp. 257–264. Springer, Berlin, Heidelberg, New York (2002)
19. Perret, M.: Multiplicative Character Sums and Nonlinear Geometric Codes. In: Eurocode 1990, pp. 158–165. Springer, Berlin, Heidelberg, New York (1991)
20. Perret, M.: Multiplicative Character Sums and Kummer Coverings. Acta Arithmetica 59, 279–290 (1991)
21. Sárközy, A.: A Finite Pseudorandom Binary Sequence. Studia Sci. Math. Hungar. 38, 377–384 (2001)
22. Shparlinski, I.E.: On the Naor-Reingold Pseudo-Random Number Function from Elliptic Curves. Appl. Algebra Engng. Comm. Comput. 11(1), 27–34 (2000)
23. Shparlinski, I.E.: Cryptographic Applications of Analytic Number Theory: Complexity Lower Bounds and Pseudorandomness. Progress in Computer Science and Applied Logic, vol. 22. Birkhauser, Basel (2003)
24. Shparlinski I. E.: Pseudorandom Points on Elliptic Curves over Finite Fields. Preprint No.33, pp. 1–15 (2005), `http://www.ics.mq.edu.au/~igor/publ.html`
25. Shparlinski, I.E., Silverman, J.H.: On the Linear Complexity of the Naor-Reingold Pseudo-random Function from Elliptic Curves. Designs, Codes and Cryptography 24(3), 279–289 (2001)
26. Vlăduţ, S.G.: Cyclicity Statistics for Elliptic Curves over Finite Fields. Finite Fields and Their Applications 5(1), 13–25 (1999)
27. Vlăduţ, S.G.: On the Cyclicity of Elliptic Curves over Finite Field Extensions. Finite Fields and Their Applications 5(3), 354–363 (1999)

Improved Bounds on the Linear Complexity of Keystreams Obtained by Filter Generators

Nicholas Kolokotronis*, Konstantinos Limniotis, and Nicholas Kalouptsidis

Department of Informatics and Telecommunications
National and Kapodistrian University of Athens
TYPA Buildings, University Campus, 15784 Athens, Greece
{nkolok,klimn,kalou}@di.uoa.gr

Abstract. Binary sequences generated by nonlinearly filtering maximal length sequences with period $2^n - 1$ are studied in this paper. We focus on the particular class of *normal filters* and provide improved lower bounds on the linear complexity of generated keystreams. This is achieved by first proving properties of a special class of determinants which are associated to *linearized polynomials* over finite fields of characteristic 2 and then by applying the above to simplify generalizations of the *root presence test*.

Keywords: Binary sequences, filter functions, linear complexity, linear feedback shift registers, linearized polynomials, stream ciphers.

1 Introduction

Stream ciphers are widely used to provide confidentiality in environments characterized by a limited computing power or memory capacity, and the need to encrypt at high speed. *Shift registers* of linear (LFSR) or nonlinear (NFSR) feedback are a basic building block of proposals in this area, like Trivium [4], Achterbahn [5] and Grain [11]. The *linear complexity*, i.e. the length of the shortest LFSR generating a given sequence, is important for assessing resistance to cryptanalytic attacks, like the *Berlekamp–Massey algorithm* [19], but also *algebraic attacks* [21]. As most constructions are ad-hoc, finding good generators (which is the objective of this work using finite fields tools) is of great theoretical and practical value.

High linear complexity keystreams are generated by applying Boolean functions either as *combiners* [7], [22], or *filters* [2], [22]. In any case, the highest value attainable by linear complexity depends on the degree of the function [10], [12]. The problem of determining the exact linear complexity attained by filterings is open. Two classes of filters have been introduced, namely *equidistant* and *normal*, that allow to derive lower bounds on the linear complexity. In the first case, the distance between successive phases is coprime to the period $N = 2^n - 1$

* This work was partially supported by a Greece–Britain *Joint Research & Technology Programme*, co-funded by the Greek General Secretariat for Research & Technology (GSRT) and the British Council, under contract GSRT132–C.

of m-sequences [6], [20], [22], whilst in the latter, phases are chosen from cosets associated with normal bases [15]. Given a filter of degree k, the best known lower bound on the linear complexity of keystreams is $\binom{n}{k} + \binom{n}{k-1}$ but only for equidistant filters [14]; the respective lower bound for normal filters equals $\binom{n}{k}$ [15]. These results rely on the so-called *root presence test* (RPT) [22].

In this paper, we focus on nonlinearly filtering m-sequences of period $N = 2^n - 1$ by *normal filters* of degree k. The work in [13], [22] is extended by deriving a simple RPT for field elements α^e, with e having Hamming weight $k-1$. This is achieved by factoring a class of determinants called *generalized linearized determinants* [16], and use it to prove the improved bound $\binom{n}{k} + n$ for such nonlinear filters. The paper is organized as follows: Section 2 gives the basic background and settles the notation. Properties of the generalized linearized determinants are studied in Section 3, while the improved lower bounds on normal filterings are derived in Section 4. Section 5 summarizes the results and gives possible future directions.

2 Background

Let $x = \{x_j\}_{j \geq 0}$ be a binary m-sequence of period $N = 2^n - 1$, and $\mu(z)$ its minimal polynomial, $\deg(\mu) = n$, whose roots lie in the extension field $\mathbb{F}_{2^n}$ [16]. It is known that the linear complexity L_x of x equals n [8], [9], [22]. Let $\alpha \in \mathbb{F}_{2^n}$ be a primitive element of $\mathbb{F}_{2^n}$ with $\mu(\alpha^{-1}) = 0$. Then

$$x_j = \mathrm{tr}_1^n(\beta\alpha^{-j}) = \beta\alpha^{-j} + \left(\beta\alpha^{-j}\right)^2 + \cdots + \left(\beta\alpha^{-j}\right)^{2^{n-1}} \tag{1}$$

for some $\beta \in \mathbb{F}_{2^n}$, where $\mathrm{tr}_1^n : \mathbb{F}_{2^n} \to \mathbb{F}_2$ is the *trace function*. The above, known as the *trace representation*, is associated with the discrete Fourier transform of x, as it always holds $x_j = \sum_{i=0}^{N-1} \beta_i \alpha^{-ij}$, $\beta_i \in \mathbb{F}_{2^n}$ [13], [20].

Let $y = \{y_j\}_{j \geq 0}$ be the sequence resulting from the nonlinear filtering of x via $h : \mathbb{F}_2^n \to \mathbb{F}_2$. Then, we have $y_j = h(x_{j-t_1}, x_{j-t_2}, \ldots, x_{j-t_n})$, where the phases t_i belong to the residue class ring $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$. Let $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$, $\boldsymbol{r} = (r_1, r_2, \ldots, r_n) \in \mathbb{F}_2^n$. It is common to express the Boolean function h in its *algebraic normal form* (ANF), given by

$$h(\boldsymbol{z}) = \sum_{\boldsymbol{r} \in \mathbb{F}_2^n} a_{\boldsymbol{r}} \boldsymbol{z}^{\boldsymbol{r}} = \sum_{\boldsymbol{r} \in \mathbb{F}_2^n} a_{\boldsymbol{r}} z_1^{r_1} z_2^{r_2} \cdots z_n^{r_n}, \qquad a_{\boldsymbol{r}} \in \mathbb{F}_2 \,. \tag{2}$$

In the sequel, we assume that $a_{\boldsymbol{0}} = 0$. The *degree* of function h is defined as $\deg(h) = \max\{\mathrm{wt}(\boldsymbol{r}) : a_{\boldsymbol{r}} = 1, \boldsymbol{r} \in \mathbb{F}_2^n\}$, where $\mathrm{wt}(\boldsymbol{r})$ denotes the weight of vector $\boldsymbol{r}$. A special form of functions to be considered in the following sections are the *elementary symmetric polynomials* of degree s, defined as $\sigma_s(\boldsymbol{z}) = \sum_{\mathrm{wt}(\boldsymbol{r})=s} z_1^{r_1} \cdots z_n^{r_n}$, where $\boldsymbol{r} \in \mathbb{F}_2^n$ [17]. Subsequently, we use the convention that $\sigma_s(\boldsymbol{z}) = 0$ if $s < 0$ or $s > n$; clearly, $\sigma_0(\boldsymbol{z}) = 1$.

The *cyclotomic coset* C_e of $e \in \mathbb{Z}_N$ is the set of distinct elements in $\{e, e2, \ldots, e2^{n-1}\}$ modulo N, and its cardinality is a divisor of n [16]. We say that

$\alpha^e \in \mathbb{F}_{2^n}$ is a *normal element* if $\{\alpha^e, \alpha^{e2}, \dots, \alpha^{e2^{n-1}}\}$ is a normal basis of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$, and also $\mathrm{wt}(\alpha^e) = s$ if and only if $\mathrm{wt}(e) = s$, that is $e = 2^{e_0} + \cdots + 2^{e_{s-1}}$. It is known that $L_y \le \sum_{i=1}^{k} \binom{n}{i}$, if $\deg(h) = k$ [12]. In the special case $y_j = x_{j-t_1} \cdots x_{j-t_k}$, the RPT for $\mathrm{wt}(\alpha^e) = k$ is given by $T_e = \det(\alpha^{t_i 2^{e_{j-1}}})_{i,j=1}^{k}$ [22]; it ensures that α^e is a root of the minimal polynomial of y whenever $T_e \neq 0$. The RPT has also been formed for the case of $\mathrm{wt}(\alpha^e) = k-1$ [13], and is given by

$$T_e = \sum_{1 \le r < s \le k} \det(\alpha^{l_i 2^{e_{j-1}+1}})_{i,j=1}^{k-1} \tag{3}$$

where $(l_1, \dots, l_{k-1}) = (t_1, \dots, t_{r-1}, t_{r+1}, \dots, t_{s-1}, t_{s+1}, \dots, t_k, \frac{1}{2}(t_r + t_s))$, $t_i \in \mathbb{Z}_N$. Next, we develop the tools to simplify (3), for the case of *normal filters*, based on properties of linearized polynomials over binary extension fields and apply the results to the design of such filters.

3 Generalized Linearized Determinants

Let $\boldsymbol{x} = (x_1, \dots, x_k)$, with $x_i \in \mathbb{F}_{2^n}$, be a vector of nonzero elements, and $R = \{r_1, \dots, r_k\}$ be an increasing sequence of nonnegative integers. Then

$$\boldsymbol{U}(\boldsymbol{x}; R) = \det(x_j^{2^{r_i}})_{i,j=1}^{k} \tag{4}$$

is called *generalized linearized determinant*; the choice $r_i = i-1$ will be denoted by $\boldsymbol{U}(\boldsymbol{x})$. They are connected to linearized polynomials that have numerous applications in coding and cryptography (*see* e.g. [16]).

Theorem 1 ([16, p. 109]). *Let $\boldsymbol{x} = (x_1, \dots, x_k) \in \mathbb{F}_{2^n}^k \setminus \{\boldsymbol{0}\}$. Then*

$$\boldsymbol{U}(\boldsymbol{x}) = \prod_{\boldsymbol{c} \in \mathbb{F}_2^k \setminus \{\boldsymbol{0}\}} (c_1 x_1 + \cdots + c_k x_k) = \prod_{\boldsymbol{c} \in \mathbb{F}_2^k \setminus \{\boldsymbol{0}\}} \langle \boldsymbol{c}, \boldsymbol{x} \rangle \tag{5}$$

is nonzero if and only if $x_1, \dots, x_k$ are linearly independent over $\mathbb{F}_2$.

Let $I = \{0, 1, \dots, r_k\} \setminus R$ be the set of *discontinuities* that appear among the elements of R. When $|I| < k$, we find it convenient to write $\boldsymbol{U}_\perp(\boldsymbol{x}; I)$ instead of $\boldsymbol{U}(\boldsymbol{x}; R)$. Furthermore, we introduce the mapping φ as follows $\varphi(\boldsymbol{x}) = \varphi(x_1, \dots, x_k) = (0, x_1, x_2, x_1 + x_2, \dots, x_1 + \cdots + x_k)$ that maps $\boldsymbol{x}$ to a vector comprised by all linear combinations between the elements of $\boldsymbol{x}$ in lexicographic ordering. Hence, (5) is written as $\boldsymbol{U}(\boldsymbol{x}) = \sigma_{2^k-1}(\varphi(\boldsymbol{x}))$.

Lemma 2. *Let us assume that $I = \{l\}$, where $0 \le l \le k$. Then, we have $\boldsymbol{U}_\perp(\boldsymbol{x}; I) = \boldsymbol{U}(\boldsymbol{x}) \sigma_{2^k - 2^l}(\varphi(\boldsymbol{x}))$.*

Proof. Let us define the polynomial $g(z) = \boldsymbol{U}(\boldsymbol{x}, z) \in \mathbb{F}_{2^n}[z]$, in terms of the linearized determinant $\boldsymbol{U}(\boldsymbol{x}, z)$ of order $k+1$. From (5) we have

$$g(z) = \boldsymbol{U}(\boldsymbol{x}) \prod_{\boldsymbol{c} \in \mathbb{F}_2^k} (\langle \boldsymbol{c}, \boldsymbol{x} \rangle + z) = \boldsymbol{U}(\boldsymbol{x}) \sum_{i=0}^{2^k} \sigma_{2^k - i}(\varphi(\boldsymbol{x})) z^i \, . \tag{6}$$

On the other hand, by expanding $\boldsymbol{U}(\boldsymbol{x}, z)$ along its $k+1$ column, we have $g(z) = \sum_{i=0}^{k} \boldsymbol{U}_{\perp}(\boldsymbol{x}; \{i\}) z^{2^i}$; comparison with (6) proves the claim. □

The following result is a direct consequence of Lemma 2.

Corollary 3. *The elementary symmetric polynomial $\sigma_s(\varphi(\boldsymbol{x}))$ of degree s, with $0 \leq s \leq 2^k$, is zero for all $s \notin \{2^k - 2^i : 0 \leq i \leq k\}$.*

To establish the main result of this section, we first need some properties of symmetric polynomials $\sigma_{2^k-2^i}(\varphi(\boldsymbol{x}))$, which are stated without a proof; the convention $\sigma_{2^k-2^i}(\varphi(\boldsymbol{x})) = 0$, if $i < 0$ or $i > k$, is subsequently used.

Proposition 4. *Let $\{A_1, A_2\}$ be a partition of the set $\{1, 2, \ldots, k\}$. Then $\sigma_i(\boldsymbol{x}) = \sum_{j_1+j_2=i} \sigma_{j_1}(\boldsymbol{x}_{A_1}) \sigma_{j_2}(\boldsymbol{x}_{A_2})$, where $\boldsymbol{x}_{A_r}$ is the vector comprised by variables with indices in A_r.*

Proposition 5. *Let $\boldsymbol{x} + z = (x_1 + z, \ldots, x_k + z)$, for any $z \in \mathbb{F}_{2^n}$. Then $\sigma_i(\boldsymbol{x} + z) = \sum_{j=0}^{i} \binom{j+k-i}{j} \sigma_{i-j}(\boldsymbol{x}) z^j$.*

Lemma 6. *With the above notation, for all $0 \leq i \leq k+1$, we have that $\sigma_{2^{k+1}-2^i}(\varphi(\boldsymbol{x}, z)) = \sigma_{2^k-2^{i-1}}(\varphi(\boldsymbol{x}))^2 + \sigma_{2^k-2^i}(\varphi(\boldsymbol{x})) \frac{\boldsymbol{U}(\boldsymbol{x},z)}{\boldsymbol{U}(\boldsymbol{x})}$.*

Proof. From the definition of $\varphi(\cdot)$ we get that $\varphi(\boldsymbol{x}, z) = (\varphi(\boldsymbol{x}), \varphi(\boldsymbol{x}) + z)$, whereas by Corollary 3 and Propositions 4, 5 we easily see that we have

$$\sigma_{2^{k+1}-2^i}(\varphi(\boldsymbol{x}, z)) = \sum_{l=0}^{k} \sigma_{2^k-2^l}(\varphi(\boldsymbol{x})) \sum_{j=2^i-2^l}^{2^k} \binom{j}{j+2^l-2^i} \sigma_{2^k-j}(\varphi(\boldsymbol{x})) \times z^{j+2^l-2^i} .$$

Since j must be a power of two by Corollary 3, say $j = 2^r$ for some $r \leq k$, Lucas' Theorem (*see* e.g. [1, p. 113], [18, p. 404]) implies that $\binom{2^r}{2^r+2^l-2^i}$ is nonzero if and only if either of the following cases hold

a. $l = i$: we get the expression $\sigma_{2^k-2^i}(\varphi(\boldsymbol{x})) \frac{\boldsymbol{U}(\boldsymbol{x},z)}{\boldsymbol{U}(\boldsymbol{x})}$ by Lemma 2; or
b. $l = r = i - 1$: where we only get the term $\sigma_{2^k-2^{i-1}}(\varphi(\boldsymbol{x}))^2$.

By combining the above two cases, we establish the required identity. □

If it holds $z = c_1 x_1 + \cdots + c_k x_k$ for some $\boldsymbol{c} \in \mathbb{F}_2^k$, then Lemma 6 gives the identity $\sigma_{2^{k+1}-2^i}(\varphi(\boldsymbol{x}, z)) = \sigma_{2^k-2^{i-1}}(\varphi(\boldsymbol{x}))^2$ since then $\varphi(\boldsymbol{x}) + z$ is just a permutation of the elements in $\varphi(\boldsymbol{x})$. The factorization of $\boldsymbol{U}(\boldsymbol{x}; R)$ is next generalized for any number of elements in I.

Theorem 7. *Let $I = \{l_1, \ldots, l_s\}$, where $0 \leq l_1 < \cdots < l_s \leq k + s - 1$. Then $\boldsymbol{U}_{\perp}(\boldsymbol{x}; I) = \boldsymbol{U}(\boldsymbol{x}) \det(\sigma_{2^k-2^{l_i-j+1}}(\varphi(\boldsymbol{x}))^{2^{j-1}})_{i,j=1}^{s}$.*

Proof (sketch). Let us denote $\sigma_{2^k-2^i}(\varphi(\boldsymbol{x}))$ by $\xi_{k,i}(\boldsymbol{x})$, for simplicity. We proceed by induction on the cardinality of I and apply arguments similar to the ones developed in [14, Theorem 3]. The identity has been proved in Lemma 2 for $|I| = 1$ and assume it also holds for $|I| = m$. Then from the induction hypothesis and Lemma 6, $g(z) = \boldsymbol{U}_{\perp}((\boldsymbol{x}, z); I)$ becomes

$$\begin{aligned} g(z) &= \boldsymbol{U}(\boldsymbol{x}, z) \det\big(\xi_{k+1,l_i-j+1}(\boldsymbol{x}, z)^{2^{j-1}}\big)_{i,j=1}^{m} \\ &= \boldsymbol{U}(\boldsymbol{x}) f(z) \sum_{\pi \in \mathcal{P}_m} \prod_{i=1}^{m} \Big(\xi_{k,l_{\pi_i}-i}(\boldsymbol{x})^2 + f(z)\xi_{k,l_{\pi_i}-i+1}(\boldsymbol{x})\Big)^{2^{i-1}} \\ &= \boldsymbol{U}(\boldsymbol{x}) \sum_{\boldsymbol{c} \in \mathbb{F}_2^m} \det\big(\xi_{k,l_i-j+c_j}(\boldsymbol{x})^{2^{j-c_j}}\big)_{i,j=1}^{m} f(z)^{c+1} \end{aligned} \tag{7}$$

where $\mathcal{P}_m$ is the set of permutations of $\{1, \dots, m\}$, $\boldsymbol{c} = (c_1, \dots, c_m)$ is the binary representation of $c \in [0, 2^m - 1]$, and $f(z) = \boldsymbol{U}(\boldsymbol{x}, z)\boldsymbol{U}(\boldsymbol{x})^{-1}$. Since the determinants involved in (7) vanish if $c \neq 2^t - 1$ [14], by substituting $f(z) = \sum_{v=0}^{k} \xi_{k,v}(\boldsymbol{x}) z^{2^v}$ and performing some easy calculations, we get

$$g(z) = \boldsymbol{U}(\boldsymbol{x}) \sum_{l_{m+1} \in R'} \det\big(\xi_{k,l_i-j+1}(\boldsymbol{x})^{2^{j-1}}\big)_{i,j=1}^{m+1} z^{2^{l_{m+1}}} \tag{8}$$

where $R' = \{0, \dots, k+m\} \setminus I = R \cup \{k+m\}$. On the other hand, we have

$$g(z) = \sum_{i=1}^{k+1} \boldsymbol{U}_{\perp}(\boldsymbol{x}; I \cup \{r_i\}) z^{2^{r_i}} = \sum_{l_{m+1} \in R'} \boldsymbol{U}_{\perp}(\boldsymbol{x}; I') z^{2^{l_{m+1}}} \tag{9}$$

where $I' = I \cup \{l_{m+1}\}$. Direct comparison of the coefficients in (8), (9) concludes our proof. □

4 Improved Bounds on Normal Filterings

In this section, we establish a new lower bound on the linear complexity of sequences obtained from filter generators, focusing on the case of maximal length sequences filtered by nonlinear Boolean functions of degree k that have the form $h(z_1, \dots, z_k) = z_1 \cdots z_k$, that is, we assume that the output sequence y is determined by the product

$$y_j = x_{j-t-d} x_{j-t-d2} \cdots x_{j-t-d2^{k-1}}, \qquad j \geq 0 \tag{10}$$

where $t \geq 0$, and the integer $d \in \mathbb{Z}_N \setminus \{0\}$ is chosen so that the cyclotomic coset $C_d = \{d, d2, \dots, d2^{n-1}\}$ corresponds to a normal basis of $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$. It is clear that the construction introduced above cannot be handled by the methodology developed in [14] based on generalized Vandermonde determinants, since the cardinality of I in that case (cf. equidistant filters) would be prohibitively large. Thus, the results of Section 3 are valuable in analyzing the linear complexity of keystreams obtained by such ciphers. In the sequel, we consider the root presence test only for elements $\alpha^e \in \mathbb{F}_{2^n}$ of weight $k-1$, i.e. (3), since it has already been proved that $L_y \geq \binom{n}{k}$ in the case of $\mathrm{wt}(e) = k$ [13]. Moreover, we write $e = 2^{e_0} + \cdots + 2^{e_{k-2}}$ and use the notation $\boldsymbol{x}_s = (x_1, \ldots, x_{s-1}, x_{s+1}, \ldots, x_k)$, i.e. $\boldsymbol{x}_s$ results from $\boldsymbol{x}$ by omitting the variable x_s, $1 \leq s \leq k$.

Theorem 8. *Let sequence y be generated by* (10)*, and let $\alpha^e \in \mathbb{F}_{2^n}$ with* $\mathrm{wt}(e) =$ $k-1$*. Then, α^e is a root of the minimal polynomial of y if and only if α^d is not a root of*

$$f_e(z) = \sum_{i=0}^{k-2} \prod_{\boldsymbol{c}_i \in \mathbb{F}_2^{k-2}} \frac{\left(g_{e,\boldsymbol{c}_i}(z) + z^{2^{e_i - 1}}\right)^3}{g_{e,\boldsymbol{c}_i}(z) + z^{2^{e_i}}} \tag{11}$$

where $g_{e,\boldsymbol{c}_i}(z) = \sum_{j \neq i} c_j z^{2^{e_j}}$ and $\boldsymbol{c}_i = (c_0, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{k-2})$.

Proof. Substituting $t_i = t + d2^i$, for $0 \leq i \leq k-1$, in (3) and expanding the determinants along their last row, the root presence test becomes by Theorem 7 as follows

$$T_e = \alpha^{2et} \sum_{0 \leq i < j < k} \sum_{s=0}^{k-2} \boldsymbol{U}_{\perp}(\boldsymbol{x}_s; \{i,j\})^2 x_s^{2^i + 2^j} = \alpha^{2et} \sum_{0 \leq i < j < k} \sum_{s=0}^{k-2} \boldsymbol{U}(\boldsymbol{x}_s)^2$$
$$\times \begin{vmatrix} \sigma_{2^{k-2}-2^i}(\varphi(\boldsymbol{x}_s)) & \sigma_{2^{k-2}-2^{i-1}}(\varphi(\boldsymbol{x}_s))^2 \\ \sigma_{2^{k-2}-2^j}(\varphi(\boldsymbol{x}_s)) & \sigma_{2^{k-2}-2^{j-1}}(\varphi(\boldsymbol{x}_s))^2 \end{vmatrix}^2 x_s^{2^i + 2^j} \tag{12}$$

where $\boldsymbol{x} = \left(\alpha^{d2^{e_0}}, \ldots, \alpha^{d2^{e_{k-2}}}\right)$, and determinant $\boldsymbol{U}_{\perp}(\boldsymbol{x}_s; \{i,j\})$ has order $k-2$. By expanding in (12) the 2×2 determinant of elementary symmetric polynomials, and using the fact that it vanishes for $i = j$, we obtain that

$$\begin{aligned} T_e &= \alpha^{2et} \sum_{s=0}^{k-2} \boldsymbol{U}(\boldsymbol{x}_s)^2 \sum_{i,j=0}^{k-1} \sigma_{2^{k-2}-2^i}(\varphi(\boldsymbol{x}_s))^2 \sigma_{2^{k-2}-2^{j-1}}(\varphi(\boldsymbol{x}_s))^4 x_s^{2^i + 2^j} \\ &= \alpha^{2et} \sum_{s=0}^{k-2} \boldsymbol{U}(\boldsymbol{x}_s)^2 \sum_{i=0}^{k-2} \sigma_{2^{k-2}-2^i}(\varphi(\boldsymbol{x}_s))^2 x_s^{2^i} \sum_{j=1}^{k-1} \sigma_{2^{k-2}-2^{j-1}}(\varphi(\boldsymbol{x}_s))^4 x_s^{2^j} \\ &= \alpha^{2et} \sum_{s=0}^{k-2} \boldsymbol{U}(\boldsymbol{x}_s)^2 \sum_{i=0}^{k-2} \sigma_{2^{k-2}-2^i}(\varphi(\boldsymbol{x}_s))^2 x_s^{2^i} \sum_{l=0}^{k-2} \sigma_{2^{k-2}-2^l}(\varphi(\boldsymbol{x}_s))^4 x_s^{2^{l+1}} \\ &= \alpha^{2et} \sum_{s=0}^{k-2} \boldsymbol{U}(\boldsymbol{x}_s)^2 \left(\sum_{i=0}^{k-2} \sigma_{2^{k-2}-2^i}(\varphi(\boldsymbol{x}_s)) \left(\sqrt{x_s}\right)^{2^i} \right)^6 \end{aligned} \tag{13}$$

since $\sigma_{2^{k-2}-2^i}(\varphi(\boldsymbol{x}_s))$ and $\sigma_{2^{k-2}-2^{j-1}}(\varphi(\boldsymbol{x}_s))$ are zero when $i = k-1$ and $j = 0$ respectively, where the change of variables $j \mapsto l+1$ was applied in the rightmost terms. From the definition of $\boldsymbol{U}(\boldsymbol{x})$ we see that the identity $\boldsymbol{U}(\boldsymbol{x}) = \boldsymbol{U}(\boldsymbol{x}_s) \prod_{\boldsymbol{c} \in \mathbb{F}_2^{k-2}} \left(\langle \boldsymbol{c}, \boldsymbol{x}_s \rangle + x_s\right)$ holds. Additionally, from the proof of Corollary 3, (13) becomes

$$T_e = \alpha^{2et} \boldsymbol{U}(\boldsymbol{x})^2 \sum_{s=0}^{k-2} \prod_{\boldsymbol{c} \in \mathbb{F}_2^{k-2}} \frac{\left(\langle \boldsymbol{c}, \boldsymbol{x}_s \rangle + \sqrt{x_s}\right)^6}{\left(\langle \boldsymbol{c}, \boldsymbol{x}_s \rangle + x_s\right)^2} = \alpha^{2et} \boldsymbol{U}(\boldsymbol{x})^2 f_e(\alpha^d)^2 \tag{14}$$

since $x_i = \alpha^{d2^{e_i}}$ where $f_e(z)$ is given by (11). It is clear that $T_e \neq 0$ if and only if $f_e(\alpha^d) \neq 0$. □

Proposition 9. *Let $f_e(z)$ be given by* (11)*, and let $l \geq 0$. Then, we have*

1. *$f_e(\alpha^{d2^l}) \neq 0$ if and only if $f_e(\alpha^d) \neq 0$, and*
2. *$f_{e2^l}(\alpha^d) \neq 0$ if and only if $f_e(\alpha^d) \neq 0$.*

Proof. This is direct result of (11), since for all integers $l \geq 0$ the identity $f_e(\alpha^{d2^l}) = f_e(\alpha^d)^{2^l} = f_{e2^l}(\alpha^d)$ holds. □

From (11), we see that the function $f_e(z)$ is well-defined if and only if all linear combinations of any $k-1$ elements $\alpha^{d2^{e_0}}, \ldots, \alpha^{d2^{e_{k-2}}}$ from the set $\{\alpha^d, \alpha^{d2}, \ldots, \alpha^{d2^{n-1}}\}$ are nonzero; this is satisfied since by hypothesis α^d is a normal element of the finite field $\mathbb{F}_{2^n}$. The simplification occurring in the root presence test, due to Theorem 8, allows the systematic analysis of the linear complexity of maximal length sequences nonlinearly filtered as in (10). In fact, according to Proposition 9, the root presence test need only be applied for $\alpha^e \in \mathbb{F}_{2^n}$, $\mathrm{wt}(e) = k-1$, where e is the coset leader of C_e (hence $e_0 = 0$). Next, $f_e(z)$ is further simplified by considering runs of 1s in the binary representation of e.

Corollary 10. *Let $\alpha^e \in \mathbb{F}_{2^n}$ with $\mathrm{wt}(e) = k-1$, and w be the number of runs of* 1*s in the binary representation of e. If the r-th run has length l_r and starts at position e_{i_r}, where $0 \leq i_0 < \cdots < i_{w-1} \leq k-2$, we have*

$$f_e(z) = \sum_{r=0}^{w-1} \prod_{\boldsymbol{c}_{i_r} \in \mathbb{F}_2^{k-2}} \frac{\left(g_{e,\boldsymbol{c}_{i_r}}(z) + z^{2^{e_{i_r}-1}}\right)^3}{g_{e,\boldsymbol{c}_{i_r}}(z) + z^{2^{e_{i_r}}}} . \tag{15}$$

Proof. It is clear from (11) that when two consecutive 1s are encountered in the binary representation of e, that is if $e_j \equiv e_{j-1 \bmod k-1} + 1 \pmod n$ for some $0 \leq j \leq k-2$, then for $i = j$ we get that

$$0 = z^{2^{e_{j-1}}} + z^{2^{e_j - 1}} \;\Big|\; \prod\nolimits_{\boldsymbol{c}_j \in \mathbb{F}_2^{k-2}} \left(g_{e,\boldsymbol{c}_j}(z) + z^{2^{e_j-1}}\right)^3 .$$

Hence, only those $0 \leq i \leq k-2$ for which e_i is the starting position of a run of 1s need to be considered in (11). □

Of particular interest is the case where the cyclotomic coset C_e of $e \in \mathbb{Z}_N$ is in the class of the so-called *regular cosets* [3]. These are cosets for which $\alpha^e \in \mathbb{F}_{2^n}$ belongs to subfields of the finite field $\mathbb{F}_{2^n}$. With the notation of Corollary 10, we see that in this case it holds $w \mid \gcd(n, k-1)$ and $w > 1$, with $l_r = l = \frac{k-1}{w}$ and $e_{i_r} = e_{rl} = rm = r\frac{n}{w}$, for $0 \leq r < w$; it is clear that $0 < l < m$. Moreover

$e_j = e_{sl+t} = e_{sl} + t = sm + t$ for some integers $0 \leq s < w$ and $0 \leq t < l$, where $0 \leq j \leq k-2$. From Corollary 10, we get

$$g_{e,\boldsymbol{c}_{i_r}}(z) = \sum_{\substack{s=0 \\ (s,t)\neq(r,0)}}^{w-1} \sum_{t=0}^{l-1} c_{sl+t} z^{2^{sm+t}} = \left(\sum_{\substack{s=0 \\ (s,t)\neq(r,0)}}^{w-1} \sum_{t=0}^{l-1} c_{sl+t} z^{2^{(s-r)m+t}} \right)^{2^{rm}}$$
$$= \left(\sum_{\substack{s=0 \\ (s,t)\neq(0,0)}}^{w-1} \sum_{t=0}^{l-1} c_{(s+r \bmod w)l+t} z^{2^{sm+t}} \right)^{2^{rm}} = g_{e,\tilde{\boldsymbol{c}}_0^r}(z)^{2^{rm}}$$

using that $z = z^{2^{wm}}$, where $\tilde{\boldsymbol{c}}_0^r = (c_{rl+1}, \ldots, c_{k-2}, c_0, \ldots, c_{rl-1})$. Thus, we have that $f_e(z) = \sum_{r=0}^{w-1} \left(\prod_{\tilde{\boldsymbol{c}}_0^r \in \mathbb{F}_2^{k-2}} (g_{e,\tilde{\boldsymbol{c}}_0^r}(z) + \sqrt{z})^3 / (g_{e,\tilde{\boldsymbol{c}}_0^r}(z) + z) \right)^{2^{rm}}$. A simple reordering of the product terms will make them independent of r, leading to the following result, simplifying the analysis of such filterings.

Corollary 11. *With the above notation, if* $e = 2^{e_0} + \cdots + 2^{e_{k-2}}$ *is such that* $e_j = sm + t$, *where* $s = \lfloor j/l \rfloor$ *and* $t = j \bmod l$, *then it holds*

$$f_e(z) = \mathrm{tr}_m^n \left(\prod_{c_1,\ldots,c_{k-2} \in \mathbb{F}_2} \frac{\left(g_{e,\boldsymbol{c}}(z) + \sqrt{z}\right)^3}{g_{e,\boldsymbol{c}}(z) + z} \right) \tag{16}$$

with $g_{e,\boldsymbol{c}}(z) = \sum_{j=1}^{k-2} c_j z^{2^{sm+t}}$ *and* $\boldsymbol{c} = (c_1, \ldots, c_{k-2})$.

Ensuring that $\alpha^d \in \mathbb{F}_{2^n}$ is not a root of $f_e(z)$, for all $e \in \mathbb{Z}_N$ with $\mathrm{wt}(e) = k-1$, to obtain the lower bound $\binom{n}{k} + \binom{n}{k-1}$ according to Theorem 8, is a hard task even for special cases of d (i.e. normal bases with some particular properties). Improved bounds have been obtained in the following case.

Theorem 12. *Let sequence* y *be given by* (10) *and* $2 \leq k \leq n$. *Then, for all* $d \in \mathbb{Z}_N$ *such that* $\alpha^d \in \mathbb{F}_{2^n}$ *is a normal element it holds* $L_y \geq \binom{n}{k} + n$.

Proof. For a normal filter of degree k let $e = 2^{k-1} - 1$, which has weight $k-1$; clearly, the cardinality of C_e equals n. From Corollary 10, we have that $f_e(\alpha^d)$ is comprised by a single product of terms whose numerator is

$$\prod\nolimits_{c_1,\ldots,c_{k-2} \in \mathbb{F}_2} \left(c_1 \alpha^{d2} + \cdots + c_{k-2} \alpha^{d2^{k-2}} + \alpha^{d2^{n-1}} \right)^3 .$$

Hence, $f_e(\alpha^d)$ is always nonzero since $\alpha^{d2}, \ldots, \alpha^{d2^{k-2}}, \alpha^{d2^{n-1}}$ are linearly independent due to hypothesis. □

The above methodology is easily extended to include the sum of normal filters, by adding shifted versions (i.e. for various values of the integer t) of the nonlinear filter given by (10). In this case, (14) leads to

$$T_e = \left(\sum\nolimits_{t \geq 0} v_t \alpha^{et} \right)^2 \boldsymbol{U}(\boldsymbol{x})^2 f_e(\alpha^d)^2, \qquad v_t \in \mathbb{F}_2 . \tag{17}$$

Thus, we also need to ensure that, for a particular choice of coefficients $v_t \in \mathbb{F}_2$, no element $\alpha^e \in \mathbb{F}_{2^n}$ with $\mathrm{wt}(e) = k - 1$ will be root of $\sum_{t \geq 0} v_t z^t$. Exhaustive search, for $2 \leq n \leq 20$, verified the above results and indicated that most degeneracies occur when $e \in \mathbb{Z}_N$ belongs to a regular coset.

5 Conclusions

Maximal length sequences nonlinearly filtered by means of normal filters were studied in this paper. It was shown that simple conditions for testing the presence of roots in the minimal polynomial of filterings can be derived by using properties of generalized linearized determinants. As a result, the improved lower bound $\binom{n}{k} + n$ on the linear complexity of filterings was obtained. Ongoing research focuses on exploiting properties of type I and II optimal normal bases in order to improve the attained lower bound.

References

1. Berlekamp, E.R.: Algebraic Coding Theory. McGraw-Hill, New York (1968)
2. Bernasconi, J., Günther, C.G.: Analysis of a nonlinear feedforward logic for binary sequence generators. In: Pichler, F. (ed.) EUROCRYPT 1985. LNCS, vol. 219, pp. 161–166. Springer, Heidelberg (1986)
3. Caballero-Gil, P.: Regular cosets and upper bounds on the linear complexity of certain sequences. In: Ding, C., et al. (eds.) Sequences and Their Applications. DMTCS, pp. 242–256. Springer, Heidelberg (1999)
4. De Cannière, C., Preneel, B.: Trivium – a stream cipher construction inspired by block cipher design principles. In: eSTREAM: ECRYPT Stream Cipher Project, Report 2005/030 (2005), http://www.ecrypt.eu.org/stream/
5. Gammel, B., Göttfert, R., Kniffler, O.: The Achterbahn stream cipher. In: eSTREAM: ECRYPT Stream Cipher Project, Report 2005/002 (2005), http://www.ecrypt.eu.org/stream/
6. García-Villalba, L.J., Fúster-Sabater, A.: On the linear complexity of the sequences generated by nonlinear filterings. Inform. Process. Lett. 76, 67–73 (2000)
7. Golić, J.D.: On the linear complexity of functions of periodic GF(q) sequences. IEEE Trans. Inform. Theory 35, 69–75 (1989)
8. Golomb, S.W.: Shift Register Sequences. Holden-Day, San Francisco (1967)
9. Göttfert, R., Niederreiter, H.: On the minimal polynomial of the product of linear recurring sequences. Finite Fields Applic. 1, 204–218 (1995)
10. Groth, E.J.: Generation of binary sequences with controllable complexity. IEEE Trans. Inform. Theory 17, 288–296 (1971)
11. Hell, M., Johansson, T., Meier, W.: Grain – a stream cipher for constrained environments. In eSTREAM: ECRYPT Stream Cipher Project, Report 2005/010 (2005), http://www.ecrypt.eu.org/stream/
12. Key, E.L.: An analysis of the structure and complexity of nonlinear binary sequence generators. IEEE Trans. Inform. Theory 22, 732–736 (1976)
13. Kolokotronis, N., Kalouptsidis, N.: On the linear complexity of nonlinearly filtered PN-sequences. IEEE Trans. Inform. Theory 49, 3047–3059 (2003)

14. Kolokotronis, N., Limniotis, K., Kalouptsidis, N.: Lower bounds on sequence complexity via generalised Vandermonde determinants. In: Gong, G., Helleseth, T., Song, H.-Y., Yang, K. (eds.) SETA 2006. LNCS, vol. 4086, pp. 271–284. Springer, Heidelberg (2006)
15. Lam, C., Gong, G.: A lower bound for the linear span of filtering sequences. In: State of the Art of Stream Ciphers – SASC (2004), pp. 220–233 (2004)
16. Lidl, R., Niederreiter, H.: Finite Fields. In: Encyclop. Math. Its Applic., 2nd edn., vol. 20, Cambridge Univ. Press, Cambridge (1996)
17. Macdonald, I.G.: Symmetric Functions and Hall Polynomials, 2nd edn. Oxford Univ. Press, Oxford (1995)
18. MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error Correcting Codes. North-Holland, Amsterdam (1977)
19. Massey, J.L.: Shift-register synthesis and BCH decoding. IEEE Trans. Inform. Theory 15, 122–127 (1969)
20. Paterson, K.G.: Root counting, the DFT and the linear complexity of nonlinear filtering. Des. Codes Cryptogr. 14, 247–259 (1998)
21. Rønjom, S., Helleseth, T.: A new attack on the filter generator. IEEE Trans. Inform. Theory 53, 1752–1758 (2007)
22. Rueppel, R.A.: Analysis and Design of Stream Ciphers. Springer, Berlin, Germany (1986)

Linear Equation on Polynomial Single Cycle T-Functions

Jin-Song Wang and Wen-Feng Qi[⋆]

Department of Applied Mathematics, Zhengzhou Information Engineering University,
P.O.Box 1001-745,
Zhengzhou, 450002, P.R. China
jinsong.wang@126.com, wenfeng.qi@263.net

Abstract. Polynomial functions are widely used in the design of cryptographic transformations such as block ciphers, hash functions and stream ciphers, which belong to the category of T-functions. When a polynomial function is used as state transition function in a pseudorandom generator, it is usually required that the polynomial function generates a single cycle. In this paper, we first present another proof of the sufficient and necessary condition on a polynomial function $f(\mathbf{x}) = c_0 + c_1\mathbf{x} + c_2\mathbf{x}^2 + \cdots + c_m\mathbf{x}^m \bmod 2^n (n \geq 3)$ being a single cycle T-function. Then we give a general linear equation on the sequences $\{\mathbf{x}_i\}$ generated by these T-functions, that is,

$$\mathbf{x}_{i+2^{j-1},j} = \mathbf{x}_{i,j} + \mathbf{x}_{i,j-1} + ajA_{i,2} + a(j-1) + b \bmod 2, 3 \leq j \leq n-1,$$

where $A_{i,2}$ is a sequence of period 4, a and b are constants determined by the coefficients c_i. This equation shows that the sequences generated by polynomial single cycle T-functions have potential secure problems.

Keywords: polynomial function, single cycle, pseudorandom generator.

1 Introduction

Polynomial functions are a class of important functions widely used in many branches of cryptography, such as block cipher and stream cipher. When a polynomial function is used as state transition function in a pseudorandom generator, it is usually required that the polynomial function generates a single cycle. Linear congruence pseudorandom generator is one of the most common pseudoramdom generators, it uses linear function as state transition function. However, the sequences generated by linear congruence generator have obvious lattice structure, thus it is not secure under cryptographical consideration. In order to avoid the lattice structure, it usually uses nonlinear function to replace linear function as state transform function, such as polynomial function with degree great than two or power function.

[⋆] This work was supported by the National Natural Science Foundation of China (Grant 60673081) and the National "863" Project(Grant 2006AA01Z417).

A function $f(\mathbf{x})$ from an n-bit input to an n-bit output with the property that the i-th bit of its outputs depends only on the first, the second, $\cdots$, and the i-th bit of its inputs is called a T-function (short for triangular function)[2]-[5]. The basic operations of T-functions are the following eight primitive operations: negation ($-\mathbf{x} \bmod 2^n$), addition ($\mathbf{x}+\mathbf{y} \bmod 2^n$), subtraction ($\mathbf{x}-\mathbf{y} \bmod 2^n$), multiplication ($\mathbf{x}\cdot\mathbf{y} \bmod 2^n$), complementation ($\neg\, \mathbf{x}$), or ($\mathbf{x}\vee\mathbf{y}$), and ($\mathbf{x}\wedge\mathbf{y}$), and xor ($\mathbf{x}\oplus\mathbf{y}$), where $\mathbf{x}$ and $\mathbf{y}$ are two n-bit words. Thus all the polynomial functions modulo 2^n are T-functions. Klimov and Shamir[2]-[5] presented several classes of single word and multiword single cycle T-functions. In 2005, Hong et.al[1] gave another multiword single cycle T-function, which can be used to construct the stream cipher TSC. One year later, Molland and Helleseth[7] proposed a linear relationship on the Klimov-Shamir T-function $f(\mathbf{x}) = \mathbf{x} + \mathbf{x}^2 \vee 5 \bmod 2^n$.

Among all the polynomial functions, permutation polynomials over $Z/(2^n)$ have been extensively studied, which belong to the category of invertible T-functions. In 2001, Rivest[8] provided a complete characterization of all the permutation polynomials modulo 2^n. Larin[6] then presented a sufficient and necessary condition on such polynomial function being a single cycle T-function.

In this paper, we give another proof on the sufficient and necessary condition that a polynomial function $f(\mathbf{x}) = c_0 + c_1\mathbf{x} + c_2\mathbf{x}^2 + \cdots + c_m\mathbf{x}^m$ with integer coefficients modulo $2^n (n \geq 3)$ is a single cycle T-function. That is, $f(\mathbf{x})$ generates a single cycle if and only if c_0, c_1 are odd, Δ_1, Δ_2 are even, $\Delta_1 + \Delta_2 + 2c_{1,0} \equiv 0 \bmod 4$, and $\Delta_1 + 2c_{2,0} + 2c_{1,1} \equiv 0 \bmod 4$, where $\Delta_1 = (c_2 + c_4 + \cdots)$, $\Delta_2 = (c_3 + c_5 + \cdots)$. Given $\mathbf{x}_0$, for $1 \leq i \leq 2^n - 1$, $\mathbf{x}_i$ is generated by using the recurrence $\mathbf{x}_i = f(\mathbf{x}_{i-1}) \bmod 2^n$. Let $\mathbf{x}_{i,j}$ be the j-th bit of $\mathbf{x}_i$, then we present a general linear equation of $\{\mathbf{x}_i\}$ generated by polynomial single cycle T-function, that is,

$$\mathbf{x}_{i+2^{j-1},j} = \mathbf{x}_{i,j} + \mathbf{x}_{i,j-1} + ajA_{i,2} + a(j-1) + b \bmod 2, 3 \leq j \leq n-1,$$

where $A_{i,2}$ is a sequence of period 4 and a, b are constants determined by the coefficients c_i. This equation shows that the sequences generated by polynomial single cycle T-functions have potential secure problems.

2 Sufficient and Necessary Condition of Polynomial Single Cycle T-Function

In this section, we present an exact characterization of the sufficient and necessary condition that the polynomial function $f(\mathbf{x}) = c_0 + c_1\mathbf{x} + c_2\mathbf{x}^2 + \cdots + c_m\mathbf{x}^m \bmod 2^n$ generates a single cycle. In the following, let $\Delta_1 = (c_2 + c_4 + \cdots)$, $\Delta_2 = (c_3 + c_5 + \cdots)$.

In 2001, Rivest provided the following complete characterization of all the permutation polynomials modulo 2^n:

Theorem 1. *[8] Let $f(\mathbf{x}) = c_0 + c_1\mathbf{x} + c_2\mathbf{x}^2 + \cdots + c_m\mathbf{x}^m$ be a polynomial with integer coefficients. Then $f(\mathbf{x})$ is a permutation polynomial modulo $2^n (n \geq 3)$ if and only if c_1 is odd, Δ_1 and Δ_2 are even.*

Corollary 1. *If $f(\mathbf{x}) = c_0+c_1\mathbf{x}+c_2\mathbf{x}^2+\cdots+c_m\mathbf{x}^m$ is a permutation polynomial, then*

$$\sum_{j=2}^{m} c_j j \equiv \sum_{j=2, j \text{ is odd}}^{m} c_j j \equiv 0 \bmod 2,$$
$$\sum_{j=2}^{m} c_j j \equiv 2(\Delta_2/2 + \sum_{j=2, j\equiv 2,3 \bmod 4}^{m} c_{j,0}) \bmod 4,$$
$$\sum_{j=2}^{m} c_j j(j-1) \equiv 2 \sum_{j=2, j\equiv 2,3 \bmod 4}^{m} c_{j,0} \bmod 4.$$

For simple representation, let

$$a = \Delta_2/2 + \sum_{j=2,\ j\equiv 2,3 \bmod 4}^{m} c_{j,0} \bmod 2,$$

thus $\sum_{j=2}^{m} c_j j \equiv 2a \bmod 4$. It is obvious that

Lemma 1. *If $n \geq 3, c_1 \equiv 1 \bmod 2$, then*

$$c_1^{2^{n-1}} \equiv 1 \bmod 2^{n+1},$$
$$c_1^{2^{n-2}} \equiv 1 + 2^n(c_{1,1} \oplus c_{1,2}) \bmod 2^{n+1}.$$

Lemma 2. *If $n \geq 2, c_0 \equiv 1 \bmod 2, c_1 \equiv 1 \bmod 2$, then*

$$\begin{aligned}&c_0(1 + c_1 + \cdots + c_1^{2^{n-1}-1})\\ &\equiv 2^{n-1}(c_{1,1} \oplus 1) + 2^n(c_{0,1} \oplus c_{0,1}c_{1,1} \oplus c_{1,1} \oplus c_{1,2}) \bmod 2^{n+1}.\end{aligned}$$

Proof. We proceed by induction on n. The case $n = 2$ is trivial. We assume that the statement is true for $n-1$, then for the case of n, we have

$$\begin{aligned}&c_0(1 + c_1 + \cdots + c_1^{2^{n-1}-1})\\ &= c_0(1 + c_1 + \cdots + c_1^{2^{n-2}-1} + c_1^{2^{n-2}}(1 + c_1 + \cdots + c_1^{2^{n-2}-1}))\\ &= c_0(1 + c_1 + \cdots + c_1^{2^{n-2}-1}) + c_0 \cdot c_1^{2^{n-2}}(1 + c_1 + \cdots + c_1^{2^{n-2}-1})\\ &\equiv c_0(1+c_1+\cdots + c_1^{2^{n-2}-1}) + c_0(1+2^n(c_{1,1} \oplus c_{1,2}))(1 + c_1 + \cdots + c_1^{2^{n-2}-1})(\text{Lemma 1})\\ &\equiv 2[c_0(1 + c_1 + \cdots + c_1^{2^{n-2}-1}) \bmod 2^{n-1}] \bmod 2^n\\ &= 2^{n-1}(c_{1,1} \oplus 1) + 2^n(c_{0,1} \oplus c_{0,1}c_{1,1} \oplus c_{1,1} \oplus c_{1,2}) \bmod 2^{n+1}\end{aligned}$$

So we have

Theorem 2. *Let $f(\mathbf{x}) = c_0 + c_1\mathbf{x} + c_2\mathbf{x}^2 + \cdots + c_m\mathbf{x}^m$ be a polynomial with integer coefficients, then $f(\mathbf{x})$ is single cycle T-function modulo $2^n (n \geq 3)$ if and only if c_0, c_1 are odd, Δ_1, Δ_2 are even, $\Delta_1 + \Delta_2 + 2c_{1,1} \equiv 0 \bmod 4$ and $\Delta_1 + 2c_{2,0} + 2c_{1,1} \equiv 0 \bmod 4$.*

Proof. It is clear that a single cycle T-function must be an invertible function. From Theorem 1, we know that a polynomial function is an invertible function if and only if c_1 is odd, Δ_1, Δ_2 are even.

1. When $n = 1$, it is easy to get $f(\mathbf{x}) \bmod 2$ generates a single cycle if and only if c_0, c_1 are odd, Δ_1, Δ_2 are even.

2. When $n = 2, 3$, first we have

$$\mathbf{x}_1 = f(\mathbf{x}_0) = c_0 + c_1\mathbf{x}_0 + \sum_{j=2}^{m} c_j \mathbf{x}_0^j,$$

$$\mathbf{x}_2 = c_0(1 + c_1) + c_1^2\mathbf{x}_0 + \sum_{j=2}^{m} c_j \mathbf{x}_1^j + c_1 \sum_{j=2}^{m} c_j \mathbf{x}_0^j,$$

$$\cdots$$

$$\mathbf{x}_{2^{n-1}} = c_0(1 + c_1 + \cdots + c_1^{2^{n-1}-1}) + c_1^{2^{n-1}}\mathbf{x}_0 + \sum_{i=0}^{2^{n-1}-1} (c_1^{2^{n-1}-1-i} \sum_{j=2}^{m} c_j \mathbf{x}_i^j). \quad (1)$$

From Lemma 1, Lemma 2 and Corollary 1, (1) can be represented as

$$\begin{aligned}
\mathbf{x}_{2^{n-1}} &\equiv 2^{n-1}(c_{1,1} \oplus 1) + \mathbf{x}_0 + \sum_{j=2}^{m} c_j \sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i} \mathbf{x}_i^j \\
&= 2^{n-1}(c_{1,1} \oplus 1) + \mathbf{x}_0 + \sum_{j=2}^{m} c_j \sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i} (\mathbf{x}_i \bmod 2^{n-1} + \mathbf{x}_{i,n-1}2^{n-1})^j \\
&\equiv 2^{n-1}(c_{1,1} \oplus 1) + \mathbf{x}_0 + \sum_{j=2}^{m} c_j \{ \sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i} (\mathbf{x}_i \bmod 2^{n-1})^j \\
&\quad + 2^{n-1} j \mathbf{x}_{i,n-1} (\mathbf{x}_i \bmod 2^{n-1})^{j-1} \} (n \geq 2) \\
&\equiv 2^{n-1}(c_{1,1} \oplus 1) + \mathbf{x}_0 + \sum_{j=2}^{m} c_j \sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i} (\mathbf{x}_i \bmod 2^{n-1})^j \bmod 2^n
\end{aligned}$$

2.1. If $n = 2$, $\mathbf{x}_2 = 2(c_{1,1} \oplus 1) + \mathbf{x}_0 + \Delta_1 + \Delta_2 \bmod 4$. It follows that $f(\mathbf{x}) \bmod 4$ generates a single cycle if and only if c_0, c_1 are odd, Δ_1, Δ_2 are even and $\Delta_1 + \Delta_2 + 2c_{1,1} \equiv 0 \bmod 4$.

2.2. If $n = 3$, from Lemma 1, $c_1^2 \equiv 1 \bmod 8$, then

$$\begin{aligned}
&\sum_{j=2}^{m} c_j (\sum_{i=0}^{3} c_1^{3-i} (\mathbf{x}_i \bmod 4)^j) \bmod 8 \\
&= \sum_{j=2}^{m} c_j \{ c_1 \cdot ((\mathbf{x}_0 \bmod 4)^j + (\mathbf{x}_2 \bmod 4)^j) + (\mathbf{x}_1 \bmod 4)^j + (\mathbf{x}_3 \bmod 4)^j \}. \quad (2)
\end{aligned}$$

If $\{\mathbf{x}_i \bmod 4\}$ generates a single cycle, then $\mathbf{x}_0 - \mathbf{x}_2 \equiv \mathbf{x}_1 - \mathbf{x}_3 \equiv 2 \bmod 4$. When $\{\mathbf{x}_0 \bmod 4, \mathbf{x}_2 \bmod 4\} = \{0, 2\}$ and $\{\mathbf{x}_1 \bmod 4, \mathbf{x}_3 \bmod 4\} = \{1, 3\}$, (2) can be represented as

$$4c_{2,0} + \sum_{j=2}^{m} c_j(1^j + 3^j) \equiv 4c_{2,0} + 2\Delta_1 + 4\Delta_2 = 2(2c_{2,0} + \Delta_1) \bmod 8 (\Delta_2 \text{ is even}).$$

It is easy to verify the above equation is also correct when $\{\mathbf{x}_0 \bmod 4, \mathbf{x}_2 \bmod 4\} = \{1,3\}$ and $\{\mathbf{x}_1 \bmod 4, \mathbf{x}_3 \bmod 4\} = \{0,2\}$. Thus $f(\mathbf{x}) \bmod 8$ generates a single cycle if and only if c_0, c_1 are odd, Δ_1, Δ_2 are even, $\Delta_1 + \Delta_2 + 2c_{1,1} \equiv 0 \bmod 4$ and $\Delta_1 + 2c_{1,1} + 2c_{2,0} \equiv 0 \bmod 4$.

3. When $n > 3$, from [5] we know that the sufficient and necessary condition on $f(\mathbf{x}) \bmod 2^n$ generating a single cycle is the same as that $f(\mathbf{x}) \bmod 8$ generating a single cycle.

3 Linear Equation on Polynomial Single Cycle T-Function

In this section, we concentrate on the algebraic structure of $\{\mathbf{x}_i\}$ generated by the following polynomial single cycle T-function

$$\begin{aligned} f(\mathbf{x}) = c_0 + c_1\mathbf{x} + c_2\mathbf{x}^2 + \cdots + c_m\mathbf{x}^m \bmod 2^n, c_0, c_1 \text{ are odd}, \Delta_1, \Delta_2 \text{ are even}, \\ \Delta_1 + \Delta_2 + 2c_{1,1} \equiv 0 \bmod 4 \text{ and } \Delta_1 + 2c_{2,0} + 2c_{1,1} \equiv 0 \bmod 4. \end{aligned} \tag{3}$$

The single cycle property of $f(\mathbf{x})$ naturally implies $\mathbf{x}_{i+2^k,k} = \mathbf{x}_{i,k} \oplus 1, 0 \le k \le n-1$. In this section, we show that $\{\mathbf{x}_i\}$ has another linear equation, i.e.,

$$\mathbf{x}_{i+2^{j-1},j} = \mathbf{x}_{i,j} + \mathbf{x}_{i,j-1} + ajA_{i,2} + a(j-1) + b \bmod 2, 3 \le j \le n-1.$$

where $\{A_{i,2}\}$ is a sequence of period 4 and a, b are constants determined by the coefficients c_i.

Lemma 3. *For $n \ge 2$, let $\{\mathbf{x}_i\}$ be the sequence generated by (3),*

$$A_{i,n} = \sum_{k=0}^{2^n-1} \mathbf{x}_{i+k,n}\mathbf{x}_{i+k,0} \bmod 2.$$

Then $A_{i,n} = A_{0,n} \oplus d_i$, where $\{d_i\}$ is the sequence of period 4 and $(d_0, d_1, d_2, d_3) = (0, \mathbf{x}_{0,0}, 1, \mathbf{x}_{0,0} \oplus 1)$.

Proof.

$$\begin{aligned} A_{i+1,n} &= \sum_{k=0}^{2^n-1} \mathbf{x}_{i+k+1,n}\mathbf{x}_{i+k+1,0} \bmod 2 \\ &= \sum_{k=1}^{2^n-1} \mathbf{x}_{i+k,n}\mathbf{x}_{i+k,0} + \mathbf{x}_{i+2^n,n}\mathbf{x}_{i+2^n,0} \bmod 2 \\ &= \sum_{k=0}^{2^n-1} \mathbf{x}_{i+k,n}\mathbf{x}_{i+k,0} + \mathbf{x}_{i,0}(\mathbf{x}_{i,n} + \mathbf{x}_{i+2^n,n}) \bmod 2 \end{aligned}$$

$$= \sum_{k=0}^{2^n-1} \mathbf{x}_{i+k,n}\mathbf{x}_{i+k,0} + \mathbf{x}_{i,0} \bmod 2$$
$$= A_{i,n} + \mathbf{x}_{i,0} \bmod 2$$

It follows that

$$A_{1,n} = A_{0,n} \oplus \mathbf{x}_{0,0}, A_{2,n} = A_{0,n} \oplus \mathbf{x}_{0,0} \oplus \mathbf{x}_{1,0} = A_{0,n} \oplus 1,$$
$$A_{3,n} = A_{0,n} \oplus \mathbf{x}_{0,0} \oplus 1, A_{4,n} = A_{0,n}.$$

In the research of algebraic structure of polynomial single cycle T-function, the following equation

$$B_n = \sum_{j=2}^{m} c_j (\sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i} (\mathbf{x}_i \bmod 2^{n-1})^j) \bmod 2^{n+1}$$

usually appears. Next we show the calculation process of B_3.

Lemma 4. *Let $\{\mathbf{x}_i\}$ be the sequence generated by (3),*

$$B_3 = \sum_{j=2}^{m} c_j (\sum_{i=0}^{3} c_1^{3-i} (\mathbf{x}_i \bmod 4)^j) \bmod 2^4.$$

Then we have

$$B_3 = 4c_{1,1} + 8\{[c_{2,0} + \Delta_1/2 \bmod 4]_1 + c_{2,1} + c_{3,0} + c_{1,1}c_{2,0} + a + c_{1,1}\mathbf{x}_{0,0}\} \bmod 2^4,$$

where $a = \Delta_2/2 + \sum_{j=2,\ j\equiv 2,3 \bmod 4}^{m} c_{j,0} \bmod 2$.

Proof. From Lemma 1, $c_1^2 \equiv 1 + 8(c_{1,1} \oplus c_{1,2}) \bmod 2^4$, then

$$\begin{aligned}
B_3 &= \sum_{j=2}^{m} c_j (\sum_{i=0}^{3} c_1^{3-i} (\mathbf{x}_i \bmod 4)^j) \bmod 2^4 \\
&\equiv \sum_{j=2}^{m} c_j c_1 \{(\mathbf{x}_0 \bmod 4)^j + (\mathbf{x}_2 \bmod 4)^j\} + \sum_{j=2}^{m} c_j \{(\mathbf{x}_1 \bmod 4)^j + (\mathbf{x}_3 \bmod 4)^j\} \\
&+ 8(c_{1,1} \oplus c_{1,2})(c_1 \sum_{j=2}^{m} c_j (\mathbf{x}_0 \bmod 4)^j + \sum_{j=2}^{m} c_j (\mathbf{x}_1 \bmod 4)^j) \bmod 2^4 \\
&\equiv \sum_{j=2}^{m} c_j c_1 \{(\mathbf{x}_0 \bmod 4)^j + (\mathbf{x}_2 \bmod 4)^j\} + \sum_{j=2}^{m} c_j \{(\mathbf{x}_1 \bmod 4)^j + (\mathbf{x}_3 \bmod 4)^j\} \bmod 2^4
\end{aligned}$$

It is easy to get

$$1^j + 3^j \bmod 16 \equiv \begin{cases} 2, & j \equiv 0 \bmod 4 \\ 4, & j \equiv 1 \bmod 4 \\ 10, & j \equiv 2 \bmod 4 \\ 12, & j \equiv 3 \bmod 4 \end{cases}.$$

1. When $\{\mathbf{x}_0 \bmod 4, \mathbf{x}_2 \bmod 4\} = \{0, 2\}$ and $\{\mathbf{x}_1 \bmod 4, \mathbf{x}_3 \bmod 4\} = \{1, 3\}$, then B_3 can be represented as

$$\begin{aligned}
B_3 &\equiv c_1(4c_2 + 8c_3) + \sum_{j=2}^{m} c_j(1^j + 3^j) \bmod 2^4 \\
&\equiv (1 + 2c_{1,1})(4c_2 + 8c_3) + 2 \sum_{j=2,\ j \text{ is even}}^{m} c_j + 4 \sum_{j=2,\ j \text{ is odd}}^{m} c_j + 8 \sum_{j=2, j\equiv 2,3 \bmod 4}^{m} c_{j,0} \\
&\equiv 4c_{2,0} + 8(c_{2,1} + c_{3,0} + c_{1,1}c_{2,0}) + 2\Delta_1 + 8a \bmod 2^4 \\
&\equiv 4([c_{2,0} + \Delta_1/2 \bmod 4]_0) \\
&+ 8\{[c_{2,0} + \Delta_1/2 \bmod 4]_1 + c_{2,1} + c_{3,0} + c_{1,1}c_{2,0} + a\} \bmod 2^4
\end{aligned}$$

Since $c_{2,0} + \Delta_1/2 \equiv c_{1,1} \bmod 2$, then

$$B_3 \equiv 4c_{1,1} + 8\{[c_{2,0} + \Delta_1/2 \bmod 4]_1 + c_{2,1} + c_{3,0} + c_{1,1}c_{2,0} + a\} \bmod 2^4.$$

2. When $\{\mathbf{x}_0 \bmod 4, \mathbf{x}_2 \bmod 4\} = \{1, 3\}$ and $\{\mathbf{x}_1 \bmod 4, \mathbf{x}_3 \bmod 4\} = \{0, 2\}$, similar to the analysis of case 1, we have

$$B_3 \equiv 4c_{1,1} + 8\{[c_{2,0} + \Delta_1/2 \bmod 4]_1 + c_{2,1} + c_{3,0} + c_{1,1}\Delta_1/2 + a\} \bmod 2^4.$$

Thus

$$B_3 \equiv 4c_{1,1} + 8\{[c_{2,0} + \Delta_1/2 \bmod 4]_1 + c_{2,1} + c_{3,0} + c_{1,1}c_{2,0} + a + c_{1,1}\mathbf{x}_{0,0}\} \bmod 2^4.$$

Lemma 5. *For $n \geq 4$, let $\{\mathbf{x}_i\}$ be the sequence generated by (3), suppose that*

$$D_n = \sum_{j=2}^{m} c_j j\Big(\sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i}\mathbf{x}_i^{j-1}\Big) \bmod 8,$$

then

$$D_n \equiv \begin{cases} 4\sum_{j=7, j\equiv 3 \bmod 4}^{m} c_{j,0}, & n = 4 \\ 0, & n \geq 5 \end{cases} \bmod 8. \tag{4}$$

Proof.

$$\begin{aligned}
D_n &= \sum_{j=2}^{m} c_j j\Big(\sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i}\mathbf{x}_i^{j-1}\Big) \bmod 8 \\
&\equiv \sum_{j=2}^{m} c_j j\Big(\sum_{i=0}^{2^{n-2}-1} (1 + 2c_{1,1} + 4c_{1,2})^{2^{n-2}-1-i} \cdot \mathbf{x}_i^{j-1}\Big) \bmod 8
\end{aligned}$$

Since

$$C_i^2 = \frac{i(i-1)}{2} \equiv \begin{cases} 0 \bmod 2, i \equiv 0, 1 \bmod 4 \\ 1 \bmod 2, i \equiv 2, 3 \bmod 4 \end{cases},$$

then

$$\begin{aligned}
&\sum_{i=0}^{2^{n-2}-1}(1+2c_{1,1}+4c_{1,2})^{2^{n-2}-1-i}\cdot\mathbf{x}_i^{j-1} \bmod 8\\
&=\sum_{i=0}^{2^{n-2}-1}(1+2c_{1,1}+4c_{1,2})^{i}\cdot(\mathbf{x}_{2^{n-2}-1-i})^{j-1} \bmod 8\\
&\equiv\sum_{i=0}^{2^{n-2}-1}(1+2c_{1,1}i+4c_{1,2}i)\cdot(\mathbf{x}_{2^{n-2}-1-i})^{j-1}\\
&+4c_{1,1}\sum_{i=0,i\equiv 2,3 \bmod 4}^{2^{n-2}-1}(\mathbf{x}_{2^{n-2}-1-i})^{j-1} \bmod 8\\
&\equiv\begin{cases}\sum_{i=0}^{2^{n-2}-1}(1+2c_{1,1}i+4c_{1,2}i)\cdot(\mathbf{x}_{2^{n-2}-1-i})^{j-1}+4c_{1,1}, & n=4\\ \sum_{i=0}^{2^{n-2}-1}(1+2c_{1,1}i+4c_{1,2}i)\cdot(\mathbf{x}_{2^{n-2}-1-i})^{j-1}, & n\geq 5\end{cases} \bmod 8
\end{aligned}$$

From Corollary 1, we have

$$4c_{1,1}\sum_{j=2}^{m}c_j j\equiv 0 \bmod 8,$$

$$4c_{1,2}\sum_{j=2}^{m}c_j j\sum_{i=0}^{2^{n-2}-1}i(\mathbf{x}_{2^{n-2}-1-i})^{j-1}\equiv 0 \bmod 8,$$

therefore

$$\begin{aligned}
D_n &\equiv\sum_{j=2}^{m}c_j j(\sum_{i=0}^{2^{n-2}-1}\mathbf{x}_i^{j-1})+2c_{1,1}\sum_{j=2}^{m}c_j j(\sum_{i=0}^{2^{n-2}-1}i\cdot\mathbf{x}_{2^{n-2}-i-1}^{j-1}) \bmod 8\\
&\hat{=} D_{n,1}+D_{n,2} \qquad (5)
\end{aligned}$$

We first calculate $D_{n,2}$, then $D_{n,1}$.

$$\begin{aligned}
D_{n,2}&=2c_{1,1}\{\sum_{j=2}^{m}c_j j(\sum_{i=0}^{2^{n-2}-1}i\cdot(\mathbf{x}_{2^{n-2}-i-1})^{j-1}) \bmod 4\} \bmod 8\\
&\equiv 2c_{1,1}\{\sum_{j=2}^{m}c_j j(\sum_{i=0}^{2^{n-2}-1}(i_0+2i_1)\cdot(\mathbf{x}_{2^{n-2}-i-1,0}+2(j-1)\mathbf{x}_{2^{n-2}-i-1,1}\mathbf{x}_{2^{n-2}-i-1,0}^{j-2}))\}\\
&\equiv 2c_{1,1}\{\sum_{j=2}^{m}c_j j(\sum_{i=0}^{2^{n-2}-1}(i_0+2i_1)\mathbf{x}_{2^{n-2}-i-1,0}) \bmod 4\} \bmod 8\\
&\equiv 0 \bmod 8(\sum_{j=2}^{m}c_j j\equiv 0 \bmod 2 \text{ and } \sum_{i=0}^{2^{n-2}-1}i_0\mathbf{x}_{2^{n-2}-i-1,0}\equiv 0 \bmod 2) \qquad (6)
\end{aligned}$$

$$\begin{aligned}
D_{n,1} &= \sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-2}-1} \mathbf{x}_i^{j-1}) \bmod 8 \\
&\equiv \sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-2}-1} (\mathbf{x}_{i,0} + 2\mathbf{x}_{i,1} + 4\mathbf{x}_{i,2})^{j-1}) \bmod 8 \\
&= \sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-2}-1} (\mathbf{x}_{i,0} + 2(j-1)\mathbf{x}_{i,1}\mathbf{x}_{i,0}^{j-2} + 4(j-1)\mathbf{x}_{i,2}\mathbf{x}_{i,0}^{j-2}) \\
&+ 4 \sum_{j=2, j\equiv 0,3 \bmod 4}^{m} c_j j \sum_{i=0}^{2^{n-2}-1} \mathbf{x}_{i,1}^2 \mathbf{x}_{i,0}^{j-3} \bmod 8
\end{aligned}$$

As $j(j-1) \equiv 0 \bmod 2$, then

$$\begin{aligned}
D_{n,1} \equiv& \sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-2}-1} \mathbf{x}_{i,0}) + 2(\sum_{j=2}^{m} c_j j(j-1)(\sum_{i=0}^{2^{n-2}-1} \mathbf{x}_{i,1}\mathbf{x}_{i,0}^{j-2}) \bmod 4) \\
&+ 4 \sum_{j=2, j\equiv 0,3 \bmod 4}^{m} c_j j \sum_{i=0}^{2^{n-2}-1} \mathbf{x}_{i,1}^2 \mathbf{x}_{i,0}^{j-3} \bmod 8 \\
&\equiv 2^{n-3} \sum_{j=2}^{m} c_j j + 2^{n-3} \sum_{j=2}^{m} c_j j(j-1) + 2^{n-2} c_{2,0} + 2^{n-2} \sum_{j=7, j\equiv 3 \bmod 4}^{m} c_{j,0} \bmod 8 \\
&\equiv \begin{cases} 4\sum_{j=7, j\equiv 3 \bmod 4}^{m} c_{j,0}, & n=4 \\ 0, & n \geq 5 \end{cases} \bmod 8 \qquad (7)
\end{aligned}$$

(By Corollary 1 and $\Delta_2/2 + c_{2,0} \equiv 0 \bmod 2$)

From (5), (6) and (7), we have

$$D_n = D_{n,1} \equiv \begin{cases} 4\sum_{j=7, j\equiv 3 \bmod 4}^{m} c_{j,0}, & n=4 \\ 0, & n \geq 5 \end{cases} \bmod 8.$$

Lemma 6. *For $n \geq 4$, let $\{\mathbf{x}_i\}$ be the sequence generated by (3), $A_{0,l}(2 \leq l \leq n-2)$ be defined as Lemma 3. Suppose that*

$$B_n = \sum_{j=2}^{m} c_j (\sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i} (\mathbf{x}_i \bmod 2^{n-1})^j) \bmod 2^{n+1},$$

then B_n can be represented as

$$\begin{aligned}
B_n = 2^{n-1} c_{1,1} + 2^n ([c_{2,0} + \Delta_1/2 \bmod 4]_1 + c_{2,1} + c_{1,1}c_{2,0} \\
+ \sum_{j=6, j\equiv 2 \bmod 4}^{m} c_j + a \sum_{l=2}^{n-2} A_{0,l} + c_{1,1}\mathbf{x}_{0,0}) \bmod 2^{n+1}.
\end{aligned}$$

Proof

$$\begin{aligned}
B_n &= \sum_{j=2}^{m} c_j \{ \sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i} (\mathbf{x}_i \bmod 2^{n-1})^j \} \bmod 2^{n+1} \\
&= \sum_{j=2}^{m} c_j \{ \sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-1}-1-i} (\mathbf{x}_i \bmod 2^{n-1})^j + \sum_{i=2^{n-2}}^{2^{n-1}-1} c_1^{2^{n-1}-1-i} (\mathbf{x}_i \bmod 2^{n-1})^j \} \\
&\equiv \sum_{j=2}^{m} c_j \{ \sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i} ((\mathbf{x}_i \bmod 2^{n-1})^j + (\mathbf{x}_i + 2^{n-2} \bmod 2^{n-1})^j) \} \\
&\quad + 2^n \sum_{j=2}^{m} c_j \{ \sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i} (c_{1,1} \oplus c_{1,2}) (\mathbf{x}_i \bmod 2^{n-1})^j \} \bmod 2^{n+1} (\text{Lemma 1})
\end{aligned}$$

It is obvious that the second term in the above equation is zero, then

$$\begin{aligned}
B_n &\equiv \sum_{j=2}^{m} c_j \{ \sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i} ((\mathbf{x}_i \bmod 2^{n-1})^j + (\mathbf{x}_i + 2^{n-2} \bmod 2^{n-1})^j) \} \\
&\equiv \sum_{j=2}^{m} c_j \{ \sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i} (2(\mathbf{x}_i \bmod 2^{n-1})^j + j 2^{n-2} (\mathbf{x}_i \bmod 2^{n-1})^{j-1}) \} \\
&\equiv 2 \sum_{j=2}^{m} c_j \{ \sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i} (\mathbf{x}_i \bmod 2^{n-1})^j \} \\
&\quad + 2^{n-2} \sum_{j=2}^{m} c_j j \{ \sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i} (\mathbf{x}_i \bmod 2^{n-1})^{j-1} \} \bmod 2^{n+1} \\
&\hat{=} B_{n,1} + B_{n,2} \bmod 2^{n+1} \qquad (8)
\end{aligned}$$

From Lemma 5, we have

$$B_{n,2} \equiv \begin{cases} 2^n \sum_{j=7, j \equiv 3 \bmod 4}^{m} c_{j,0}, & n = 4 \\ 0, & n \geq 5 \end{cases} \bmod 2^{n+1} . \qquad (9)$$

$$\begin{aligned}
B_{n,1} &= 2 \sum_{j=2}^{m} c_j (\sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i} (\mathbf{x}_i \bmod 2^{n-1})^j) \bmod 2^{n+1} \\
&= 2 \sum_{j=2}^{m} c_j (\sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i} (\mathbf{x}_i \bmod 2^{n-2} + \mathbf{x}_{i,n-2} 2^{n-2})^j) \\
&= 2 \sum_{j=2}^{m} c_j (\sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i} ((\mathbf{x}_i \bmod 2^{n-2})^j + j 2^{n-2} \mathbf{x}_{i,n-2} (\mathbf{x}_i \bmod 2^{n-2})^{j-1}))
\end{aligned}$$

$$= 2\sum_{j=2}^{m} c_j(\sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i}(\mathbf{x}_i \bmod 2^{n-2})^j)$$

$$+ 2^{n-1}\sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i}\mathbf{x}_{i,n-2}(\mathbf{x}_i \bmod 2^{n-2})^{j-1}) \bmod 2^{n+1}(n \geq 4)$$

Since

$$\sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i}\mathbf{x}_{i,n-2}(\mathbf{x}_i \bmod 2^{n-2})^{j-1}) \bmod 4$$

$$\equiv \sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i}\mathbf{x}_{i,n-2}(\mathbf{x}_{i,0} + 2(j-1)\mathbf{x}_{i,0}^{j-2}\mathbf{x}_{i,1})) \bmod 4$$

$$\equiv \sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-2}-1} \mathbf{x}_{i,n-2}\mathbf{x}_{i,0}) \bmod 4(j(j-1) \equiv 0 \bmod 2)$$

$$\equiv aA_{0,n-2} \bmod 4(\sum_{j=2}^{m} c_j j \equiv 0 \bmod 2)$$

then

$$B_{n,1} = 2\sum_{j=2}^{m} c_j(\sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i}(\mathbf{x}_i \bmod 2^{n-2})^j) + 2^n aA_{0,n-2} \bmod 2^{n+1}. \quad (10)$$

From (9) and (10), we have

$$B_n = \sum_{j=2}^{m} c_j(\sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i}(\mathbf{x}_i \bmod 2^{n-1})^j \bmod 2^{n+1})$$

$$\equiv 2\sum_{j=2}^{m} c_j(\sum_{i=0}^{2^{n-2}-1} c_1^{2^{n-2}-1-i}(\mathbf{x}_i \bmod 2^{n-2})^j) + 2^n aA_{0,n-2} \bmod 2^{n+1}$$

$$\equiv \cdots$$

$$\equiv 2^{n-4}\sum_{j=2}^{m} c_j(\sum_{i=0}^{2^3-1} c_1^{2^3-1-i}(\mathbf{x}_i \bmod 2^3)^j) + 2^n a\sum_{l=3}^{n-2} A_{0,l} \bmod 2^{n+1}$$

$$\equiv 2^{n-3}\sum_{j=2}^{m} c_j(\sum_{i=0}^{2^2-1} c_1^{2^2-1-i}\mathbf{x}_i{}^j) + 2^n \sum_{j=7, j\equiv 3 \bmod 4}^{m} c_{j,0} + 2^n \sum_{l=2}^{n-2} aA_{0,l} \bmod 2^{n+1}$$

$$\equiv 2^{n-1}c_{1,1} + 2^n(c_{2,1} + c_{1,1}c_{2,0} + [c_{2,0} + \Delta_1/2 \bmod 4]_1$$

$$+ \sum_{j=6, j\equiv 2 \bmod 4}^{m} c_j + a\sum_{l=2}^{n-2} A_{0,l} + c_{1,1}\mathbf{x}_{0,0}) \bmod 2^{n+1}(\text{Lemma 4 and Lemma 5})$$

Theorem 3. *For $n \geq 5$, let $\{\mathbf{x}_i\}$ be the sequence generated by (3). Suppose that*

$$b = c_{0,1} + c_{0,1}c_{1,1} + c_{1,1} + c_{1,2} + c_{2,1} + c_{1,1}c_{2,0} + [c_{2,0} + \Delta_1/2 \bmod 4]_1 + \sum_{j=6, j\equiv 2 \bmod 4}^{m} c_{j,0} \bmod 2.$$

Then

$$\mathbf{x}_{i+2^{j-1},j} = \mathbf{x}_{i,j} + \mathbf{x}_{i,j-1} + c_{1,1}\mathbf{x}_{i,0} + a\sum_{l=2}^{j-1} A_{i,l} + b \bmod 2, 3 \leq j \leq n-1,$$

where $a = \Delta_2/2 + \sum_{j=2,j\equiv 2,3 \bmod 4}^{m} c_{j,0} \bmod 2$, $A_{i,l}$ is defined as Lemma 3.

Proof. From (1), Lemma 1 and Lemma 2, we have

$$\mathbf{x}_{2^{n-1}} = 2^{n-1}(c_{1,1} \oplus 1) + 2^n(c_{0,1} \oplus c_{0,1}c_{1,1} \oplus c_{1,1} \oplus c_{1,2}) + \mathbf{x}_0 + \sum_{j=2}^{m} c_j(\sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i}\mathbf{x}_i^j) \bmod 2^{n+1}. \quad (11)$$

Since

$$\sum_{j=2}^{m} c_j(\sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i}\mathbf{x}_i^j) \bmod 2^{n+1}$$
$$\equiv \sum_{j=2}^{m} c_j\{\sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i}(\mathbf{x}_i \bmod 2^{n-1})^j + j\mathbf{x}_{i,n-1}2^{n-1}(\mathbf{x}_i \bmod 2^{n-1})^{j-1}$$
$$+ j\mathbf{x}_{i,n}2^n(\mathbf{x}_i \bmod 2^{n-1})^{j-1}\} \bmod 2^{n+1} (n \geq 5)$$

It is clear that the above equation can be divided into three parts, next we calculate them seperately.

1. As $\sum_{j=2}^{m} c_j j \equiv 0 \bmod 2$(Corollary 1), then

$$2^n\{\sum_{j=2}^{m} c_j j \sum_{i=0}^{2^{n-1}-1} c_{1,0}(\mathbf{x}_{i,n}\mathbf{x}_{i,0}) \bmod 2\} = 0 \bmod 2^{n+1}. \quad (12)$$

2.

$$2^{n-1}\{\sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i}\mathbf{x}_{i,n-1}(\mathbf{x}_i \bmod 2^{n-1})^{j-1}) \bmod 4\} \bmod 2^{n+1}$$
$$\equiv 2^{n-1}\{\sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i}\mathbf{x}_{i,n-1}(\mathbf{x}_{i,0} + 2\mathbf{x}_{i,1})^{j-1}) \bmod 4\} \bmod 2^{n+1}$$
$$= 2^{n-1}\{\sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i}\mathbf{x}_{i,n-1}(\mathbf{x}_{i,0} + 2\mathbf{x}_{i,1})^{j-1}) \bmod 4\} \bmod 2^{n+1}$$

$$= 2^{n-1}\{\sum_{j=3}^{m} c_j j(\sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i}(\mathbf{x}_{i,n-1}\mathbf{x}_{i,0} + 2(j-1)\mathbf{x}_{i,0}^{j-2}\mathbf{x}_{i,1}\mathbf{x}_{i,n-1})) \bmod 4\} \bmod 2^{n+1}$$

$$= 2^{n-1}\{\sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-1}-1} c_1^{2^{n-1}-1-i}\mathbf{x}_{i,n-1}\mathbf{x}_{i,0}) \bmod 4\} \bmod 2^{n+1}$$

$$\equiv 2^{n-1}(\sum_{j=2}^{m} c_j j(\sum_{i=0}^{2^{n-1}-1} \mathbf{x}_{i,n-1}\mathbf{x}_{i,0} \bmod 2) \bmod 4) \bmod 2^{n+1} \ (\sum_{j=2}^{m} c_j j \equiv 0 \bmod 2)$$

$$\equiv 2^n a A_{0,n-1} \bmod 2^{n+1} (\text{Lemma } 4) \tag{13}$$

From Lemma 6, (12) and (13), we have

$$\begin{aligned}
\mathbf{x}_{2^{n-1}} &= (\mathbf{x}_0 \bmod 2^{n-1}) + 2^{n-1}(\mathbf{x}_{0,n-1} + (c_{1,1} \oplus 1) + c_{1,1}) \\
&+ 2^n\{\mathbf{x}_{0,n} + c_{0,1} + c_{0,1}c_{1,1} + c_{1,1} + c_{1,2} + c_{2,1} + c_{1,1}c_{2,0} \\
&+ [c_{2,0} + \Delta_1/2 \bmod 4]_1 + \sum_{j=6, j\equiv 2 \bmod 4}^{m} c_{j,0} + a\sum_{l=2}^{n-1} A_{0,l} + c_{1,1}\mathbf{x}_{0,0}\} \bmod 2^{n+1} \\
&\equiv (\mathbf{x}_0 \bmod 2^{n-1}) + 2^{n-1}(\mathbf{x}_{0,n-1} \oplus 1) + 2^n\{\mathbf{x}_{0,n} + \mathbf{x}_{0,n-1} \\
&+ c_{0,1} + c_{0,1}c_{1,1} + c_{1,1} + c_{1,2} + c_{2,1} + c_{1,1}c_{2,0} \\
&+ a\sum_{l=2}^{n-1} A_{0,l} + [c_{2,0} + \Delta_1/2 \bmod 4]_1 + \sum_{j=6, j\equiv 2 \bmod 4}^{m} c_{j,0} + c_{1,1}\mathbf{x}_{0,0}\} \\
&\equiv (\mathbf{x}_0 \bmod 2^{n-1}) + 2^{n-1}(\mathbf{x}_{0,n-1} \oplus 1) + 2^n\{\mathbf{x}_{0,n} + \mathbf{x}_{0,n-1} \\
&+ a\sum_{l=2}^{n-1} A_{0,l} + c_{0,1} + c_{0,1}c_{1,1} + c_{1,1} + c_{1,2} + c_{2,1} + c_{1,1}c_{2,0} \\
&+ [c_{2,0} + \Delta_1/2 \bmod 4]_1 + \sum_{j=6, j\equiv 2 \bmod 4}^{m} c_{j,0} + c_{1,1}\mathbf{x}_{0,0}\}
\end{aligned}$$

From the definitions of a and b,

$$\mathbf{x}_{2^{n-1},n} = \mathbf{x}_{0,n} + \mathbf{x}_{0,n-1} + c_{1,1}\mathbf{x}_{0,0} + a\sum_{l=2}^{n-1} A_{0,l} + b \bmod 2,$$

As the above equation is correct for all $\mathbf{x}_0$, it is also correct for the sequence shift i positions, that is,

$$\mathbf{x}_{i+2^{n-1},n} = \mathbf{x}_{i,n} + \mathbf{x}_{i,n-1} + c_{1,1}\mathbf{x}_{i,0} + a\sum_{l=2}^{n-1} A_{i,l} + b \bmod 2.$$

As $\mathbf{x}_i \bmod 2^n$ contains all the subsequences $\mathbf{x}_i \bmod 2^j (1 \le j \le n-1)$, then the above equation is correct for all $j(4 \le j \le n-1)$, that is

$$\mathbf{x}_{i+2^{j-1},j} = \mathbf{x}_{i,j} + \mathbf{x}_{i,j-1} + c_{1,1}\mathbf{x}_{i,0} + a\sum_{l=2}^{j-1} A_{i,l} + b \bmod 2, 3 \le j \le n-1. \tag{14}$$

Theorem 4. *Let $n \geq 5$, $\{\mathbf{x}_i\}$ be the sequence generated by (3), a, b are defined as Theorem 3 and $A_{i,l}$ is defined in Lemma 3, then we have*

$$\mathbf{x}_{i+2^{j-1},j} = \mathbf{x}_{i,j} + \mathbf{x}_{i,j-1} + ajA_{i,2} + a(j-1) + b \bmod 2, 3 \leq j \leq n-1.$$

Proof

$$\begin{aligned}
A_{i,j} &= \sum_{k=0}^{2^j-1} \mathbf{x}_{i+k,j}\mathbf{x}_{i+k,0} \bmod 2 \\
&= \sum_{k=0}^{2^{j-1}-1} \mathbf{x}_{i+k,0}(\mathbf{x}_{i+k,j} + \mathbf{x}_{i+k+2^{j-1},j}) \bmod 2 \\
&= \sum_{k=0}^{2^{j-1}-1} \mathbf{x}_{i+k,0}(\mathbf{x}_{i+k,j-1} + a\sum_{l=2}^{j-1} A_{i+k,l} + b + c_{1,1}\mathbf{x}_{i+k,0}) \bmod 2 \\
&= A_{i,j-1} + a \sum_{k=0}^{2^{j-1}-1} \mathbf{x}_{i+k,0} \sum_{l=2}^{j-1} A_{i+k,l} \bmod 2
\end{aligned}$$

From Lemma 3, the period of $A_{i+k,l}$ is 4, thus the period of $\mathbf{x}_{i+k,0}A_{i+k,l}$ is also 4. Thus

$$A_{i,j} = \begin{cases} A_{i,j-1}, & j \geq 4 \\ A_{i,2} + a(j-2), & j = 3 \end{cases} \bmod 2,$$

then

$$A_{i,j} = A_{i,j-1} = \cdots = A_{i,3} = A_{i,2} + a(j-2) \bmod 2. \tag{15}$$

From (14) and (15), we have

$$\mathbf{x}_{i+2^{j-1},j} = \mathbf{x}_{i,j} + \mathbf{x}_{i,j-1} + ajA_{i,2} + a(j-1) + b \bmod 2, 3 \leq j \leq n-1. \tag{16}$$

Remark 1. If $aj \equiv 0 \bmod 2$, then (16) can be changed into $\mathbf{x}_{i+2^{j-1},j} = \mathbf{x}_{i,j} + \mathbf{x}_{i,j-1} + a + b$. If $aj \equiv 1 \bmod 2$, then (16) can be simplified as $\mathbf{x}_{i+2^{j-1},j} = \mathbf{x}_{i,j} + \mathbf{x}_{i,j-1} + A_{i,2} + a + b + 1$, which is connected with the sequence $\{A_{i,2}\}$ of period 4. By this means, the algebraic structure of polynomial single cycle T-function is simple.

4 Cryptography Applications of This Linear Equation

The linear equation (16) may be combined with other cryptanalysis techniques and give a powerful attacks. From (16), we can easily get

$$\begin{aligned}
&P(\mathbf{x}_{i+2^{j-1},j} = \mathbf{x}_{i,j}) \\
&= P(\mathbf{x}_{i,j-1} + ajA_{i,2} + a(j-1) + b = 0) \\
&= P(\mathbf{x}_{i,j-1} = 0)P(ajA_{i,2} + a(j-1) + b = 0 | \mathbf{x}_{i,j-1} = 0) \\
&+ P(\mathbf{x}_{i,j-1} = 1)P(ajA_{i,2} + a(j-1) + b = 1 | \mathbf{x}_{i,j-1} = 1) \\
&= \frac{1}{2}(P(ajA_{i,2} + a(j-1) + b = 0) + P(ajA_{i,2} + a(j-1) + b = 1)) = \frac{1}{2}
\end{aligned}$$

Assume that we find a condition K_i which gives the correlation $P(\mathbf{x}_{i+2^{j-1},j} = \mathbf{x}_{i,j}) \neq 0.5$, where $\mathbf{x}_{i,j}$ is known, $\mathbf{x}_{i+2^{j-1},j}$ is unknown. Next, assume we know a, b and $A_{i,2}$. Then $\mathbf{x}_{i+2^{j-1},j}$ can be guessed since there are only four possible choices of $\mathbf{x}_{i,j}$ and $\mathbf{x}_{i+2^{j-1},j}$. From (16) we have $P(\mathbf{x}_{i,j} = \mathbf{x}_{i,j-1}|K_i) \neq 0.5$. Thus, the linear equation (16) combined with K_i leaks information about the keys.

5 Conclusions

We found a general linear equation over $GF(2)$ that always holds for all sequences generated by polynomial single cycle T-functions. This linear relation shows that these T-functions have strong algebraic structure, which can be used in cryptanalysis. Further, the equation may be used as a basic tool for analyzing some pseudoramdon generator using polynomial functions.

Acknowledgments. The authors would like to express their deep gratitude to the anonymous referees for their valuable comments on this paper.

References

1. Hong, J., Lee, D.H., Yeom, Y., Han, D.: A New Class of Single Cycle T-Functions. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 68–82. Springer, Heidelberg (2005)
2. Klimov, A., Shamir, A.: A New Class of Invertible Mappings. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 470–483. Springer, Heidelberg (2003)
3. Klimov, A., Shamir, A.: Cryptographic Applications of T-Functions. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 248–261. Springer, Heidelberg (2004)
4. Klimov, A., Shamir, A.: New Cryptographic Primitives Based on Multiword T-Functions. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 1–15. Springer, Heidelberg (2004)
5. Klimov, A.: Application of T-functions in Cryptography, PhD Thesis, Weizmann Institute of Science (2005)
6. Larin, M.V.: Transitive Polynomial Transformations of Residue Class Rings. Discrete Mathematics and Applications, 141–154 (February 2002)
7. Molland, H., Helleseth, T.: Linear properties in T-functions. IEEE Trans. Inform. Theory, 5151–5157 (November 2006)
8. Rivest, R.: Permutation Polynomials Modulo 2^{ω}. Finite Fields and their Applications, 287–292 (September 2001)
9. Wang, J.S., Qi, W.F.: Trace Presentation of Bent Sequence Families. Journal of Communications(China), 8–13 (January 2006)
10. Wang, J.S., Qi, W.F.: Analysis of Design Interleaved ZCZ Sequence Family. In: Gong, G., Helleseth, T., Song, H.-Y., Yang, K. (eds.) SETA 2006. LNCS, vol. 4086, pp. 129–140. Springer, Heidelberg (2006)
11. Wang, J.S., Qi, W.F.: A Class of Binary ZCZ Sequence Families Constructed by Extending Period Twice. Journal of Electronics(China), 301–304 (May 2007)
12. Zhang, W.Y., Wu, C.K.: The Algebraic Normal Form, Linear Complexity and k-Error Linear Complexity of Single-Cycle T-Function. In: Gong, G., Helleseth, T., Song, H.-Y., Yang, K. (eds.) SETA 2006. LNCS, vol. 4086, pp. 391–401. Springer, Heidelberg (2006)

Weight Support Technique and the Symmetric Boolean Functions with Maximum Algebraic Immunity on Even Number of Variables[⋆]

Longjiang Qu[1] and Chao Li[1,2]

[1] Department of Mathematic and System Science, National University of Defense Technology, ChangSha, 410073, China
[2] Key Lab of Network Security and Cryptology, FuJian Normal University, FuZhou, China
ljqu_happy@hotmail.com, lichao_nudt@sina.com

Abstract. The weight support technique is applied to study the symmetric Boolean functions with maximum algebraic immunity on even number of variables. The problem to study the n-variable(n even) symmetric Boolean functions with maximum algebraic immunity is reduced to the problem to determine $WS_{min}(n, \frac{n}{2})$. Then some new results about $WS_{min}(n, \frac{n}{2})$ are got. A fast algorithm to get all the n-variable(n even) symmetric Boolean functions with maximum algebraic immunity is also given.

Keywords: Algebraic Attack, Algebraic Immunity, Symmetric Boolean Function, Weight Support.

1 Introduction

In recent years algebraic attacks([1,2,7,8,9]) have become an important tool in cryptanalysis of stream and block cipher systems. A new cryptographic property for designing Boolean functions to resist this kind of attacks, called algebraic immunity, has been introduced([10,15]). Since then several classes of Boolean functions with large algebraic immunity have been investigated and constructed in order to against the algebraic attack([6,10,11,13,15]) and the symmetric Boolean functions have been paid particular attention([3,12,14,16,17]).

Let B_n be the ring of Boolean functions with n variables $x_1, x_2, \cdots, x_n$. Let SB_n be the ring of symmetric Boolean functions with n variables $x_1, x_2, \cdots, x_n$. For $f \in B_n$, the annihilators of f is the set

$$Ann(f) = \{g \in B_n | f \cdot g = 0\}$$

$Ann(f)$ is the ideal of the ring B_n generated by $1 + f$: $Ann(f) = (1 + f) = (1 + f)B_n$([15],Theorem1).

⋆ This work is supported by the National Natural Science Foundation of China(No. 60573028), the Open Funds of Key Lab of FuJian Province University Network Security and Cryptology(No. 07A003) and the Basic Research Foundation of National University of Defense Technology(No. JC07-02-03).

Definition 1. *For $f \in B_n$, the algebraic immunity(AI) of f is the minimum degree of non-zero functions $g \in B_n$ such that $gf = 0$ or $g(f+1) = 0$. Namely,*

$$AI(f) = min\{\deg(g)|0 \neq g \in Ann(f) \cup Ann(1+f) = (f+1) \cup (f)\}$$

A symmetric Boolean function $f \in SB_n$ can be expressed by a vector

$$v_f = (v_f(0), \cdots, v_f(n)) \in F_2^{n+1}$$

where $v_f(i) = f(x)$ for $x \in F_2^n$ with Hamming weight $wt(x) = i$. On the other hand, f can also be expressed as

$$f(x_1, \cdots, x_n) = \sum_{i=0}^{n} \lambda_f(i)\sigma_i^n$$

where $\lambda_f(i) \in F_2$ and σ_i^n is the i-th elementary symmetric function of $x_1, x_2, \cdots, x_n$. The relation between v_f and $\lambda_f = (\lambda_f(0), \cdots, \lambda_f(n))$ is:

Lemma 1. *[4]*

$$v_f(i) = \sum_{k \preceq i} \lambda_f(k), \lambda_f(i) = \sum_{k \preceq i} v_f(k) \, \textit{for any } i \in \{0, 1, \cdots, n\}$$

where, for the 2-adic expressions

$$k = \sum_{j=0}^{l} k_j 2^j, i = \sum_{j=0}^{l} i_j 2^j (k_j, i_j \in \{0, 1\})$$

$k \preceq i$ means that for any integer $j(0 \leq j \leq l)$, $k_j \leq i_j$.

Lemma 1 can be derived directly from the Lucas formula

$$\binom{i}{k} \equiv \binom{i_0}{k_0}\binom{i_1}{k_1} \cdots \binom{i_l}{k_l} (mod 2) = \begin{cases} 1, \text{ if } k \preceq i \\ 0, \text{ otherwise} \end{cases}$$

Note that it is easy to see that if we generalize the binomial coefficient $\binom{i}{k} = 0$ for $i < k$, then the Lucas formula is also correct, and $v_{\sigma_k^n}(i) = \binom{i}{k}$ also holds.

When $n = 2k + 1 \geq 3$ is odd, it was proved in [16] that there are only two symmetric Boolean functions f and $1+f$ with maximal $AI(= k+1)$ where $v_f = (\underbrace{1, 1, ..., 1}_{k+1}, \underbrace{0, 0, ..., 0}_{k+1})$.

When n is even, many n-variable symmetric Boolean functions with maximum AI were found ([3,12,14]) and several necessary conditions for a symmetric Boolean function to arrive maximum AI were given([3,17]). For $n = 2^m$, all n-variable symmetric Boolean functions with maximum AI were got by the following theorem in [14].

Theorem 1. *[14] Suppose that $n = 2^m (m \geq 2)$ and $f \in SB_n$. Then $AI(f) = \frac{n}{2}$ if and only if the following two conditions are both satisfied:*

(1) $v_f(2^{m-1} - 2^t + k_t) = v_f(2^{m-1} + k_t) + 1$ for all $1 \leq k_t \leq 2^t - 1$.$(1 \leq t \leq m-1)$
(2) $(v_f(0), v_f(2^{m-1}), v_f(2^m)) \notin \{(0,0,0), (1,1,1)\}$

Given an n-variable Boolean function f and a positive integer r, we denote by $R_f(r, n)$ the restriction of the generator matrix of the rth-order Reed-Muller code to the support of f. Clearly, an n-variable Boolean function f has no annihilator of algebraic degree at most k if and only if all the matrices $R_f(r, n), r \leq k$, are of full rank. Then studying the AI of f can be reduced to studying the ranks of $R_f(r, n)$ and of $R_{1+f}(r, n)$. Based on this idea, it was shown in [5] that for a random balanced function, it should have: for even n, AI is almost always equal to $\frac{n}{2}$; for odd n, AI is almost always greater than or equal to $\frac{n-1}{2}$. Also based on this idea, a novel method was presented to construct and count of the Boolean functions with maximum AI in [13](also in [18]). As the structure of $R_f(r, n)$ is special when f is a symmetric Boolean function, several constructions of symmetric Boolean functions with maximum AI were given in [12] by using some combinational results. In theory, one can construct all Boolean functions with maximum AI by analyzing the ranks of $R_f(r, n)$ and of $R_{1+f}(r, n)$, certainly including all symmetric such functions. However, even for the function f of Theorem 1 whose structure is very special, one can't analysis the behavior of the ranks of $R_f(r, n)$ and of $R_{1+f}(r, n)$ by using the existing mathematic theory as far as the author knows. This is the reason why the sufficient condition in Theorem 1 was only presented as a conjecture in [17].

We need a new method, which is the "weight support" technique presented in [14]. In [14] the relation between the AI of a Boolean function and its weight supports was studied. Based on this relation, a sufficient and necessary condition(Theorem 1) for a 2^m-variable symmetric Boolean function to have maximum AI was presented. Then all such functions were got. This result shows that the coefficient matrixes of the equation systems constructed from the annihilators of these functions are full rank, which is not a easy task even in the combinational theory. Motivated by their work, we apply the weight support technique to study the symmetric Boolean functions with maximum AI and get some new results.

The paper is organized as follows: the weight support technique and some basic symbols are introduced in the following section; some new results and their proofs are given in Section 3; the conclusion is given in Section 4.

2 Weight Support Technique

Definition 2. *[14] For $f \in B_n$, the weight support of f is defined by*

$$WS(f) = \{i \in N | \exists \ a \in F_2^n \text{ , such that } wt(a) = i \text{ and } f(a) = 1\}$$

where $wt(a) = \#\{l \mid 1 \leq l \leq n, a_l = 1\}$ represents the Hamming weight of $a = (a_1, a_2, \cdots, a_n) \in F_2^n$. It's easy to see that for $f \in SB_n$, $WS(f) = \{i \in N | v_f(i) = 1\}$.

In [14], a partial order of B_n is defined by: for $f, g \in B_n$, $f \leq g$ if and only if $WS(f) \subseteq WS(g)$. So for $f \in SB_n$ and $g \in B_n$, we have

$$\begin{aligned} fg = 0 &\Leftrightarrow \text{ If } \exists\, a \in F_2^n \text{ such that } wt(a) = i \text{ and } g(a) = 1, \text{then } v_f(i) = 0 \\ &\Leftrightarrow WS(g) \subseteq \overline{WS(f)} \end{aligned}$$

where $\overline{WS(f)} = N \setminus WS(f)$ represents the complementary set of $WS(f)$ in N and $N = \{0, 1, 2, \cdots, n\}$. Similarly,

$$(f+1)g = 0 \Leftrightarrow WS(g) \subseteq WS(f)$$

then it can be concluded that:

Lemma 2. *[14] Suppose that $n \geq 2$ and $1 \leq d \leq \lceil \frac{n}{2} \rceil$. For $f \in SB_n$, $AI(f) \geq d$ if and only if for any $g \in B_n$ such that $0 \leq \deg(g) \leq d-1$, we have $WS(g) \nsubseteq WS(f)$ and $WS(g) \nsubseteq \overline{WS(f)}$.*

For each $l \geq 1$, let

$$p_l = p_l(x_1, x_2, \cdots, x_{2l}) = (x_1+x_2)(x_3+x_4)\cdots(x_{2l-1}+x_{2l}) \in B_{2l}$$

Then $WS(p_l) = \{l\}$.

Lemma 3. *[14] Suppose that $n \geq 2$ and $f \in SB_n$. If there exists $0 \neq g \in B_n$ such that $fg = 0$, then there exists an integer l, $0 \leq l \leq \lfloor \frac{n}{2} \rfloor$ and $0 \neq h = h(x_{2l+1}, x_{2l+2}, \cdots, x_n) \in SB_{n-2l}$, $\deg(h) \leq \deg(g) - l$ such that $fhp_l = 0$.*

Let

$$S(n,d) = \left\{ f_{n,b} = h_{n-2b}p_b \left| \begin{array}{c} h_{n-2b} = h_{n-2b}(x_{2b+1}, x_{2b+2}, \cdots, x_n) \in SB_{n-2b}, \\ 0 \leq \deg(h_{n-2b}) < d-b, \\ 0 \leq b \leq d-1 \end{array} \right. \right\}$$

Then by Lemma 2 and Lemma 3 we have

Lemma 4. *[14] Suppose that $1 \leq d \leq \lceil \frac{n}{2} \rceil$ and $f \in SB_n$. Then $AI(f) \geq d$ if and only if for any $g \in S(n,d)$,*

$$WS(g) \nsubseteq WS(f) \text{ and } WS(g) \nsubseteq \overline{WS(f)}$$

Furthermore, let $S_{min}(n,d)$ be the set of minimum elements of the partial order set $(S(n,d), \preceq)$. Then Lemma 4 is also true if $S(n,d)$ is replaced by $S_{min}(n,d)$. If the set $S_{min}(n,d)$ has a simple structure, then we can get a nice characterization of all $f \in SB_n$ satisfying $AI(f) \geq d$. So the following open problem was raised in [14]:

Problem 1. [14] For $1 \leq d \leq \lceil \frac{n}{2} \rceil$, to determine the set $S_{min}(n,d)$.

Let n be an even integer, $d = \frac{n}{2}$ and

$$S(n, \frac{n}{2}) = \{h_b p_b | h_b = h_b(x_{2b+1}, x_{2b+2}, \cdots, x_n) \in SB_{n-2b}, 0 \leq \deg(h_b) < \frac{n}{2} - b, 0 \leq b \leq \frac{n}{2} - 1\}$$

where p_b is defined by $p_b = \prod_{j=1}^{b}(x_{2j-1} + x_{2j}) \in B_{2b}$, and $WS(p_b) = \{b\}$.

For $n = 2^m$, suppose that $1 \leq t \leq m-1$ and $1 \leq i_t \leq 2^t - 1$. Let

$$q_{t,i_t} = q_{t,i_t}(x_{n-2^{t+1}+1}, \cdots, x_n) \in SB_{2^{t+1}}, \quad WS(q_{t,i_t}) = \{i_t, i_t + 2^t\}$$

$$q = q(x_1, x_2, \cdots, x_n) \in SB_n, \quad WS(q) = \{0, \frac{n}{2}, n\}$$

Let

$$S' = \{f_{t,i_t} = q_{t,i_t} p_{\frac{n}{2} - 2^t} | 1 \leq t \leq m-1, 1 \leq i_t \leq 2^t - 1\} \cup \{q\}$$

$$WS(f_{t,i_t}) = \{i_t, i_t + 2^t\} + (\frac{n}{2} - 2^t) = \{\frac{n}{2} + i_t - 2^t, \frac{n}{2} + i_t\}$$

Then the following theorem was proved in [14] in highly combinational way.

Theorem 2. *[14] $S_{min}(n, \frac{n}{2}) = S'$*

By Theorem 2 one can deduce that the sufficient condition of Theorem 1 is true.

3 Results and Proofs

Let $SB(n,d) = \{f \in SB_n, \deg(f) < d\}$ and $WSB(n,d) = \{WS(f), f \in SB(n,d)\}$. Recall that $S(n,d) = \{h_b p_b | h_b \in SB(n-2b, d-b), 0 \le b < d\}$, then

$$S(n,d) = \bigcup_{l=0}^{d-1} \{SB(n-2l, d-l) \cdot p_l\}$$

here for a set of Boolean functions A, $p_l \cdot A$ means the set $\{p_l \cdot f | f \in A\}$.

Let $WS(n,d) = \{WS(f), f \in S(n,d)\}$, then similarly we have

$$WS(n,d) = \bigcup_{l=0}^{d-1} \{WSB(n-2l, d-l) + l\}$$

here for a set of integer sets A, $l + A$ means the set $\{\{s + l, s \in S\} | \forall S \in A\}$.(Any element of $WSB(n-2l, d-l)$ is weight support of some function, thus a set of integers)

For convenience, we use not {} but () to denote the weight support of a Boolean function, for example $WS(f) = (a_1, a_2, \cdots, a_t)$. And we call t the size of $WS(f)$ and $WS(f)$ a size t element.

The problem to determine the set $S_{min}(n,d)$ was raised in [14]. However, to study the AI of symmetric Boolean functions, according to Lemma 4 what we actually need is to determine the set $WS_{min}(n,d)$. Particularly, to study the n-variable symmetric Boolean functions with maximum AI for even n, what we actually need is to determine the set $WS_{min}(n, \frac{n}{2})$.

Problem 2. For $1 \le d \le \lceil \frac{n}{2} \rceil$, to determine the set $WS_{min}(n,d)$. Particularly, when n is even, let $d = \frac{n}{2}$, to determine the set $WS_{min}(n, \frac{n}{2})$.

From the definition of $WS(n, \frac{n}{2})$ we can get the following proposition.

Proposition 1. $WS_{min}(n, \frac{n}{2}) \subseteq \bigcup_{l=0}^{\frac{n}{2}-1} \{WSB_{min}(n-2l, \frac{n}{2} - l) + l\}$

From [14] we know that the set $WS_{min}(n, \frac{n}{2})$ for $n = 2^m$ is

$$WS_{min}(2^m, 2^{m-1}) = \{(2^{m-1} + i_t - 2^t, 2^{m-1} + i_t) | 1 \le t \le m-1, 1 \le i_t \le 2^t - 1\} \cup \{(0, 2^{m-1}, 2^m)\}$$

Here we show some properties of the set $WS_{min}(n, \frac{n}{2})$ for general n. From now on let n be an even integer satisfying $2^m < n = 2^m + 2s < 2^{m+1}$. Let

$$WS(S') = \{(\frac{n}{2} + i_t - 2^t, \frac{n}{2} + i_t) | 1 \le t \le m-1, 1 \le i_t \le 2^t - 1\} \cup \{(\frac{n}{2} - 2^{m-1}, \frac{n}{2}, \frac{n}{2} + 2^{m-1})\}$$

It's easy to show that $WS(S')$ contains all size two elements of $WS_{min}(n, \frac{n}{2})$.

Proposition 2. *Suppose that* $WS(f) \in WS_{min}(n, \frac{n}{2})$ *and* $|WS(f)| = 2$. *Then* $WS(f) \in WS(S')$.

Proof: Assume that $WS(f) \notin WS(S')$. Let $WS(f) = (a, b), a < b$. Then for any integer $1 \le t \le m-1$ and any integer $1 \le i_t \le 2^t - 1$, we have

$$(\frac{n}{2} + i_t - 2^t, \frac{n}{2} + i_t) \ne (a, b) \qquad (1)$$

Let $f' = f \cdot P_{2^m - \frac{n}{2}} = f \cdot (x_{n+1} + x_{n+2}) \cdots (x_{2^{m+1}-1} + x_{2^{m+1}})$. Then $f' \in S(2^{m+1}, 2^m)$ and $WS(f') = WS(f) + \frac{2^{m+1}-n}{2} = (a,b) + \frac{2^{m+1}-n}{2}$. So there exist an integer $1 \le t_0 \le m$ and an integer $1 \le i_{t_0} \le 2^{t_0} - 1$ such that

$$(2^m + i_{t_0} - 2^{t_0}, 2^m + i_{t_0}) = (a,b) + \frac{2^{m+1} - n}{2}$$

This means

$$(\frac{n}{2} + i_{t_0} - 2^{t_0}, \frac{n}{2} + i_{t_0}) = (a,b) \qquad (2)$$

From equation (1) and (2), we have $t_0 = m$ and $b = a + 2^m$. Let $f = h \cdot p_l$. Then $h \in SB(n-2l, \frac{n-2l}{2})$ and $WS(h) = (a-l, a-l+2^m)$. Now let $h = \sum_{i=0}^{n-2l} \lambda_h(i)\sigma_i^{n-2l}$. Then by Lemma 1,

$$\lambda_h(2^m - 1) \equiv \binom{2^m - 1}{a - l} + \binom{2^m - 1}{a - l + 2^m} \equiv \binom{2^m - 1}{a - l} \equiv 1 \pmod 2$$

The last equality holds because $a - l \le 2^m - 1$. However, as $2^m - 1 = \frac{2^{m+1}-2}{2} \ge \frac{n}{2} \ge \frac{n-2l}{2}$, $\deg(h) \ge \frac{n-2l}{2}$ follows. This is a contradiction with the fact that $h \in SB(n-2l, \frac{n-2l}{2})$. Thus the proposition is proved. □

Proposition 3

$$(1)\ WS(S') \subseteq WS_{min}(n, \frac{n}{2});$$

$$(2)\ WS_{min}(n, \frac{n}{2}) \subseteq \bigcup_{l=1}^{s} \{WSB_{min}(2^m + 2l, 2^{m-1} + l) + (s - l)\} \cup WS(S')$$

Proof: At first, it should have $|A| \ge 2$ for any $A \in WS_{min}(n, \frac{n}{2})$. Otherwise, if there exists $A \in WS_{min}(n, \frac{n}{2})$ such that $|A| = 1$, then by Lemma 1 any n-variable symmetric Boolean function can't arrive maximum AI. This is impossible. So we have $|A| \ge 2$ for any $A \in WS_{min}(n, \frac{n}{2})$. And if $A \in WS(n, \frac{n}{2})$ and $|A| = 2$, then $A \in WS_{min}(n, \frac{n}{2})$. For $B = (\frac{n}{2} - 2^{m-1}, \frac{n}{2}, \frac{n}{2} + 2^{m-1}) \in WS(S')$, it's easy to verify that $A \not\subseteq B$ holds for any $B \ne A \in WS(S')$. And by Proposition 2, we know the size two elements are all in $WS(S')$. So $B \in WS_{min}(n, \frac{n}{2})$. Then (1) is proved.

To prove (2), from Proposition 1, what we need is to prove that for any even $n' \le 2^m$ and any $f \in S(n', \frac{n'}{2})$, there exist an element $A \in WS(S')$ such that $A \subseteq \{WS(f) + \frac{n-n'}{2}\}$.

As $n' \le 2^m$, let $f' = f \cdot P_{2^{m-1} - \frac{n'}{2}} = f(x_1, \cdots, x_{n'})(x_{n'+1} + x_{n'+1}) \cdots (x_{2^m - 1} + x_{2^m})$. Then $f' \in S(2^m, 2^{m-1})$. So by Theorem 2 there exists $A \in WS(S')$ such that $A - (\frac{n}{2} - 2^{m-1}) \subseteq WS(f')$. And as $WS(f') = WS(f) + (2^{m-1} - \frac{n'}{2})$, so $WS(f) + \frac{n-n'}{2} = WS(f') + \frac{n}{2} - 2^{m-1}$. Finally one can conclude that $A \subseteq \{WS(f') + (\frac{n}{2} - 2^{m-1})\} = \{WS(f) + \frac{n-n'}{2}\}$ □

Remark 1. Let $2^m < n = 2^m + 2s < 2^{m+1}$. Suppose that $f \in SB_n$ and $AI(f) = \frac{n}{2}$. Then by Lemma 4 and Proposition 3, we have

$$A \not\subseteq WS(f) \text{ and } A \not\subseteq WS(f+1), \quad \forall A \in WS(S')$$

This is just the condition: For any $1 \le t \le m-1$ and any $1 \le k_t \le 2^t - 1$, we have

$$v_f(\frac{n}{2} - 2^t + k_t) = v_f(\frac{n}{2} + k_t) + 1$$

and $(v_f(\frac{n}{2} - 2^{m-1}), v_f(\frac{n}{2}), v_f(\frac{n}{2} + 2^{m-1})) \notin \{(0,0,0),(1,1,1)\}$. This is just Theorem 2.2 of [17].

Then by Lemma 4, Proposition 3 and the definition of AI, one can get the following theorem easily.

Theorem 3. *Suppose that* $2^m < n = 2^m + 2s < 2^{m+1}$ *and* $f \in SB_n$. *Then* $AI(f) = \frac{n}{2}$ *if and only if the following two conditions are both satisfied:*

(1) For any $1 \le t \le m-1$ *and any* $1 \le k_t \le 2^t - 1$, *we have*

$$v_f(\frac{n}{2} - 2^t + k_t) = v_f(\frac{n}{2} + k_t) + 1$$

and $(v_f(\frac{n}{2} - 2^{m-1}), v_f(\frac{n}{2}), v_f(\frac{n}{2} + 2^{m-1})) \notin \{(0,0,0),(1,1,1)\}$.

(2) For any even $2^m < n' \le n$ *and any* $g \in SB_{min}(n', \frac{n'}{2})$, *we have*

$$WS(g) + \frac{n-n'}{2} \nsubseteq WS(f) \text{ and } WS(g) + \frac{n-n'}{2} \nsubseteq \overline{WS(f)}$$

Theorem 3 tells us a fast algorithm to test whether an n-variable symmetric Boolean function arrives maximum AI. For a given even integer n', there are $2^{\frac{n'}{2}}$ functions in $SB(n', \frac{n'}{2})$, so the time complexity to test whether Condition (2) holds is no more than

$$2^{\frac{2^m+2}{2}} + 2^{\frac{2^m+4}{2}} + \cdots + 2^{\frac{n}{2}} = 2^{\frac{n}{2}+1} - 2^{2^{m-1}+1} = 2^{2^{m-1}+s+1} - 2^{2^{m-1}+1}$$

And the time complexity to test whether Condition (1) holds is quite small. So when $s \ge 1$, the time complexity for this algorithm to test whether a n-variable symmetric Boolean function arrive maximum AI is $O(2^{2^{m-1}+s+1})$. Further, there are $\sum_{t=1}^{m-1}(2^t - 1) + 1 = 2^m - m$ equations in Condition (1). The probability for a random function $f \in SB_n$ to satisfy the last equation is $\frac{3}{4}$, while this probability is turned to $\frac{1}{2}$ for any other equation. Then there are total $\frac{3}{4} \times 2^{n+1-(2^m-m-1)} = 3 \times 2^{2s+m}$ functions $f \in SB_n$ who can satisfy Condition (1). So the time complexity to get all the n-variable symmetric Boolean function with maximum AI by this algorithm is $O(3 \times 2^{2s+m} \times 2^{2^{m-1}+s+1}) = O(3 \times 2^{2^{m-1}+3s+m+1})$.

Theorem 3 also shows that when studying the structure of $WS_{min}(n, \frac{n}{2})$, one does not need to study the weight supports of all the $n' \le 2^m$-variable symmetric Boolean functions, because what these elements(after shift) can contribute to $WS_{min}(n, \frac{n}{2})$ are just $WS(S')$. One only needs to study the weight supports of all the $2^m < n' \le n$-variable symmetric Boolean functions and then a proper shift($\frac{n-n'}{2}$) of these weight supports.

When $2s = 2$ is the smallest positive even integer, we can get the following proposition.

Proposition 4. *Suppose that* $n = 2^m + 2$. *Then*

$$WS_{min}(n, \frac{n}{2}) \subseteq WS(S') \cup WS(\{f \in SB_n, \deg f = 2^{m-1}\})$$

Proof: From Proposition 3 we know

$$WS_{min}(n, \frac{n}{2}) \subseteq WSB_{min}(2^m + 2, 2^{m-1} + 1) \cup WS(S')$$

By the definition of $S(n, d)$, $\deg(f) \leq 2^{m-1}$ holds for any $f \in SB(2^m + 2, 2^{m-1} + 1)$. Then it's enough to show that if $\deg(f) < 2^{m-1}$, then there exist $A \in WS(S')$ such that $A \subseteq WS(f)$.

If $\deg(f) < 2^{m-1}$, let $f = \sum_{i=0}^{2^{m-1}-1} \lambda_f(i)\sigma_i^{2^m+2}$ and $f' = \sum_{i=0}^{2^{m-1}-1} \lambda_f(i)\sigma_i^{2^m}$. Then $WS(f') \subseteq WS(f)$. As $f' \in SB(2^m, 2^{m-1})$, we know there exist $A \in \{(2^{m-1}+i_t-2^t, 2^{m-1}+i_t) | 1 \leq t \leq m-1, 1 \leq i_t \leq 2^t-1\} \cup \{(0, 2^{m-1}, 2^m)\}$ such that $A \subseteq WS(f') \subseteq WS(f)$. Now we discuss the situations into four cases:

(1) If $(0, 2^{m-1}, 2^m) \subseteq WS(f') \subseteq WS(f)$, let $t = m-1$ and $i_{m-1} = 2^{m-1} - 1$. Then we have $(\frac{n}{2} + i_{m-1} - 2^{m-1}, \frac{n}{2} + i_{m-1}) = (2^{m-1}, 2^m) \in WS(S')$ and $(2^{m-1}, 2^m) \subseteq WS(f') \subseteq WS(f)$.

(2) If there exists $1 \leq t \leq m-1$ and $2 \leq i_t \leq 2^t-1$ satisfying $(2^{m-1}+i_t-2^t, 2^{m-1}+i_t) \subseteq WS(f') \subseteq WS(f)$, then $(\frac{n}{2}+(i_t-1)-2^t, \frac{n}{2}+(i_t-1)) \subseteq WS(f') \subseteq WS(f)$. And as $i_t \geq 2$, $i_t - 1 \geq 1$, we also have $(\frac{n}{2} + (i_t-1) - 2^t, \frac{n}{2} + (i_t-1)) \in WS(S')$.

(3) If $(1, 2^{m-1} + 1) \subseteq WS(f')$, then by Lemma 1 we know $(\frac{n}{2} - 2^{m-1}, \frac{n}{2}, \frac{n}{2} + 2^{m-1}) = (1, 2^{m-1} + 1, 2^m + 1) \subseteq WS(f)$.

(4) If the above three cases all don't occur, then we have for some $1 \leq t \leq m-2$, $(2^{m-1} + 1 - 2^t, 2^{m-1} + 1) \subseteq WS(f') \subseteq WS(f)$. Then it can be concluded that $WS(f') = \{2^{m-1} + 1 - 2^{t_1}, 2^{m-1} + 1 - 2^{t_2}, \cdots, 2^{m-1} + 1 - 2^{t_l}, 2^{m-1} + 1\}$, where $1 \leq t_1 \leq t_2 \leq \cdots \leq t_l \leq m-2$. However, $\binom{2^{m-1}+1}{2^{m-1}+1-2^t} = 0$ holds for any $1 \leq t \leq m-2$. So by Lemma 1 we have

$$\begin{aligned}\lambda_f(2^{m-1}+1) &= \binom{2^{m-1}+1}{2^{m-1}+1} + \binom{2^{m-1}+1}{2^{m-1}+1-2^{t_1}} + \binom{2^{m-1}+1}{2^{m-1}+1-2^{t_2}} + \cdots + \binom{2^{m-1}+1}{2^{m-1}+1-2^{t_l}} \\ &= 1 + 0 + 0 + \cdots + 0 = 1\end{aligned}$$

This is a contradiction with $\deg(f) < 2^m$.

From the above discussions, we know that if $\deg(f) < 2^{m-1}$, then there always exists $A \in WS(S')$ such that $A \subseteq WS(f)$. So if $WS(f) \in WS_{min}(n, \frac{n}{2}) \setminus WS(S')$, then $f \in SB_n$ and $\deg(f) = 2^{m-1}$. The proposition is thus proved. □

Proposition 5. *Suppose that* $n = 2^m + 2 (m \geq 2)$, $f = \sum_{i=0}^{n} \lambda_f(i)\sigma_i^n \in SB(n, \frac{n}{2})$, *and* $WS(f) \in WS_{min}(n, \frac{n}{2}) \setminus WS(S')$. *If* $\lambda_f(0) = 0$, *then*

$$\lambda_f(2) = \lambda_f(2^2) = \cdots = \lambda_f(2^{m-2}) = 1$$

Proof: Assume there exists $1 \leq t \leq m-2$ such that $\lambda_f(2^t) = 0$. By Proposition 4 we know $\lambda_f(2^{m-1}) = 1$. Then by Lemma 1 we have

$$v_f(2^{m-1}) = \lambda_f(2^{m-1}) + \lambda_f(0) = 1 + 0 = 1$$

And by $2^{m-1}+2^t > \frac{n}{2} = 2^{m-1}+1$ and $f \in SB(n, \frac{n}{2})$, one can conclude that $\lambda_f(2^{m-1}+2^t) = 0$. So by Lemma 1 we know

$$v_f(2^{m-1} + 2^t) = \lambda_f(0) + \lambda_f(2^{m-1}) + \lambda_f(2^t) + \lambda_f(2^{m-1} + 2^t) = 0 + 1 + 0 + 0 = 1$$

Thus $v_f(2^{m-1}) = v_f(2^{m-1} + 2^t) = 1$ holds. This means $(2^{m-1}, 2^{m-1} + 2^t) \subseteq WS(f)$. Let $i_t = 2^t - 1$. Then

$$(\frac{n}{2} + i_t - 2^t, \frac{n}{2} + i_t) = (2^{m-1}, 2^{m-1} + 2^t) \in WS(S')$$

This contradicts with the fact that $WS(f) \in WS_{min}(n, \frac{n}{2}) \setminus WS(S')$. So $\lambda_f(2^t) = 1$ holds for any $1 \le t \le m-2$. This completes the proof. □

For $n = 2^m + 2$, Proposition 4 and Proposition 5 tell one a smaller range to find the elements of $WS_{min}(n, \frac{n}{2})$. This means a lower time complexity of the algorithm to get all the n-variable symmetric Boolean functions with maximum AI. Computer simulations show that there are many elements in $WS_{min}(n, \frac{n}{2}) \backslash WS(S')$ and it seems difficult to find a simple rule to describe $WS_{min}(n, \frac{n}{2}) \backslash WS(S')$. This means for $n = 2^m + 2$, it's difficult to get a nice characterization of all $f \in SB_n$ with maximum AI.

Next proposition is a result about $WSB_{min}(n, \frac{n}{2})$ for $n = 2^{m+1} - 2$.

Proposition 6. *Suppose that* $n = 2^{m+1} - 2$. *Let* $WS_3 = \{WS(g) = (a, 2^m - 1, a + 2^m) | g \in SB_n, 0 \le a \le 2^m - 2\}$ *and* $WS_4 = \{WS(g) = (a, b, a + 2^m, b + 2^m) | g \in SB_n, 0 \le a < b \le 2^m - 2\}$. *Then* $WSB_{min}(n, \frac{n}{2}) \subseteq WS_3 \cup WS_4$.

Proof: Let $0 \ne f \in SB(n, \frac{n}{2})$ and $f = \sum_{i=1}^{n} \lambda_f(i) \sigma_i^n$. As $\deg(f) < \frac{n}{2} = 2^m - 1 < 2^m$, we have

$$v_f(2^m + j) = \sum_{i \le 2^m + j} \lambda_f(i) = \sum_{i \le j} (\lambda_f(i) + \lambda_f(2^m + i)) = \sum_{i \le j} \lambda_f(i) = v_f(j)$$

for any $0 \le j \le 2^m - 2$.

Let $t = \#\{j | 0 \le j \le 2^m - 2, v_f(j) = 1\}$. Then one can conclude that $t > 0$. Otherwise we have $v_f(0) = v_f(1) = \cdots = v_f(2^m - 2) = v_f(2^m) = v_f(2^m + 1) = \cdots = v_f(2^{m+1} - 2)$. By $f \ne 0$, we have $v_f(2^m - 1) = 1$. Then by Lemma 1, we have $\lambda_f(2^m - 1) = 1$. This is a contradiction with $\deg(f) \le 2^m - 2$. Now we discuss the situations according to the value of t.

(1) If $t \ge 2$, then there exists $A \in WS_4$ such that $A \subseteq WS(f)$;

(2) If $t = 1$, assume $v_f(a) = 0$. If $v_f(2^m - 1) = 1$, then $WS(f) = (a, 2^m - 1, 2^m + a) \in WS_3$. If $v_f(2^m - 1) = 0$, then $WS(f) = (a, 2^m + a)$. So we have

$$\lambda_f(2^m - 1) = \binom{2^m - 1}{a} + \binom{2^m - 1}{2^m + a} = 1 + 0 = 1$$

This is a contradiction with $\deg(f) < \frac{n}{2}$.

So for any $f \in SB(n, \frac{n}{2})$, there exists $A \in WS_3 \cup WS_4$ such that $A \subseteq WS(f)$. Then we can conclude that $WSB_{min}(n, \frac{n}{2}) \subseteq WS_3 \cup WS_4$. □

Though we can't get $WS_{min}(n, \frac{n}{2})$ for general n, we can have some results about the small size elements of $WS_{min}(n, \frac{n}{2})$. The following two propositions are about the size three elements of $WS_{min}(n, \frac{n}{2})$.

Proposition 7. *If* $(a, b, c) = WS(f) \in WS_{min}(n, \frac{n}{2})$, $a < b < c$, *then* $c - a = 2^m$.

Proof: If $WS(f) \in WS(S')$, then $WS(f) = (\frac{n}{2} - 2^{m-1}, \frac{n}{2}, \frac{n}{2} + 2^{m-1}) = (a, b, c)$. Thus $c - a = 2^m$. The proposition is true.

If $WS(f) \notin WS(S')$, then by the similar proof as Proposition 2, there exists $1 \le i_m < 2^m$ such that

$$(\frac{n}{2} + i_m - 2^m, \frac{n}{2} + i_m) \subseteq (a, b, c)$$

This means

$$b - a = 2^m \text{ or } c - a = 2^m \text{ or } c - b = 2^m$$

Now we go to prove that only $c - a = 2^m$ can holds.

First we assume that $f \in SB(n, \frac{n}{2})$, then we have:

If $b - a = 2^m$, then $c > b \ge 2^m$ and $a \le 2^m - 1$. So we have $\lambda_f(2^m - 1) = \binom{2^m-1}{a} + \binom{2^m-1}{2^m+a} + \binom{2^m-1}{c} = \binom{2^m-1}{a} = 1$. But $2^m - 1 = \frac{2^{m+1}-2}{2} \ge \frac{n}{2}$, this means $\deg(f) \ge \frac{n}{2}$. This is a contradiction with the fact that $f \in SB_{min}(n, \frac{n}{2})$.

If $c - b = 2^m$, then $a < b \le 2^m - 1$. So we have $\lambda_f(2^m + a) = \binom{2^m+a}{a} + \binom{2^m+a}{b} + \binom{2^m+a}{2^m+b} = \binom{2^m+a}{a} = 1$. But $2^m + a > \frac{n}{2}$, this means $\deg(f) \ge \frac{n}{2}$. This is a contradiction with the fact that $f \in SB_{min}(n, \frac{n}{2})$.

So we have $c - a = 2^m$ for $f \in SB(n, \frac{n}{2})$. If $f \notin SB(n, \frac{n}{2})$, let $f = h \cdot p_l$. Then $h \in SB(n - 2l, \frac{n-2l}{2})$ and $WS(h) = (a - l, b - l, c - l)$. Similarly, we can prove that $(c - l) - (a - l) = 2^m$, which also means that $c - a = 2^m$. We then prove the proposition. □

Proposition 8. *Suppose that* $n = 2^m + 2s$ *and* $0 < 2s < 2^m$. *Let* $0 \le l \le m - 2$ *be the integer such that* $2^{m-1} - 1 - 2^{l+1} < s \le 2^{m-1} - 1 - 2^l$. *Let* $f \in SB(n, \frac{n}{2})$, $WS(f) \in WS_{min}(n, \frac{n}{2})$, *and* $|WS(f)| = 3$. *Then* $WS(f) = (a, b, a+2^m)$, *where* $0 \le a \le 2s$, $2s < b \le 2^m - 1$, *and* $a \equiv b(2^{l+1})$.

Proof: By Proposition 7, we know $WS(f) = (a, b, a + 2^m)$, $0 \le a \le 2s$, and $2s < b \le 2^m - 1$. Then we only to prove that $a \equiv b(2^{l+1})$.

From $f \in SB(n, \frac{n}{2})$ we have

$$\lambda_f(2^{m-1} + s) = \lambda_f(2^{m-1} + s + 1) = \cdots = \lambda_f(2^m - 1) = 0$$

This means

$$\lambda_f(2^{m-1} + j) = \binom{2^{m-1} + j}{a} + \binom{2^{m-1} + j}{b} = 0, \quad \forall j, \ s \le j \le 2^{m-1} - 1 \qquad (*)$$

Assume $a \not\equiv b(2^{l+1})$. Let $a = (a_k a_{k-1} \cdots a_0)_2$ and $b = (b_k b_{k-1} \cdots b_0)_2$. Then there exists $0 \le i \le l$ such that $a_i \ne b_i$. Let $j_0 = 2^{m-1} - 2^i - 1$. Then we have

$$\lambda_f(2^{m-1} + j_0) = \binom{2^{m-1} + j_0}{a} + \binom{2^{m-1} + j_0}{b} = 1$$

Since $s \le j_0 \le 2^{m-1} - 1$, this is a contradiction with the equation $(*)$. So $a \equiv b(2^{l+1})$ must holds. □

4 Conclusion

The weight support technique of [14] was developed in the paper. For a general even integer n, the problem to study the n-variable symmetric Boolean functions with maximum AI was reduced to the problem to determine $WS_{min}(n, \frac{n}{2})$. Then some new results about the general properties of $WS_{min}(n, \frac{n}{2})$ were got. These results showed a fast algorithm to get all $f \in SB_n$ with maximum AI. Then some results for special values of $n(n = 2^m + 2$ and $n = 2^{m+1} - 2)$ were introduced. At last, some results about the size three elements of $WS_{min}(n, \frac{n}{2})$ were shown. Theory analysis and computer simulations both show that for a general even integer n, it seems not easy to get a beautiful characterization as for $n = 2^m$. Some new results were given in the paper. These results are helpful to solve all $f \in SB_n$ with maximum AI for general even n, but the final solution remains open.

References

1. Armknecht, F.: Improving Fast Algebraic Attacks. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 65–82. Springer, Heidelberg (2004)
2. Batten, L.M.: Algebraic Attacks over GF(q). In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 84–91. Springer, Heidelberg (2004)
3. Braeken, A., Preneel, B.: On the algebraic immunity of symmetric Boolean functions. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 35–48. Springer, Heidelberg (2005)
4. Canteaut, A., Videau, M.: Symmetric Boolean functions. IEEE Tran. Inf. Theory 51(8), 2791–2811 (2005)
5. Carlet, C., Gaborit, P.: On the construction of balanced Boolean functions with a good algebraic immunity. In: Proceedings of BFCA (First Workshop on Boolean Functions: Cryptography and Applications), Rouen, France, March 2005, pp. 1–14 (2005)
6. Carlet, C., Dalai, D.K., Gupta, K.C., Maitra, S.: Algebraic Immunity for Cryptographically Significant Boolean Functions: Analysis and Construction. IEEE Tran. Inf. Theory 52(7), 3105–3121 (2006)
7. Courtois, N., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)
8. Courtois, N., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003)
9. Courtois, N.: Fast algebraic attacks on stream ciphers with linear feedback. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 176–194. Springer, Heidelberg (2003)
10. Dalai, D.K., Gupta, K.C., Maitra, S.: Results on Algebraic Immunity for Cryptographically Significant Boolean Functions. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 92–106. Springer, Heidelberg (2004)
11. Dalai, D.K., Maitra, S., Sarkar, S.: Basic theory in construction of Boolean functions with maximum possible annihilator immunity. Des. Codes, Cryptogr. 40(1), 41–58 (2006), http://eprint.iacr.org/2005/229
12. Feng, K.Q., Liu, F., Qu, L.J., Wang, L.: Constructing Symmetric Boolean Functions with Maximum Algebraic Immunity (preprint, 2006)
13. Li, N., Qu, L.J., Qi, W.F., Feng, G.Z., Li, C., Xie, D.Q.: Construction and Count the Boolean Functions with Optimum Algebraic Immunity. IEEE Tran. Inf. Theory 54(3), 1330–1334 (2008)

14. Liu, F., Feng, K.Q.: On the 2^m-variable symmetric Boolean functions with maximum algebraic immunity 2^{m-1}. In: Workshop on Coding and Cryptography 2007, to be published in Des. Codes, Cryptogr. (2007)
15. Meier, W., Pasalic, E., Carlet, C.: Algebraic attacks and decomposition of Boolean functions. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 474–491. Springer, Heidelberg (2004)
16. Qu, L.J., Li, C., Feng, K.Q.: A Note on Symmetric Boolean Functions with Maximum Algebraic Immunity in Odd Number of Variables. IEEE Tran. Inf. Theory 53(8), 2908–2910 (2007)
17. Qu, L.J., Li, C.: On the 2^m-variable Symmetric Boolean Functions with Maximum Algebraic Immunity. Science in China Series F-Information Sciences 51(2), 120–127 (2008)
18. Qu, L.J., Feng, G.Z., Li, C.: On the Boolean Functions with Maximum Possible Algebraic Immunity : Construction and A Lower Bound of the Count, http://eprint.iacr.org/2005/449

Anonymity and k-Choice Identities[⋆]

Jacek Cichoń and Mirosław Kutyłowski

Institute of Mathematics and Computer Science,
Wrocław University of Technology, Poland
jacek.cichon@pwr.wroc.pl, miroslaw.kutylowski@pwr.wroc.pl

Abstract. We consider pervasive systems and identifiers for objects in these systems. Using unique global identifiers for these objects increases the size of the ID's and requires some global coordination. However, severe privacy threats are the key issues here.

On the other hand, for performing the goals of a pervasive system the identifiers are normally used in *small local environments*, and we need uniqueness limited to these environments only. This yields an opportunity to re-use the ID's and in this way anonymize the objects. The problem is that we cannot predict assignment of the objects to local environment or set it in advance, while on the other hand in many application scenarios we cannot change an ID already assigned to an object. Random predistribution of ID's is a technique that partially solves this problem, but has drawbacks due to the birthday paradox.

We propose a solution in which each object holds k preinstalled ID's (where k is a small parameter like $k = 2, 3, ...$). While entering a local environment, one of its ID's not used so far in this local environment is chosen for the object. We analyze probability of a conflict, i.e. of the event that no identity can be chosen for this object. We show that the size of ID's may be significantly reduced compared to random predistribution without increasing conflict probability. Apart from implementation advantages it contributes to privacy protection: since globally a large number of objects holds the same ID, privacy threats are reduced.

Keywords: anonymity set, identifier, two-choice paradigm, birthday paradox.

1 Introduction

Recently, it has been widely recognized that processing electronic data may yield severe privacy problems (see for instance a report [13]). The problem is that single data transaction might be regarded as fully safe from a privacy point of view, but collections of these data may leak sensitive private information. Among others, this is caused by the fact that electronic identification is expanding and used not only to recognize physical persons or machines, but also simple electronic

[⋆] The paper is partially supported by EU within the 6th Framework Programme under contract 001907 (DELIS).

devices used for diverse purposes. Since the number of such devices (like for instance RFID tags) is growing, the scope of the problem is expanding as well.

Technology for privacy protection has been recently developed by many research groups and companies. For instance, policy based approach has been proposed within an EU Project PRIME [12]. Most of the work is devoted to identity management and limiting privacy threats by appropriate organization, privacy policies and similar measures. This approach has limitations – it is not much useful when the devices concerned have very limited resources or the mentioned measures are poorly implemented. However, some papers propose solutions that are not based on proper behavior of system actors (see e.g. [9,5]). The concept is to guarantee some level of security by technical means only.

Privacy Problems for Pervasive Systems and RFID's. The objects in pervasive systems must be identified for the purposes of running these systems. Once the objects are given unique identifiers, the system can be used for tracing objects and, in this way, for tracing people holding these objects. This turns out to be one of the major problems in usage of RFID systems is retail stores. Alone the possibility of illegitimate tracing the clients and their preferences brings severe legal problems for the enterprises deploying such systems. Since more and more simple devices can include RFID's, this becomes one of the most acute security problems for emerging technologies of pervasive systems. Unfortunately, the devices need to be extremely simple, so the solutions designed for traditional networks composed of powerful computational units are of no use in majority of cases. In the simplest case we have to do with a memory unit with just a few bytes, which can be read by an external reader with no authorization performed before giving access to data.

Identification Problem in Local Environments. Our approach is inspired by the fact that even if a global system may consist of a huge number of possible ID's, the system is composed of a number of small environments with a limited number of objects and each function of the system is performed in some small environment. We have the following conflicting demands:

– all objects in a small environment should have different ID's,
– each ID may be linked to a large number of different objects in the global system, so that tracing a single object becomes complex.

Let us remark that the social mechanisms are exactly of this kind: ID's used in most cases of everyday life are not unique. For instance, even the first name and family name of a person does not identify a physical person uniquely, nevertheless the goals of identification are achieved. The general framework looks as follows:

– each person holds a couple of ID's (such as the first names) that are not unique in the whole society,
– in each system concerned there is a limited number of participants, identification within the system is performed with the ID's mentioned.

Nevertheless, there is a tendency to build information systems where each unit is assigned a unique digital ID. This makes design of databases much easier, but on the other hand imposes serious privacy threats. Consequently, a lot of money need to be invested in protection of databases, restricting access, authorization.

In this paper we examine how to achieve local uniqueness with as much repetitions of a single identifier on a global scale as possible.

2 *k*-Choice ID's Protocol

The k-choice protocol examined in this paper is extremely simple. It consists of the following subprotocols:

Predistribution Phase. In each object, the manufacturer installs k ID's that are derived at random, or in a pseudo-random way. For instance, they might be $H(K,T,1),\dots H(K,T,k)$, where K is the master key of the manufacturer, T is a serial number or the moment of generating the ID's, and H is a secure hash function truncated to the required length m of the ID's.

Registration in a local environment. When an object is to be registered in a local system, then the system inspects which of the k ID's assigned to the object has not been used yet in this system, and chooses one of them as the identifier of the object to be used in this environment.

Of course, for $k = 1$ we get the well-known random predistribution scheme. We consider the following parameters of the solution proposed:

- m is the length of the ID numbers,
- t is an upper bound for the number of objects in a local environment,
- n is the global number of objects,
- L is the the minimal size of each anonymity set, that is, the number of objects holding the same ID (or ID's) as a given object,
- ε is the admissible probability of a collision, that is, of an event that a new object to be registered in a local environment has ID's that are all already in use in this environment.

The design goals are the following:

- m should be as small as possible (in this way L becomes large),
- ε should be as small as possible.

Of course, these are conflicting goals and we have to balance them. In this paper we examine exactly this relationship. The most important issue is the impact of k. For practical reasons, we are interested in estimations concerning small k's. The most important issue is to find the difference between the situation for $k = 1$ and for $k = 2$, since decision to use $k \geq 2$ has severe consequences for system architecture and anonymity level. We also examine how much improvement can be reached, if we use higher values of k.

Related results. We deal here with the celebrated two-choices paradigm (see [1,7]). There are many papers concerning this technique, they concentrate on the problem of load balancing while allocating t items to N bins: the standard case is that the number of items t equals the number of locations N. The other case considered is that the number of items is higher than the number of locations (see e.g. [3]). The technical results obtained there show that two-choices paradigm (or more generally k-choice paradigm) reduces significantly the maximal load (but it is still above $\Theta(1)$, for the case $t = N$).

In this paper we are interested in a different question. Namely, we ask

what is the maximal number of balls thrown to 2^m bins according to k-choice protocol so that no bin receives more than one ball with a high probability.

Up to our knowledge, for $k > 1$ this problem has not been treated in the literature so far, probably since most of the work has been focused on load balancing.

For $k = 1$, the problem considered here concerns the well-studied birthday paradox (see for instance [4]). As one may expect, applying two-choices paradigm improves situation compared to the case of $k = 1$ and problems related to the birthday paradox. Our main technical contribution are exact analytical results showing which parameters have to be used for the k-choice protocol in order to avoid collisions.

The problem of k-collisions of random assignment (or k-collision of hash functions) considered in [8] is somewhat similar to the problem of collisions for the k-choices protocol. Interestingly, this stochastic process is completely different from the k-choices protocol, but the results concerning the number of items necessary to get k-collision and a collision for the k-choices protocol involve almost the same formulas which differ only by constant factors involved (see [8], compare also [10]). Mathematical nature of these intriguing phenomenon remains unknown to the authors at this moment.

3 Stochastic Model and Outline of Results

3.1 Random Assignment Process

Let us consider the classical bins and balls process:

Given N bins that are initially empty. At each step a new ball is placed in a bin chosen uniformly at random. Proceed until a collision occurs, that is, a bin is chosen which already contains a ball.

Let B_N be a random variable denoting the number of steps executed by random assignment process. It is well known (see e.g. [6]) that $\mathrm{E}[B_N]$, the expected value of B_N, equals $\sqrt{\frac{N\pi}{2}} + \frac{2}{3} + O(\frac{1}{\sqrt{N}})$. However, it is less known that the standard deviation of the random variable B_N is large, namely $\mathrm{std}[B_N] \simeq \sqrt{(2 - \frac{\pi}{2})N} \approx 0.523 \cdot \mathrm{E}[B_N]$. Therefore, finding $\mathrm{E}[B_N]$ does not necessarily provide a satisfactory answer to our problem, since we are interested in collision freedom with high probability. For $N \geq 20$ and $t \leq \sqrt{\frac{\pi N}{2}}$, in Theorem 1 we show that

$$\Pr[B_N \leq t+1] \approx 1 - e^{-\frac{t(t+1)}{2N}} \ . \qquad (1)$$

Let us fix the number of balls t and a (small) probability p. We calculate the minimal number of bins N such that $\Pr[B_N \leq t+1] \leq p$. We get

$$1 - e^{-\frac{t(t+1)}{2N}} \leq p \quad \text{iff} \quad N \geq \frac{-t(t+1)}{2\ln(1-p)} \ . \tag{2}$$

3.2 Two-Choices Random Assignment Process

Let us consider now a process that describes two-choices protocol:

> *Given N bins that are initially empty. At each step a new ball is placed in a bin: first two bins are chosen uniformly at random. Then the ball is placed in any of these bins that is empty. Proceed until a collision occurs, that is, both bins chosen already contain a ball.*

This modification of the classical random assignment process was analyzed by Azar et al. (see [1,2]). Let C_N be a random variable denoting the length of this process. It can be shown that

$$\mathrm{E}[C_N] \approx \sqrt[3]{3} \cdot \Gamma(\tfrac{4}{3}) \cdot n^{\frac{2}{3}} \approx 1.2879 \cdot N^{\frac{2}{3}} \ . \tag{3}$$

(Γ denotes here the well known *Gamma function*). So the expected value of the moment of the first collision in this process is essentially bigger than in the previous process. For $N \geq 5$ and $t < \sqrt[3]{3} \cdot \Gamma(\frac{4}{3}) \cdot N^{2/3}$, we show in Theorem 2 that

$$\Pr[C_N \leq t+1] \approx 1 - e^{-\frac{t(t+1)(2t+1)}{6N^2}} \ . \tag{4}$$

Let us fix the number of balls t and a (small) probability p. We calculate the minimal number of bins N such that $\Pr[C_N \leq t+1] \leq p$. We get

$$1 - e^{-\frac{t(t+1)(2t+1)}{6N^2}} \leq p \quad \text{iff} \quad N \geq \sqrt{\frac{-t(t+1)(2t+1)}{6\ln(1-p)}} \ . \tag{5}$$

3.3 The Case of $k > 2$

Let $N_k(t,p)$ denote the minimal N for which collision probability is at most p for t steps of k-choice random assignment process. Then

$$N_k(t,p) = \left(-\frac{\sum_{a=1}^{t} a^k}{\ln(1-p)} \right)^{1/k} \approx \left(-\frac{t^{k+1}}{(k+1)\ln(1-p)} \right)^{1/k} \tag{6}$$

The last approximation is very precise for $k \geq 3$.

4 Applications - Municipal Ticket System

We shall discuss a large municipal public transportation system. Recently, it becomes popular to switch from traditional paper tickets into systems with electronically readable tickets. The purpose is not merely ease of automatic reading. One of the basic functions of such a system is to provide data about usage of transportation system. Namely:

– each single ride of a passenger should be recorded indicating its start point and the endpoint.

These data is necessary for two reasons. In order to optimize routes and service frequencies on the routes it is necessary to have reliable data on the system usage. The second reason is rational allocating subsidies received by transportation companies - if the subsidies are payed per route serviced (and not per passenger ride), then the transportation companies might be discouraged to optimize their service.

In the situation described the following requirements for the system can be formulated (in fact, they are taken from a real project):

– each passenger is holding either a period ticket or a single trip ticket,
– while entering a car the passenger presents his ticket via a wireless channel to a registration device; the same occurs when the passenger is leaving the car (as in the Singaporean public transportation system),
– each registration device collects all data on the passengers points of entering and leaving the car, this data is transmitted periodically to a central database,
– electronic tickets might be held by sophisticated devices (like cryptographic cards with multiple advanced functions), as well as the simplest memory devices that do not support write operations after completing manufacturing process, except a few bit flags.
– Ticket cloning should be detectable, at least if the scale of cloning becomes economically non-negligible.

The simplest solution in the above scenario would be to place a sufficiently long random identifier on each ticket. This identifier would be used for collecting data on the passengers' trips.

Privacy Problems. The straightforward solution with the unique identifiers is unacceptable due to revealing personal data: records on passenger's traffic at the same time provide data on behavior of single persons. Since the period tickets are nontransferable in the case considered, the name of the ticket holder must be easily accessible and therefore one may link a person with data concerning her/his usage of public transportation. This data can be misused in multiple ways including criminal purposes (for instance it would help a lot to choose targets of burglary without tedious observation of the potential victims).

Even if the data concerned is not used for criminal purposes it is a clear case of violating personal data protection rules. Legal systems in many countries (USA being an important exception) impose very strict rules that should prohibit leaking <u>any</u> personal data. In particular, the information system implemented should <u>guarantee</u> appropriate data protection measures. Of course, many protection mechanisms can be designed, but the price of high security installations is an important issue.

Solution Scenario. We propose the following simple scenario:

1. Each ticket contains a radio frequency tag with k different ID's selected at random from a pool K of N different ID's.
2. When a passenger enters a car, then a control device reads the ID's from the ticket and chooses one of the ID's from the ticket currently not in use in the car, say s. The device puts the flag associated to s to the "on" position and records that a passenger with a ticket s is in the car. No other ID from the tag is recorded by the control device.
3. When a passenger leaves the car, then the flag corresponding to s in the ticket is set on the "off" position. Simultaneously, it is marked in the record corresponding to s that the passenger has finished his travel at this point.
4. Periodically, the records are transmitted to a central database. However not all records are transmitted, but only some of them chosen according to a coloring scheme that will be described later.

Note that the devices do not erase the identifiers related to single rides. This is necessary, since the tickets are low end products and cloning them is not necessarily very difficult. However, if cloning occurs and fake tickets are used in a scale that matters economically, then some ID's occur more frequently than the others and this cannot be explained by stochastic deviations. We assume that the ID's are created cryptographically by the transportation authority and they cannot be determined by the other parties except by reading from valid tickets.

For obvious reasons, the number of passengers in a car is bounded from above by some number t. Therefore the pool of ID's K need not to be very big and the same ID's may be inserted into very many tickets. The only problem that may occur is that all k ID's from a ticket are already in use in the car, i.e. there is a *collision*. There is a trade-off between N (the size of K), parameters k, t and the probability of a collision.

The second issue related to N is the size of anonymity set for each passenger. Probability that a given ID is contained in a single ticket is roughly k/N, so the average number of passengers holding this ID is $\approx nk/N$, where n is the total number of passengers. Additionally, since there are k ID's on each ticket, the average size of the anonymity set for a given passenger is $\approx nk^2/N$ (this is the number of passengers that share at least one ID with a given passenger).

The main technical problem considered in this paper is the choice of the parameters mentioned. Our goal is to choose k and N so that nk^2/N is sufficiently large, while probability of a collision is still small enough. The first goal requires N to be small, the second one requires that N is sufficiently large. As we shall see, for $k > 1$ we get quite satisfactory results.

4.1 Discussion of the Parameter Settings

Let us now derive the consequences of the above estimations for the protocol discussed. We assume that the total number of passengers is $n = 10^6$.

Table 1. Choice of N for the protocol for $k = 1$, collision probability p, and a bound t on the number of passengers; mark $*$ denotes a value higher than n, which means that it is better to use unique identifiers

	$t = 50$	$t = 100$	$t = 200$	$t = 300$
$p = 10^{-2}$	126861	502470	$*$	$*$
$p = 10^{-3}$	$*$	$*$	$*$	$*$

Table 2. Estimated size of anonymity set for $k = 1$

	$t = 50$	$t = 100$	$t = 200$	$t = 300$
$p = 10^{-2}$	7.9	2	1	1
$p = 10^{-3}$	1	1	1	1

Table 3. Appropriate size of N in the case $k = 2$

	$t = 50$	$t = 100$	$t = 200$	$t = 300$
$p = 10^{-2}$	2066	5802	16350	29999
$p = 10^{-3}$	6550	18389	51820	95081
$p = 10^{-4}$	20718	58166	163907	300742
$p = 10^{-5}$	65517	183942	518332	951052

Table 1 contains some values of appropriate N that guarantees a given upper bound on collision probability. We can read from this table that the protocol for $k = 1$ is of no use: collision probability is high, anonymity set is so small that it provides no real privacy protection even for the smallest size of parameters. However, from Table 3 we read that in the case of 2-choices protocol N can be much lower, therefore the anonymity sets are much larger.

As we see, there is a trade-off between collision probability and the size of anonymity set. If we admit $p = 0.01$ (that is, in 1% of cases some passanger will not be registered, which makes about 1% rides unregistered), then for $k = 2$ we get socially acceptable anonymity sets.

One can show that the size of anonymity sets can be further significantly improved for higher values of k. Tables 5 and 6 show the results for $k = 3$.

4.2 Further Security Issues

Of course, any reader placed in a car may record the data from the RFID's of the passengers. So it is unjustified to worry that the control device may store all k keys from a ticket. On the other hand, solutions using more sophisticated equipment than simple passive RFID's might be unattractive from economic point of view.

Table 4. Estimated size of anonymity set for $k = 2$

	$t = 50$	$t = 100$	$t = 200$	$t = 300$
$p = 10^{-2}$	1936	689	244	133
$p = 10^{-3}$	610	217	77	42
$p = 10^{-4}$	193	68	24	13
$p = 10^{-5}$	61	21	7	4

Table 5. Appropriate size of N in the case $k = 3$

	$t = 50$	$t = 100$	$t = 200$	$t = 300$
$p = 10^{-2}$	537	1384	3414	5862
$p = 10^{-3}$	1160	2923	7366	12649
$p = 10^{-4}$	2499	6299	15873	27256
$p = 10^{-5}$	5386	13572	34199	58722

Table 6. Estimated size of anonymity set for $k = 3$

	$t = 50$	$t = 100$	$t = 200$	$t = 300$
$p = 10^{-2}$	16759	6502	2636	1535
$p = 10^{-3}$	7758	3079	1221	711
$p = 10^{-4}$	3601	1428	567	330
$p = 10^{-5}$	1670	663	263	153

The real danger of the system is that all data are collected in one place and therefore can be gained by a malicious adversary in a relatively easy way. If all data on passenger rides are collected, it would be possible through appropriate statistic tools to couple all ID's from one card together. Namely, the adversary would use the points of transfer and try to find first a typical route (via determining which identifiers appear frequently at a certain transfer point), and later find sets of identifiers that can be coupled in the same way on the route quite frequently. We skip here the details of the statistic attack that should be clear to the reader.

A countermeasure against such an attack is to build *a meeting graph* G: its vertices are labeled by all cars. We say that two cars A and B are connected by an edge in G, if there is a transfer point that is visited by A and B and a passenger leaving A may transfer to $B -$ no other car of the same line as B visits the transfer point in the meantime. Having graph G we determine partition of its vertices into independent sets, each marked with a different color.

The scheme of collecting data is that from one period of time we store only data from cars of a single color in G. This assures that we cannot use correlation from the transfer points. The ID's on the tickets are changed each month, so statistic analysis of such data becomes very complex.

Anonymity Set versus Anonymity. So far we have focused our attention on the size of anonymity sets - as it is often the case in the anonymity literature. However even if the size of this set is large, one should be aware of the danger of data mining based on particular properties of a given application. For instance, not all rides with the same ID can be attributed to the same person - this follows from impossibility of certain routes. Another potential chance for traffic analysis is utilizing possible personal behavior patterns.

In a classical approach, the population is partitioned into disjoint anonymity sets based on some observable characteristics. With multiple ID's the situation is more complex and we cannot confine analysis to a given anonymity set: If we know that Alice can use identities $s_1, \ldots, s_k$, then each time we see that identifier s_i is used, we must consider all other passengers holding the same s_i. In order to exclude them we have to analyze them and therefore take into account their anonymity sets. However, these anonymity sets are different than the anonymity set of Alice. So in principle proceeding this way we take into consideration all participants of the protocol and the number of cases that have to be considered explodes.

5 Application - Simple Identifier Tags in Shops

In many systems with diverse objects (like shops), RFID tags are very useful for managing physical objects. As before, the problem is that an object can be used for tracing the clients. Of course, one can kill the tags when leaving a shop, but we can achieve something even if killing is impossible.

We consider an additional requirement: a tag must be readable without any electronic equipment. For instance, there are bar code numbers printed on a sticker containing an RFID-tag so that they are readable by a human. This is motivated by possible hardware failures, lack of appropriate readers at some locations, or dual systems in which RFID-tags are functioning in parallel with printed codes.

While it is easy to insert printed codes during manufacturing phase, they cannot be easily changed later. So we require a predistribution scheme: ID-tags coming to a user are already set and a new ID cannot be assigned to a tag.

The second limitation is the size of ID's. If it is small, then we can reuse the memory place saved. Moreover, if a printed ID is to be read by a human, it is less error-prone to have a short ID. Secondly, if an object is leaving a shop, then its ID should not serve for tracing its holder. So we are again in a scenario considered in Sec. 4, where repetitions between tags serve for hiding information.

We can use two-choices random assignment scheme in the following way:

– Each RFID tag receives two predistributed ID's. Apart from being recorded in the tag, they are printed on a sticker with the RFID tag.
– Once an item holding the tag arrives in a system, it is checked which of the ID's from the sticker is still unused in the system. One of them is chosen. Then the printed code of the second ID is marked or made unreadable while its electronic version is erased releasing some of tag's memory.

– The chosen ID is also recorded in the system in a database holding all ID's of objects in the system.

6 Distribution with DHT

The idea of the protocol presented and analyzed in this paper can be used in other application areas. We mention an example of this kind.

In distributed memory systems based on addressing with distributed hash tables (DHT) it is usually the case that demand for data/services is very nonuniform. There are popular and unpopular topics. So, given a distribution of an address space, we wish to allocate the hot topics so that no server is responsible for more than one of them. It has to be done without changing the basic principle that the location of a given service/data is available at address indicated by the hash value. The problem is that it is not known at the moment of designing the system which topics will become popular. So in particular we cannot adjust the hash function to the list of most popular topics.

A simple solution is to use two-choices random assignment principle. Location of a topic X is indicated by two values $H(X,1)$ and $H(X,2)$, where H is an appropriate hash function. In a regular case, we assign to X the location indicated by $H(X,1)$. In case of a conflict one can move X to the location indicated by $H(X,2)$. According to our results, this allows to avoid conflicts for much larger lists of hot topics. At the same time, accessing data is not harder than before. Two hops necessary to reach the right location at some situations are compensated by a better response time.

Note that we are not focusing here on load balancing of topics (their number is usually much higher than the number of servers), but allocating at most one hot topic to each server. This is motivated by the fact that from efficiency point of view it is more important to avoid collisions between hot topics than to equalize the number of topics allocated to the servers.

7 Analysis of k-Choice Protocol

The estimation of probabilities related to the birthday paradox has a big literature. We use the methodology from [11] to obtain precise estimations. First let us recall the following well known facts:

(i) $1 - x \le e^{-x}$,

(ii) $\sum_{j=1}^{k} j = \frac{k(k+1)}{2}$,

(iii) $\sum_{j=1}^{k} j^2 = \frac{k(k+1)(2k+1)}{6}$,

(iv) $\sum_{j=1}^{k} j^b \le \frac{(k+1)^{b+1}}{b+1}$,

(v) $\ln(1-x) = -\sum_{j=1}^{\infty} \frac{1}{j} x^j$.

Analysis of Random Assignment Process. Let us fix a number N and let S_t denote the event that there was no collision in the first t steps of putting balls into N bins of the random assignment process. Notice that $\Pr[S_1] = 1$ and that $\Pr[S_{t+1}|S_t] = 1 - \frac{t}{N}$. Obviously $S_{t+1} \subseteq S_t$, therefore $\Pr[S_{t+1}] = \prod_{a=1}^{t}(1 - \frac{a}{N})$. Recall that B_N denotes the first moment when the collision occurs in this process. Then $\Pr[B_N \le t] = 1 - \Pr[S_t]$ and we have

$$\Pr[B_N \le t+1] = 1 - \prod_{a=0}^{t}\left(1 - \frac{a}{N}\right) \quad . \tag{7}$$

Lemma 1. *If* $0 \le t < N$, *then*

$$e^{-\frac{t(t+1)}{2N}} - \frac{(t+1)^3}{6N^2(1-\frac{t+1}{N})} \le \prod_{a=1}^{t}(1 - \frac{a}{N}) \le e^{-\frac{t(t+1)}{2N}} \quad . \tag{8}$$

Proof. Let $a_{N,t} = \prod_{a=1}^{t}(1 - \frac{a}{N})$. Then

$$-\ln a_{N,t} = \sum_{a=1}^{t}\sum_{b=1}^{\infty}\frac{1}{b}\left(\frac{a}{N}\right)^b = \sum_{b=1}^{\infty}\left(\frac{1}{b}\cdot\frac{1}{N^b}\sum_{a=1}^{t}a^b\right) \tag{9}$$

$$= \frac{t(t+1)}{2N} + \sum_{b=2}^{\infty}\left(\frac{1}{b}\cdot\frac{1}{N^b}\sum_{a=1}^{t}a^b\right) \quad . \tag{10}$$

Therefore $\ln a_{N,t} \le -\frac{t(t+1)}{2N}$ and $a_{N,t} \le \exp(-\frac{t(t+1)}{2N})$. Next, we have

$$-\ln a_{N,t} \le \frac{t(t+1)}{2N} + \sum_{b=2}^{\infty}\left(\frac{1}{b}\cdot\frac{1}{N^b}\cdot\frac{(t+1)^{b+1}}{b+1}\right) \tag{11}$$

$$\le \frac{t(t+1)}{2N} + \frac{t+1}{6}\sum_{b=2}^{\infty}\left(\frac{t+1}{N}\right)^b \tag{12}$$

$$= \frac{t(t+1)}{2N} + \frac{t+1}{6}\cdot\left(\frac{t+1}{N}\right)^2\cdot\frac{1}{1-\frac{t+1}{N}} \quad . \tag{13}$$

Therefore

$$a_{N,t} \ge \exp\left(-\frac{t(t+1)}{2N}\right)\cdot\exp\left(\frac{(t+1)^3}{6N^2(1-\frac{t+1}{N})}\right) \tag{14}$$

$$\ge \exp\left(-\frac{t(t+1)}{2N}\right)\cdot\left(1 - \frac{(t+1)^3}{6N^2(1-\frac{t+1}{N})}\right) \tag{15}$$

$$\ge \exp\left(-\frac{t(t+1)}{2N}\right) - \frac{(t+1)^3}{6N^2(1-\frac{t+1}{N})} \quad . \tag{16}$$

This concludes the proof. □

Theorem 1. *Let B_N denotes the the first moment that a collision occurs in the random assignment process with N bins. If $N \geq 20$ and $t \leq \sqrt{\frac{\pi N}{2}}$, then*

$$1 - e^{\frac{t(t+1)}{2N}} \leq \Pr[B_N \leq t+1] \leq 1 - e^{\frac{t(t+1)}{2N}} + \frac{1}{\sqrt{N}} \ . \tag{17}$$

Proof. From equation (7) and Lemma 1 we get

$$1 - e^{\frac{t(t+1)}{2N}} \leq \Pr[B_N \leq t+1] \leq 1 - e^{\frac{t(t+1)}{2N}} + \frac{(t+1)^3}{6N^2(1-\frac{t+1}{N})} \ . \tag{18}$$

The function $f_N(x) = \frac{(x+1)^3}{6N^2(1-\frac{x+1}{N})}$ is increasing on interval $[0, N-1)$, so if $t \leq \sqrt{\frac{\pi N}{2}}$, then $\frac{(t+1)^3}{6N^2(1-\frac{t+1}{N})} \leq f_N(\sqrt{\frac{\pi N}{2}})$. Moreover, the sequence $a_N{=}f_N(\sqrt{\frac{\pi N}{2}})\sqrt{N}$ is decreasing for $N > 4$ and $a_{20} \approx 0.801692$. So the theorem follows. □

Let us recall that $\mathrm{E}[B_N] \approx \sqrt{\frac{\pi N}{2}}$. Note that from the Theorem 1 we get $\Pr[B_N \leq \sqrt{\frac{\pi N}{2}} + 1] \approx 1 - e^{-\pi/4} \approx 0.544062$.

Analysis of Two-choices Random Assignment Process. Let us fix a number N and let Z_t denote the event that there was no collision in the first t steps of putting balls into N bins during two-choice random assignment process. Notice that $\Pr[Z_1] = 1$ and that $\Pr[Z_{t+1}|Z_t] = 1 - \left(\frac{t}{N}\right)^2$. Obviously $Z_{t+1} \subseteq Z_t$, therefore $Z_{t+1} = \prod_{a=1}^{t}(1 - \left(\frac{a}{N}\right)^2)$. Let C_N denote the first moment when the collision occurs during this process. Then $\Pr[C_N \leq t] = 1 - \Pr[Z_t]$ and we have

$$\Pr[C_N \leq t+1] = 1 - \prod_{a=1}^{t}\left(1 - \left(\frac{a}{N}\right)^2\right) \ . \tag{19}$$

Lemma 2. *If $0 \leq t < N$, then*

$$e^{-\frac{t(t+1)(2t+1)}{6N^2}} - \frac{(t+1)^5}{10N^4(1-(\frac{t+1}{N})^2)} \leq \prod_{a=1}^{t}\left(1 - \left(\frac{a}{N}\right)^2\right) \leq e^{-\frac{t(t+1)(2t+1)}{6N^2}} \ . \tag{20}$$

Proof. Let $a_{N,t} = \prod_{a=1}^{t}(1 - \left(\frac{a}{N}\right)^2)$. Then

$$-\ln a_{N,t} = \sum_{a=1}^{t}\sum_{b=1}^{\infty}\frac{1}{b}\left(\frac{a}{N}\right)^{2b} \tag{21}$$

$$= \sum_{b=1}^{\infty}\left(\frac{1}{b}\cdot\frac{1}{N^{2b}}\sum_{a=1}^{t}a^{2b}\right) \tag{22}$$

$$= \frac{t(t+1)(2t+1)}{6N^2} + \sum_{b=2}^{\infty}\left(\frac{1}{b}\cdot\frac{1}{N^{2b}}\sum_{a=1}^{t}a^{2b}\right) \ . \tag{23}$$

Therefore

$$\ln a_{N,t} \leq -\frac{t(t+1)(2t+1)}{6N^2} \quad \text{and} \quad a_{N,t} \leq \exp(-\frac{t(t+1)(2t+1)}{6N^2}) \ . \tag{24}$$

Next, we have

$$-\ln a_{N,t} \leq \frac{t(t+1)(2t+1)}{6N^2} + \sum_{b=2}^{\infty}\left(\frac{1}{b}\cdot\frac{1}{N^{2b}}\cdot\frac{(t+1)^{2b+1}}{2b+1}\right) \tag{25}$$

$$\leq \frac{t(t+1)(2t+1)}{6N^2} + \frac{t+1}{10}\sum_{b=2}^{\infty}\left(\frac{t+1}{N}\right)^{2b} \tag{26}$$

$$= \frac{t(t+1)(2t+1)}{6N^2} + \frac{t+1}{10}\left(\frac{t+1}{N}\right)^4\frac{1}{1-(\frac{t+1}{N})^2} \ . \tag{27}$$

Therefore

$$a_{N,t} \geq \exp\left(-\frac{t(t+1)(2t+1)}{6N^2}\right)\exp\left(\frac{(t+1)^5}{10N^4(1-(\frac{t+1}{N})^2)}\right) \tag{28}$$

$$\geq \exp\left(-\frac{t(t+1)(2t+1)}{6N^2}\right)\left(1-\frac{(t+1)^5}{10N^4(1-(\frac{t+1}{N}))^2}\right) \tag{29}$$

$$\geq \exp\left(-\frac{t(t+1)(2t+1)}{6N^2}\right) - \frac{(t+1)^5}{10N^4(1-(\frac{t+1}{N})^2)} \ . \tag{30}$$

□

Theorem 2. *Let C_N denote the first moment that a collision occurs while putting balls into N bins during two-choices random assignment process. If $N \geq 5$ and $t < \sqrt[3]{3}\cdot\Gamma(\frac{4}{3})\cdot N^{\frac{2}{3}}$, then*

$$1-e^{\frac{t(t+1)(2t+1)}{6N^2}} \leq \Pr[C_N \leq t+1] \leq 1-e^{\frac{t(t+1)(2t+1)}{6N^2}} + \frac{1}{N^{\frac{2}{3}}} \ . \tag{31}$$

Proof. From equation (19) and Lemma 2 we get

$$1-e^{\frac{t(t+1)(2t+1)}{6N^2}} \leq \Pr[C_N \leq t+1] \leq 1-e^{\frac{t(t+1)(2t+1)}{6N^2}} + \frac{(t+1)^5}{10N^4(1-(\frac{t+1}{N})^2)} \ . \tag{32}$$

The function $f_N(x) = \frac{(x+1)^5}{10N^4(1-(\frac{x+1}{N})^2)}$ is increasing on interval $[0, N-1)$, so if $t \leq \sqrt[3]{3}\cdot\Gamma(\frac{4}{3})\cdot N^{\frac{2}{3}}$, then

$$\frac{(t+1)^5}{10N^4(1-(\frac{t+1}{N})^2)} \leq f_N(\sqrt[3]{3}\cdot\Gamma(\tfrac{4}{3})\cdot N^{\frac{2}{3}}) \ . \tag{33}$$

Moreover, the sequence $a_N = f_N(\sqrt[3]{3}\cdot\Gamma(\frac{4}{3})\cdot N^{\frac{2}{3}})N^{\frac{2}{3}}$ is decreasing for $N > 2$ and $a_5 \approx 0.818813$. So the theorem follows. □

Notice that the approximation from Theorem 2 is sharper than the approximation obtained by Theorem 1. Let us recall that $\mathrm{E}[C_N] \approx \sqrt[3]{3}\cdot\Gamma(\frac{4}{3})\cdot N^{\frac{2}{3}}$. By Theorem 2 we get $\Pr[C_N < \sqrt[3]{3}\cdot\Gamma(\frac{4}{3})\cdot N^{\frac{2}{3}}+1] \approx 1-e^{-\Gamma(\frac{4}{3})^2} \approx 0.509374$.

References

1. Azar, Y., Broder, A., Karlin, A., Upfal, E.: Balanced Allocations. SIAM J. Comput. 29(1), 180–200 (1999)
2. Azar, Y., Broder, A., Karlin, A.: Online Load Balancing. In: 33rd Annual Symposium on Foundations of Computer Science, pp. 218–225. IEEE Computer Society, Los Alamitos (1992)
3. Berenbrink, P., Czumaj, A., Steger, A., Vöcking, B.: Balanced Allocations: The Heavily Loaded Case. SIAM J. Computing 35(6), 1350–1385 (2006)
4. Bloom, D.: A Birthday Problem. American Mathematical Monthly 80, 1141–1142 (1973)
5. Cichoń, J., Klonowski, M., Kutyłowski, M.: Privacy Protection for Dynamic Systems Based on RFID Tags. In: 4th IEEE International Workshop on Pervasive Computing and Communication Security, Proceedings of PERCOM 2007 Workshops, pp. 235–240. IEEE Computer Society, Los Alamitos (2007)
6. Flajolet, P., Sedgewick, R.: Analytic Combinatorics, `http://algo.inria.fr/flajolet/Publications/books.html`
7. Mitzenmacher, M.: The Power of Two Choices in Randomized Load Balancing. IEEE Trans. Parallel Distrib. Syst. 12(10), 1094–1104 (2001)
8. Klamkin, M.S., Newman, D.J.: Extensions of the Birthday Surprise. J. Combin. Theory 3, 279–282 (1967)
9. Ren, K., Lou, W.: Privacy-enhanced, Attack-resilient Access Control in Pervasive Computing Environments with Optional Context Authentication Capability. Mobile Networks and Applications (MONET) 12, 79–92 (2007)
10. Suzuki, K., Tonien, D., Kurosawa, K., Toyota, K.: Birthday Paradox for Multi-Collisions. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 29–40. Springer, Heidelberg (2006)
11. Tsaban, B.: Bernoulli Numbers and the Probability of a Birthday Surprise. arXiv:math.NA/0304028 v3, November 9 (2004)
12. PRIME - Privacy and Identity Management for Europe, `https://www.prime-project.eu/`
13. RAPID (Roadmap for Advanced research in Privacy and IDentity management), `ec.europa.eu/information_society/activities/egovernment_research/doc/rapid_roadmap.doc`

Deniable Authentication on the Internet
(Extended Abstract)

Shaoquan Jiang

Center for Information Security and Cryptography
University of Calgary
sqjiang@ucalgary.ca

Abstract. Deniable authentication is a technique that allows one party to send messages to another while the latter can not prove to a third party the fact of communication. In this paper, we formalize a natural notion of deniable security and naturally extend the basic authenticator theorem by Bellare et al. [1] to the setting of deniable authentication. Of independent interest, this extension is achieved by defining a deniable MT-authenticator via a game. This game is essentially borrowed from the notion of universal composition [6] although we do not assume any result or background about it. Then we construct a 3-round deniable MT-authenticator. Finally, as our application, we obtain a key exchange protocol that is deniably secure in the real world.

Keywords: Deniability, Authentication, Protocol, Key Exchange.

1 Introduction

Authentication is a communication process, in which a receiver is assured of the authenticity of the peer identity and the authenticity of the incoming messages. This property can be guaranteed by means of a signature. Since a secure signature is unforgeable and publicly verifiable, it in other words means undeniability. This undeniability is not always desirable. For example, when you do an Internet shopping, you do not want your shopping privacy to be transferred to a third party. In this paper, we will investigate techniques for achieving deniable authentication.

1.1 Related Work

Deniable authentication was first considered in [12]. Earlier concepts occurred in [9]. Since deniability essentially requires that whatever computable through interactions is computable by adversary himself, a natural tool to achieve it is zero knowledge [16]. However, it is known that under a general complexity assumption any black-box concurrent zero-knowledge has a round complexity $\tilde{\omega}(\log \kappa)$ [26,20,25]. This implies that the practical deniability from it is almost impossible as most of the applications require concurrency. To overcome this barrier, [15,13,14,19,23,10] relaxed the concurrency with a locally timing constraint.

However, timing constraint is not satisfiable as it artificially increases the delay. An alternative approach is to adopt a non-standard complexity assumption. Di Rainmondo et al. [11], based on an assumption of plaintext awareness [2,8], showed that SKEME [21] is deniably secure. But the assumption is very strong. Another alternative is to adopt a common reference string (CRS) model. In this setting, efficient concurrent zero-knowledge does exist [7,17]. Unfortunately, it has been pointed out in the literature (e.g., Pass [24]) that deniability obtained in this way is not satisfiable as the simulator usually owns a secret of CRS while it is impossible to a real adversary. Similarly, an original random oracle [4] based solution is not satisfiable, either. Pass [24] defined a revised random oracle model (we call it an uncontrollable random oracle (uRO) model), which is different from the original one in that the output of the oracle is maintained by an uncontrollable third party (instead of a simulator) although the simulator can still view the input-output pair of each query. Deniability under this model is practical since whatever computable by the simulator is computable by the adversary himself. However, authentication is not a research concern in [24]. As a summary, known research in deniable authentication is still not very satisfiable.

1.2 Contribution

In this paper, we first present an authentication model [1,5] with featuring a concurrent computation model [22]. Under this, we formalize a notion of *deniable security* and naturally extend the authenticator theorem in [1] to the setting of deniable computation. This extension is essentially achieved by deploying a universal composition technique. This strategy is of independent interest. Then we construct a provably deniable MT-authenticator based on uncontrollable random oracle [24]. Finally, as our application, we obtain a key exchange protocols that is deniably UM-secure.

2 Model

Bellare *et al.* [1,5] formalized two models: unauthenticated-link model (UM) and authenticated-link model (AM) for cryptographic protocols. This model is very useful in a modular design of UM-secure protocols. On the other hand, a concurrent composition model (e.g., [22]) is convenient in analysis of protocols. We now present UM/AM models with featuring [22].

Assume $P_1, \cdots, P_n$ are n-parties. π is an arbitrary protocol. The execution of π is modeled as follows. Each party is regarded as a polynomial time interactive Turning machine. Initially, P_i is invoked with input, identity and random input. Then he waits for an activation. P_i can be activated by incoming messages from other parties and/or external input. Once activated, P_i follows the specification of π by computing

$$\begin{aligned}&\pi(\text{input, internal state, incoming message})\\&=(\text{new state, outgoing messages, output}).\end{aligned}$$

Initial internal state is the party's input, identity and random input. After each activation, the internal state is updated by a new state. Each activation could generate outgoing messages (to other parties). It may also generate a local output and label the sensitive part as 'secret'. Each P_i could concurrently run many copies of π. A copy is called a *session.* Each session has a session ID. The only requirement for a session ID is its uniqueness in P_i. The input for different activations of each session in P_i might be different. It is generated by a probabilistic polynomial time algorithm Φ_i. For the ℓth activation of the jth session, the input is $x_{\ell,j} = \Phi_i(\ell, j, x_i, \mathsf{hist})$, where x_i is the initial input to Φ_i and hist is the output history of all copies of π in P_i. Note $x_{\ell,j}$ could be empty. In order of delivery (also for security), *each message sent into the channel is assumed to contain* (*sender, sender session ID, receiver, receiver session ID*). In addition, we implicitly assume that π has been ideally initialized by a function I : for $r \leftarrow \{0,1\}^\kappa$, $I(r) = I(r)_0, I(r)_1, \cdots, I(r)_n$, where κ is the security parameter, $I(r)_0$ is the public information known to all participants, and $I(r)_i$ is the secret key for P_i.

2.1 Unauthenticated-Link Model

Roughly speaking, the unauthenticated-link model is a model for the concurrent execution of a protocol where a malicious adversary presents. In this model, the scheduling of events is completely determined by adversary $\mathcal{U}$. Such a scheduling consists of a sequence of activations to parties. He can activate any party P_i with an arbitrary incoming message. He can also invoke P_i to start a new session. In both cases, it is assumed that Φ_i has already supplied the input (if any). He can also delete, block, modify and insert any message over the channel. Once a party completes an activation, the outgoing message and the local output (if any), except the parts labeled as 'secret', are available to $\mathcal{U}$. $\mathcal{U}$ can corrupt a party at any time. When one party gets corrupted, all sessions' internal states and the secret key within this party are available to $\mathcal{U}$. A special note 'corrupted' is appended to the output of this party. It will not produce an output any more. In addition, his future action is fully taken by $\mathcal{U}$. $\mathcal{U}$ can also corrupt a particular session in P_i. In this case, he obtains the current internal state for this session. A special note of corruption is appended to this session's output. Later, it will not produce an output any more. The future execution of this session is fully taken by $\mathcal{U}$. We assume session corruption is possible only if the session has started. This reflects the practical concern where a session is attacked only if the attacker sees the session's activity.

Assume the protocol is initialized by a trusted third party $\mathbb{T}$. Specifically, before the protocol starts, $\mathbb{T}$ takes $s \leftarrow \{0,1\}^\kappa$ and executes the initialization function $I(s) = \{I(s)_i\}_{i=0}^n$. Then he provides $I(s)_i$ to party P_i as his secret. The global public information is $I(s)_0$. At the end of protocol execution, $\mathbb{T}$ outputs

$$I(s)_0 \cup \{I(s)_i \mid P_i \text{ corrupted}, 1 \leq i \leq n\}.$$

The final output of a party is defined to be the concatenation of his output history from all sessions. Let $\boldsymbol{x} = (x_1, \cdots, x_n)$, where x_i is the initial input for

Φ_i. Let $\boldsymbol{r} = (r_0^0, r_0^1, r_1^0, r_1^1, \cdots, r_n^0, r_n^1)$ be the random input, where r_0^1 for $\mathcal{U}$, r_0^0 for $\mathbb{T}$, r_i^0 for P_i and r_i^1 for Φ_i. Let $\boldsymbol{\Phi} = (\Phi_1, \cdots, \Phi_n)$. We use $\mathrm{Adv}_{\pi,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x}, \boldsymbol{r})$ to denote the output of $\mathcal{U}$, and use $\mathrm{UnAuth}_{\pi,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x}, \boldsymbol{r})_i$ to denote the output of P_i. $\mathrm{UnAuth}_{\pi,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x}, \boldsymbol{r})_0$ denotes the output of $\mathbb{T}$. Let

$$\mathrm{UnAuth}_{\pi,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x}, \boldsymbol{r}) = \mathrm{Adv}_{\pi,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x}, \boldsymbol{r}),\ \mathrm{UnAuth}_{\pi,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x}, \boldsymbol{r})_0,\ \cdots,\ \mathrm{UnAuth}_{\pi,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x}, \boldsymbol{r})_n.$$

Let $\mathrm{UnAuth}_{\pi,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x})$ be the random variable describing $\mathrm{UnAuth}_{\pi,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x}, \boldsymbol{r})$. Our inclusion of the output of $\mathbb{T}$ in the global output is for defining deniable security later (See Section 2.3).

2.2 Authenticated-Link Model

Authenticated-link model is similar to unauthenticated-link model, except that any outgoing message sent by an uncorrupted party (if not blocked) will be faithfully delivered.

Authentication Functionality. The following functionality (See Figure 1) is to formalize the authenticated channel. Unlike [6], here we directly consider the multiple sessions of the functionality. This seems helpful to simplify the presentation. The action of $\tilde{P}_i$ is defined as follows. Whenever upon input $(\mathsf{send}, \tilde{P}_j, m)$, copy it to the input tape of $\hat{\mathcal{F}}$; whenever receiving $(\mathsf{receiv}, id, \tilde{P}_\ell, \tilde{P}_i, m)$ from $\hat{\mathcal{F}}$, directly output it. The procedure (in Figure 1) for $\tilde{P}_i$ to send m to $\tilde{P}_j$ is called a session for $\tilde{P}_i, \tilde{P}_j$ and $\hat{\mathcal{F}}$, respectively. We simply say a $\hat{\mathcal{F}}$-session. Assume an uncorrupted sender $\tilde{P}_i$ never sends a message m twice. This convention allows us to easily identify a replay attack. Thus, a session for an uncorrupted sender can be identified by m itself. A session in $\hat{\mathcal{F}}$ or in a receiver $\tilde{P}_j$ can be identified by id (**for simplicity, we assume $id \leftarrow \{0,1\}^\kappa$ never repeats in this paper**). However, when a party $\tilde{P}_i$ is corrupted, our functionality allows $\hat{\mathcal{A}}$, in the name of $\tilde{P}_i$, to send any m to $\tilde{P}_j$ (through $\hat{\mathcal{F}}$). This reflects the concern when one party is adversarial, cryptographic authentication techniques can not prevent it from flooding a receiver. Further remarks follow. First, message exchanges between

$\hat{\mathcal{F}}$ runs with parties $\tilde{P}_1, \cdots, \tilde{P}_n$ and adversary $\hat{\mathcal{A}}$

- Whenever receiving $(\mathsf{send}, \tilde{P}_j, m)$ from $\tilde{P}_i$, take $id \leftarrow \{0,1\}^\kappa$, send $(id, \tilde{P}_i, \tilde{P}_j, m)$ to $\hat{\mathcal{A}}$ and wait for a bit c from $\hat{\mathcal{A}}$. After $\hat{\mathcal{A}}$ computes c (it could take arbitrary length of time), it sends (c, id) back to $\hat{\mathcal{F}}$.
- After receiving (c, id) from $\hat{\mathcal{A}}$, if $c = 1$ and if $(id, \tilde{P}_i, \tilde{P}_j, m)$ for some $(\tilde{P}_i, \tilde{P}_j, m)$ has been sent to $\hat{\mathcal{A}}$ but $(*, id)$ was not received before, send $(\mathsf{receiv}, id, \tilde{P}_i, \tilde{P}_j, m)$ to $\tilde{P}_j$. In any case, mark id as '$\mathsf{answered}$'.

Fig. 1. Ideal functionality $\hat{\mathcal{F}}$ for authentication

$\tilde{P}_\ell$ and $\hat{\mathcal{F}}$ are ideally secure and immediate (i.e., no adversary plays between). Second, $\hat{\mathcal{A}}$ can have an arbitrary delay to return (c, id). This reflects the nature of an asynchronous network. $c = 1$ means the message m is not blocked. Third, $\hat{\mathcal{A}}$ can corrupt any $\tilde{P}_i$ or a session in it. If a session is corrupted, the session state (i.e., input or output for this session) is provided to $\hat{\mathcal{A}}$ and a note 'corrupted' is appended in his output. The future action is fully taken by $\hat{\mathcal{A}}$. If $\tilde{P}_i$ is corrupted, all the sessions in it are corrupted by $\hat{\mathcal{A}}$. In addition, the future action of $\tilde{P}_i$ is taken by $\hat{\mathcal{A}}$. Especially, $\hat{\mathcal{A}}$ can represent $\tilde{P}_i$ to send any message m (including a previously sent message) through $\hat{\mathcal{F}}$ to a party $\tilde{P}_j$.

Authenticated-Link model. We are now ready to present the authenticated-link model (AM). Let $P_1, \cdots, P_n$ be n parties for executing π. AM follows the order in UM, except messages are sent/received through $\hat{\mathcal{F}}$ and the adversarial behavior is restricted correspondingly. Formally,

- When P_i needs to send a message m to P_j, it invokes $\tilde{P}_i$ in Figure 1 with input $(\mathsf{send}, \tilde{P}_j, m)$ to do this.
- All incoming messages for a party P_j are received through reading the output of $\tilde{P}_j$.
- Upon output $(\mathsf{receiv}, id, \tilde{P}_i, \tilde{P}_j, m)$ of $\tilde{P}_j$, P_j executes π with incoming message m.

The action of an adversary $\mathcal{A}$ is as follows.

- When P_i invokes $\tilde{P}_i$ with input $(\mathsf{send}, \tilde{P}_j, m)$, $\mathcal{A}$ plays the role of $\hat{A}$ in Figure 1 to participate.
- When a session sid in P_i is corrupted, it is assumed that all the $\hat{\mathcal{F}}$-sessions of $\tilde{P}_i$ that send/receive messages for session sid are corrupted. As a result, $\mathcal{A}$ will receive the internal state of P_i in π including states from these $\hat{\mathcal{F}}$-sessions. Finally, a note 'corrupted' appears in the output of session sid. Later it is no longer active. Its action will be fully taken by $\mathcal{A}$.
- When a party P_i is corrupted, all sessions in P_i are corrupted. As a result, the secret key $I(r)_i$ and all internal states are available to $\mathcal{A}$. The future action of P_i is taken by $\mathcal{A}$.

The protocol is initialized by a third party $\mathbb{T}$. Specifically, before the protocol starts, $\mathbb{T}$ takes $s \leftarrow \{0,1\}^\kappa$ and executes the initialization function $I(s) = \{I(s)_i\}_{i=0}^n$. Then he provides $I(s)_i$ to party P_i. The global public information is $I(s)_0$ for all parties. In addition, $\mathbb{T}$ can execute an extra function $I'(s') = \{I'(s')_i\}_{i=0}^n$ for $s' \leftarrow \{0,1\}^\kappa$. Initially, $I'(s')_0$ and $I(s)_0$ will be provided to $\mathcal{A}$. Later whenever P_i is corrupted, $I(s)_i$ and $I'(s')_i$ will be provided to $\mathcal{A}$. At the end of protocol execution, $\mathbb{T}$ outputs

$$\{I(s)_0, I(s')_0\} \cup \{I(s)_i, I'(s')_i \mid P_i \text{ corrupted}, 1 \le i \le n\}.$$

Note our treatment for introducing the extra $I'(s')$ is for defining deniable security (See Section 2.3), where $I'(s')$ will be the initialization function for the protocol realizing $\hat{\mathcal{F}}$. As for UM, let $\boldsymbol{x} = (x_1, \cdots, x_n)$, where x_i is the initial input for $\varPhi_i$. Let $\boldsymbol{r} = (r_f, r_0^0, r_0^1, r_1^0, r_1^1, \cdots, r_n^0, r_n^1)$ be the random input, where r_f is for $\hat{\mathcal{F}}$, r_0^0 is for $\mathbb{T}$, r_0^1 is for $\mathcal{A}$, r_i^0 is for P_i, r_i^1 is for $\varPhi_i$. Analogous to UM, we can define the adversary output $\mathrm{Adv}^{\hat{\mathcal{F}}}_{\pi,\mathcal{A},\boldsymbol{\varPhi},I'}(\boldsymbol{x}, \boldsymbol{r})$, the output of $\mathbb{T}$ $\mathrm{Auth}^{\hat{\mathcal{F}}}_{\pi,\mathcal{A},\boldsymbol{\varPhi},I'}(\boldsymbol{x}, \boldsymbol{r})_0$, the output of party P_i $\mathrm{Auth}^{\hat{\mathcal{F}}}_{\pi,\mathcal{A},\boldsymbol{\varPhi},I'}(\boldsymbol{x}, \boldsymbol{r})_i$, the global output $\mathrm{Auth}^{\hat{\mathcal{F}}}_{\pi,\mathcal{A},\boldsymbol{\varPhi},I'}(\boldsymbol{x}, \boldsymbol{r})$ and the corresponding random variable $\mathrm{Auth}^{\hat{\mathcal{F}}}_{\pi,\mathcal{A},\boldsymbol{\varPhi},I'}(\boldsymbol{x})$. Note in the UM case, I' is empty. Also since I is already implicitly included in π, there is no need to explicitly label it on the above variables.

2.3 Deniable Security

For a protocol π to be deniably secure, we essentially hope whatever computable by an attacker through interactions can be computed by himself alone. There are two factors to prevent a simulator from executing such a deniable computation. First, $\boldsymbol{x}$ could be unknown. However, if $\boldsymbol{x}$ is private, it is hard to see what type of deniability can be formalized. To simplify the problem, we only consider functionalities, where $\boldsymbol{x}$ is not a secret. For instance, in key exchange, x_i is a request to carry out key exchange. One may wonder why not just define the security model such that the adversary additionally plays the role of $\boldsymbol{\varPhi}$ to supply protocol inputs. We stress that for some functionalities such as oblivious transfer and e-voting, the inputs are secret. Thus, the adversary is not allowed to know them unless the underlying party gets corrupted. The perfect version of security in a multi-party computation is formalized as an ideal process, where the parties hands their inputs to a trusted party who feeds back an output for each party by computing the functionality himself. Details can be found in the literature (e.g., [22]). In our setting, input $\boldsymbol{x}$ is not a secret. It follows that this formulation can also be regarded as an ideal version of deniable security. Again following the multi-party tradition, the deniable security of a protocol π can be defined as requiring an ideal process simulator to simulate a real system such that the global output of the ideal process is indistinguishable to that in the real execution. However, the second problem comes. Recall that in π, each party P_i receives a secret key $I(r)_i$ from the setup function I and an adversary is unable to access an uncorrupted $I(r)_i$. Thus, in order to be deniable, a simulator should not be allowed to know uncorrupted $I(r)_i$ either. To do this, we let a third party $\mathbb{T}$ to take $I(r) = \{I(r)_i\}$ for $r \leftarrow \{0,1\}^\kappa$ and provide $I(r)_0$ to ideal process simulator. Later, $I(r)_i$ is provided to the latter if and only if P_i is corrupted. At the end of the simulation, $\mathbb{T}$ output $I(r)_0$ and all corrupted $I(r)_i$. The global output in the ideal process is expanded with the output of $\mathbb{T}$. If π is a $\hat{\mathcal{F}}$-hybrid protocol, then I used by $\mathbb{T}$ in the above is replaced by (I, I') for an arbitrary given extra I'. We use $\mathrm{IDEAL}_{\mathcal{G},\mathcal{S},\boldsymbol{\varPhi},(I,I')}$ to denote the global output in the ideal process for computing functionality $\mathcal{G}$, where an adversary is $\mathcal{S}$, the input function is $\boldsymbol{\varPhi}$, the initialization function is I and the extra initialization function is I'. This global output is the concatenation of output by $\mathbb{T}, \{P_i\}$ and adversary (or simulator) $\mathcal{S}$. We use $\mathrm{REAL}_{\pi,\mathcal{O},\boldsymbol{\varPhi},I'}(\boldsymbol{x})$ to denote the global output in the real process (i.e., in executing π), where $\mathcal{O}$ is the adversary. When π is an $\hat{\mathcal{F}}$-hybrid protocol, this global output is $\text{Auth}^{\hat{\mathcal{F}}}_{\pi,\mathcal{O},\boldsymbol{\Phi},I'}(\boldsymbol{x})$. When π is a protocol in the UM, this global output is $\text{UnAuth}_{\pi,\mathcal{O},\boldsymbol{\Phi}}(\boldsymbol{x})$, where I' is enforced to be empty.

Definition 1. *Let π be a protocol with initialization function I for computing a functionality $\mathcal{G}$. Let I' be arbitrary extra initialization function (I' is empty if π is a* UM *protocol). π is said to be* deniably secure *if for any feasible $\boldsymbol{x}, I'$ and any PPT adversary $\mathcal{O}$ against π there exists a PPT adversary $\mathcal{S}$ against the ideal process such that*

$$\text{IDEAL}_{\mathcal{G},\mathcal{S},\boldsymbol{\Phi},(I,I')}(\boldsymbol{x}) \overset{c}{\equiv} \text{REAL}_{\pi,\mathcal{O},\boldsymbol{\Phi},I'}(\boldsymbol{x}). \tag{1}$$

3 Deniable Authentication Theorem

Essentially, we wish to construct a protocol ρ to realize $\hat{\mathcal{F}}$. Then for any protocol π in the $\hat{\mathcal{F}}$-hybrid model (i.e., AM), we replace $\hat{\mathcal{F}}$ by ρ and hope the composed protocol (denoted by π^ρ) is secure. Bellare *et al.* [1] proposed a notion of MT-authenticator, which is a realization of $\hat{\mathcal{F}}$ in the UM. They confirmed the above result when ρ is a MT-authenticator. However, here we are mainly interested in finding a ρ that does not introduce additional undeniability. Their MT-authenticator does not guarantee this since the simulator there initializes the MT-authenticator himself. In order to be deniable, a simulator should not be allowed to know the secret key (say, $I_\rho(r)_i$) of party in ρ unless he is corrupted. To achieve this, we introduce a third party $\mathbb{T}$ to generate and maintain parties' private keys. Especially, a simulator is allowed to access $I_\rho(r)_i$ if and only if party i is corrupted. This is what we have done in the authenticated-link model. We formalize this intuition into the following notion of deniable authentication.

Definition 2. *Assume ρ is a protocol for computing $\hat{\mathcal{F}}$. Let π be any protocol in the $\hat{\mathcal{F}}$-hybrid model. Let I_ρ be the initialization function for ρ. π^ρ is said to be* deniably authenticated *if for any adversary $\mathcal{U}$ against π^ρ and any $\boldsymbol{x}$, there exists an adversary $\mathcal{A}$ against π such that*

$$\text{Auth}^{\hat{\mathcal{F}}}_{\pi,\mathcal{A},\boldsymbol{\Phi},I_\rho}(\boldsymbol{x}) \overset{c}{\equiv} \text{UnAuth}_{\pi^\rho,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x}). \tag{2}$$

Since MT-authenticator in [1] provides an authenticated transformation for any AM protocol, a question is whether there exists a natural property for ρ such that as long as ρ satisfies it π^ρ will be deniably authenticated for any π. In the following, we introduce a notion of deniable MT-authenticator. We show that given a deniable MT-authenticator ρ, for any π in the $\hat{\mathcal{F}}$-hybrid model, π^ρ is deniably authenticated. We define this notion through two protocol games. These game are essentially borrowed from the notion of universal composition [6] although we do not need any result or background about it.

The first game is denoted by G_0. Assume $\tilde{P}_1, \cdots, \tilde{P}_n$ is running ρ in the UM with a dummy adversary $\mathcal{A}_0$. $\mathcal{Z}$ is a PPT interactive Turing machine. Assume $I_\rho(r) = \{I_\rho(r)_i\}_{i=0}^n$ is the initialization function for ρ. Initially, $\mathcal{Z}$ will receive the public information $I_\rho(r)_0$. On the one hand, $\mathcal{Z}$ plays the role of $\boldsymbol{\Phi}$ to supply inputs for $\tilde{P}_i$. On the other hand, $\mathcal{Z}$ can supply $\mathcal{A}_0$ with instructions obeying the UM rules at any time. These instructions include (1) starting a session at some $\tilde{P}_i$ or (2) activating a session with a specific incoming message or (3) corrupting a session or a party. Upon an instruction, $\mathcal{A}_0$ follows it faithfully. In case of (1)(2), $\mathcal{A}_0$ reports to $\mathcal{Z}$ the outgoing message (if any) generated by the activated session. In case of (3), $\mathcal{A}_0$ reports the collected information. $\mathcal{Z}$ can read the outputs from $\{\tilde{P}_i\}$. He can get the reports from $\mathcal{A}_0$ by reading his output. At the end of the game (which is decided by $\mathcal{Z}$), $\mathcal{Z}$ outputs a bit b'. This completes the description of G_0. Now we define the second game. Denote it by G_1. Assume $\tilde{P}_1, \cdots, \tilde{P}_n$ is executing $\hat{\mathcal{F}}$ with an adversary $\mathcal{A}_1$. A PPT machine $\mathcal{Z}$ is described as follows. Initially, a third party $\mathbb{T}_\rho$ takes $I_\rho(r) = \{I_\rho(r)_i\}_{i=0}^n$ for $r \leftarrow \{0,1\}^\kappa$ and provides $I_\rho(r)_0$ to both $\mathcal{A}_1$ and $\mathcal{Z}$. Later $I_\rho(r)_i$ is provided to $\mathcal{A}_1$ if $\tilde{P}_i$ is corrupted. The remaining description for $\mathcal{Z}$ (i.e., supplying inputs to $\tilde{P}_i$ and instructions to $\mathcal{A}_1$) is exactly as in G_0, except $\mathcal{A}_1$ instead of $\mathcal{A}_0$ will respond to these instructions. The action of $\mathcal{A}_1$ is arbitrary, except that he follows the rules of ideal process in computing $\hat{\mathcal{F}}$ (see Section 2.2). At the end of G_1, $\mathcal{Z}$ generates a bit b'. Now we are ready to define our notion of deniably secure MT-authenticator.

Definition 3. *Let ρ be a protocol for computing $\hat{\mathcal{F}}$. ρ is said to be a* deniable MT-authenticator *if there exists a PPT simulator $\mathcal{A}_1$ such that for every PPT machine $\mathcal{Z}$,*

$$\Pr[\mathcal{Z}(\mathsf{G}_0) = 1] - \Pr[\mathcal{Z}(\mathsf{G}_1) = 1] \tag{3}$$

is negligible.

Essentially, G_b is placed in a black box. The task of $\mathcal{Z}$ is to guess which game is inside. Intuitively, G_0 is the execution of ρ by $\{\tilde{P}_i\}$ and adversary $\mathcal{A}_0$ in the UM, except that $\mathcal{A}_0$ is instructed by $\mathcal{Z}$ and that the inputs of each $\tilde{P}_i$ is supplied by $\mathcal{Z}$ too. G_1 is the execution of $\hat{\mathcal{F}}$ by $\{P_i\}$ and adversary $\mathcal{A}_1$ in the AM, except the input of $\tilde{P}_i$ is supplied by $\mathcal{Z}$ and that $\mathcal{A}_1$ has to pretend to be $\mathcal{A}_0$ s.t. $\mathcal{Z}$ can not decide whether he is interacting with G_1 or with G_0. In order for $\mathcal{Z}$ to fail in realizing which is the case, one might hope that $\mathcal{A}_1$ internally simulates the execution of ρ with the initialization of I_ρ by himself (thus he knows all parties' secret keys). In this way, as long as the output of the internal simulated ith party is identical to that of $\tilde{P}_i$, then $\mathcal{Z}$ can not distinguish $\mathsf{G}_1/\mathsf{G}_0$ since the simulation can be perfect as in the real execution of ρ in G_0. However, $\mathcal{Z}$ has the official public key $I_\rho(r)_0$ of ρ received from $\mathbb{T}$. If $\mathcal{A}_1$ initializes ρ by himself, the simulation will not be consistent with $I_\rho(r)_0$. For example, $\mathcal{Z}$ could realize that a message reported by $\mathcal{A}_1$ is computed using a public-key not in $I_\rho(r)_0$. In this case, $\mathcal{Z}$ immediately realizes he is interacting with G_1. Thus, to make Definition 3 satisfied, $\mathcal{A}_1$ needs to simulate the execution of ρ based on $I_\rho(r)_0$ (and corrupted $\{I_\rho(r)_i\}$). The following theorem says, if such a $\mathcal{A}_1$ indeed exists, then composing ρ with $\hat{\mathcal{F}}$-hybrid protocol π will give arise to a deniably authenticated π^ρ. Formally,

Theorem 1. *If ρ is a deniable* MT-*authenticator and π is a protocol in the $\hat{\mathcal{F}}$-hybrid model, then π^ρ is deniably authenticated.*

Before our actual proof, we first present the main idea. For an UM adversary $\mathcal{U}$, we need to present an AM simulator $\mathcal{A}$ such that the global output in the AM is computationally indistinguishable to that in the UM. Essentially, $\mathcal{A}$ follows $\mathcal{U}$, except the part in executing $\hat{\mathcal{F}}$, where $\mathcal{A}$ activates $\mathcal{A}_1$ through a sequence of instructions. If the ideal process in executing $\hat{\mathcal{F}}$ with $\mathcal{A}_1$ is replaced with the real execution of ρ with $\mathcal{A}_0$, then $\mathcal{A}$ becomes identical to $\mathcal{U}$. Thus, if (2) is violated, we can construct a PPT machine $\mathcal{Z}_\rho$ to distinguish G_0 and G_1 as follows. $\mathcal{Z}_\rho$ simulates $\{P_i\}$ in π and $\boldsymbol{\Phi}$, and also follows $\mathcal{A}$, except that whenever he needs to simulate a message transmission, he does it through the challenge game G_b for $b \leftarrow \{0,1\}$. Finally, $\mathcal{Z}_\rho$ provides the simulated global output to a distinguisher and outputs whenever he does. If $b = 1$, the global output in the simulation of $\mathcal{Z}_\rho$ is distributed as $\mathrm{Auth}^{\hat{\mathcal{F}}}_{\pi,\mathcal{A},\boldsymbol{\Phi},I_\rho}(\boldsymbol{x})$; otherwise, it is distributed as $\mathrm{UnAuth}_{\pi^\rho,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x})$. As a result, violation of (2) implies a non-negligible advantage of $\mathcal{Z}_\rho$, contradicting the assumption of ρ. Now we start to implement the above idea.

Proof. Let $\mathcal{U}$ be against π^ρ. Assume $I(r) = \{I(r)_i\}_{i=0}^n$ and $I_\rho(r') = \{I_\rho(r')_i\}_{i=0}^n$ be the initialization function for π and ρ respectively. With the above idea in mind, we first construct $\mathcal{A}$ against π in the $\hat{\mathcal{F}}$-hybrid model. $\mathcal{A}$ will involve in executing π in the $\hat{\mathcal{F}}$-hybrid model with n parties as an AM adversary. For ease of presentation, we use P_i, $\tilde{P}_i$ and P'_i to denote the ith party in executing π, ρ (or $\hat{\mathcal{F}}$), and π^ρ respectively. The code of $\mathcal{A}$ is as follows.

a. Initially, $\mathcal{A}$ will receive $I(r)_0$ for π and an extra initialization $I_\rho(r')_0$ (supposedly for ρ) from $\mathbb{T}$. P_i will receive $I(r)_i$ and $I(r)_0$ from $\mathbb{T}$. On the one hand, $\mathcal{A}$ is involved in the execution of π with $P_1, \cdots, P_n$ and $\hat{\mathcal{F}}$. On the other hand, he internally activates $\mathcal{U}$ with $I(r)_0$ and $I_\rho(r')_0$ and simulates π^ρ with $\mathcal{U}$ by playing the roles of (uncorrupted) $P'_1, \cdots, P'_n$. To do this, $\mathcal{A}$ will initialize $\mathcal{A}_1$ with I_ρ to assist his simulation. This internal simulation will be useful to his action in π. Details follow.
b. Whenever P_i wishes to send m to P_j, P_i will play the role of $\tilde{P}_i$ (in Figure 1) to send $(\mathsf{send}, \tilde{P}_j, m)$ to $\hat{\mathcal{F}}$, who will take $id \leftarrow \{0,1\}^\kappa$ and send $(id, \tilde{P}_i, \tilde{P}_j, m)$ to $\mathcal{A}$. $\mathcal{A}$ then activates $\mathcal{A}_1$ with $(id, \tilde{P}_i, \tilde{P}_j, m)$. After seeing any output by $\mathcal{A}_1$, forward it to $\mathcal{U}$.
c. Whenever $\mathcal{U}$ requests to deliver a message msg to P'_j, activate $\mathcal{A}_1$ to deliver msg to $\tilde{P}_j$ in ρ. In turn, if $\mathcal{A}_1$ generates any output msg', $\mathcal{A}$ provides it to $\mathcal{U}$. (Remark: as the output of $\mathcal{A}_0$ in such a query is to report the outgoing message from P_j, the output of $\mathcal{A}_1$ is supposed to be a simulation of such a message.) If $\mathcal{A}_1$ generates an outgoing message (c, id) to $\hat{\mathcal{F}}$, $\mathcal{A}$ sends (c, id) to $\hat{\mathcal{F}}$ as his reply for the request of bit c.
d. Whenever $\mathcal{U}$ requests to see an output of P'_i, collect the output of P_i in π (not including the parts labeled as 'secret') and provide it to $\mathcal{U}$. Note since both $\mathcal{A}$ and $\mathcal{U}$ are not allowed to see the secret parts, this simulation is perfect.
e. Whenever $\mathcal{U}$ asks to corrupt a session id in P'_i, corrupt the corresponding session in π and obtain the session state $stat$. In addition, he, the role of $\mathcal{Z}$ in G_1, requests $\mathcal{A}_1$ to corrupt all the sessions in $\tilde{P}_i$ that sending messages

containing session id (Recall each message contains the sender session ID of π; See the protocol syntax in Section 2). As a result, $\mathcal{A}_1$ will output simulated internal states $stat'$ for all these sessions. $\mathcal{A}$ merges $stat'$ and $stat$ to provide a complete session state st for session id of P_i' in the simulated π^ρ. Finally, $\mathcal{A}$ provides st to $\mathcal{U}$.

f. Whenever $\mathcal{U}$ asks to corrupt a party P_i', corrupt P_i in π to get $I(r)_i$ and then obtain the secret key $I_\rho(r)_i$ (from $\mathbb{T}$ by requesting $\mathcal{A}_1$ to corrupt $\tilde{P}_i$). Obtain internal states for all sessions in P_i through session corruption as in item (e). Finally, provide all the information to $\mathcal{U}$.

Finally, $\mathcal{A}$ outputs whatever $\mathcal{U}$ does.

We claim that $\mathcal{A}$ will satisfy (2). Otherwise, we construct a PPT machine $\mathcal{Z}_\rho$ to distinguish $\mathsf{G}_0/\mathsf{G}_1$ with a non-negligible advantage. To do this, we first show that the simulation of $\mathcal{A}$ can be completed by black-box access to the game G_1. Indeed, we only need to check the black-box access restriction for $\mathcal{A}$ can be satisfied.

– In item (a), this will be violated when $\mathcal{A}$ initializes $\mathcal{A}_1$ with $I_\rho(r')_0$. However, since in G_1, $\mathbb{T}$ already initializes $\mathcal{A}_1$ with it, this operation is not required any more.
– In item (b), the code exactly follows the description of G_1, except that $\mathcal{A}$ forwards the request $(id, \tilde{P}_i, \tilde{P}_j)$ to $\mathcal{A}_1$. Thus, under the black-box access to G_1, $\mathcal{A}$ only needs to feed input $(\mathsf{send}, \tilde{P}_j, m)$ to $\tilde{P}_i$ and read the output of $\mathcal{A}_1$ (if any).
– In item (c), $\mathcal{A}$ only needs to feed the instruction "deliver message msg to $\tilde{P}_j$" to $\mathcal{A}_1$. The remaining computation will be perfectly simulated in G_1.
– Item (d)(e)(f) do not violate black-box restriction.

This revision does not change the global output of the simulation (i.e., $\mathrm{Auth}^{\hat{\mathcal{F}}}_{\pi,\mathcal{A},\boldsymbol{\Phi},I_\rho}(\boldsymbol{x})$). On the other hand, when the black-box G_1 is replaced with G_0, then the global output of the simulated game is distributed exactly as $\mathrm{UnAuth}_{\pi^\rho,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x})$. Now we are ready to describe the code of $\mathcal{Z}_\rho$. Given black-box G_b, auxiliary input $\boldsymbol{x}$ and $I_\rho(r')_0$, he initializes $\{I(r)_i\}$ for π in $\hat{\mathcal{F}}$-hybrid mode, simulates $\{P_i\}$ faithfully, plays the role of $\boldsymbol{\Phi}$, and also follow the revised $\mathcal{A}$ with black-box access to G_b. Finally, $\mathcal{Z}_\rho$ provides the global output out of the simulated game to the distinguisher of equation (2) and outputs whatever he does. Note that if $b = 0$, then out is distributed according to the right hand of equation (2); the left hand of (2) otherwise. Thus, non-negligible advantage of the distinguisher implies non-negligible advantage of $\mathcal{Z}_\rho$, contradiction to assumption on ρ. ■

Corollary 1. *Assume ρ is a deniably* MT-*authenticator and π is deniably secure in the $\hat{\mathcal{F}}$-hybrid model, then π^ρ is deniably secure in the* UM.

Proof. Let I, I_ρ be the initialization function for ρ and π, respectively. By Theorem 1, for any UM adversary $\mathcal{U}$ against π^ρ, there exists an AM simulator $\mathcal{A}$ against π such that

$$\mathrm{Auth}^{\hat{\mathcal{F}}}_{\pi,\mathcal{A},\boldsymbol{\Phi},I_\rho}(\boldsymbol{x}) \overset{c}{\equiv} \mathrm{UnAuth}_{\pi^\rho,\mathcal{U},\boldsymbol{\Phi}}(\boldsymbol{x}). \tag{4}$$

Assume π is to realize the functionality $\mathcal{G}$. Since π is deniably secure, there must exists an ideal process simulator $\mathcal{S}$ such that

$$\mathrm{IDEAL}_{\mathcal{G},\mathcal{S},\boldsymbol{\Phi},(I,I_\rho)}(\boldsymbol{x}) \overset{c}{\equiv} \mathrm{Auth}^{\hat{\mathcal{F}}}_{\pi,\mathcal{A},\boldsymbol{\Phi},I_\rho}(\boldsymbol{x}). \tag{5}$$

Combining (4)(5), we conclude the proof. ■

4 Uncontrollable Random Oracle Based Deniable MT-Authenticator

In this section, we will construct a deniable MT-authenticator from random oracle. Notice that the original random oracle [4] is completely controlled by a simulator. Especially, if an oracle query is issued by the simulator himself, he can first choose the output and then decide the query input. This provides too much freedom to a simulator. As pointed out by Pass [24], a solution obtained in this way is *not* deniable as a real attacker does not have this right at all. The random oracle we use here is defined by Pass. This object is maintained by an uncorruptible third party but all the input-output pairs are seen by the simulator. We call it *Uncontrollable Random Oracle* (uRO). Deniability makes sense in this model since whatever computable by a simulator is computable by an attacker himself.

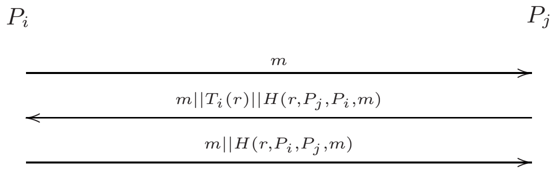


Fig. 2. Our Deniable MT-Authenticator uRO-Auth (Note the complete details appear in the context)

Now we describe our uRO based MT-authenticator. We call it uRO-Auth MT-authenticator. Assume P_i wishes to send a message m to P_j. Let T_i be the public-key of a trapdoor permutation owned by party P_i and D_i be the trapdoor for T_i. P_i first sends m to P_j. P_j then takes $r \leftarrow \{0,1\}^\kappa$, computes and sends back $m\|T_i(r)\|H(r,P_j,P_i,m)$ to P_i. Receiving $m\|\alpha\|\beta$, P_i computes $r' = D_i(\alpha)$. If $r' \neq \perp$, it checks whether $\beta = H(r',P_j,P_i,m)$. If yes, he computes and sends out $m\|\gamma$ to P_j, where $\gamma = H(r',P_i,P_j,m)$. If $r' = \perp$ or β is not successfully verified, he does nothing. Upon receiving $m\|\gamma$, P_j checks whether $\gamma = H(r,P_i,P_j,m)$. If yes, he generates a local output "(receiv, id, P_i, P_j, m)" for $id \leftarrow \{0,1\}^\kappa$; otherwise, it does nothing. The graphic interpretation of the protocol is presented in Figure 2.

Theorem 2. *If H is an uRO, then* $\mathsf{uRO\text{-}Auth}$ *is a deniable* MT*-authenticator.*

Proof. Keep the notations as in the definition of deniable MT-authenticator. We need to construct a simulator $\mathcal{A}_1$ such that for any PPT machine $\mathcal{Z}$

$$\Pr[\mathcal{Z}(\mathsf{G}_0) = 1] - \Pr[\mathcal{Z}(\mathsf{G}_1) = 1] \tag{6}$$

is negligible. The code of $\mathcal{A}_1$ is as follows. First of all, $\mathbb{T}$ randomly samples $\{(T_i, D_i)\}$ and provides $\{T_i\}$ to both $\mathcal{Z}$ and $\mathcal{A}_1$. The uncontrollable random oracle H is assumed to work as follows. It maintains a list L_H which is initially empty. Upon a hash query x, this H-oracle checks whether x was queried before. If not, it takes $y \leftarrow \{0,1\}^\kappa$ and adds (x, y) to L_H; otherwise, it takes the existing record (x, y) from L_H. The answer to query x is y. The detailed simulation by $\mathcal{A}_1$ is as follows. Essentially, $\mathcal{A}_1$ internally simulates ρ based on $I_\rho(r)$ and corrupted $I_\rho(r)_i$ in order to properly behave in the execution of $\hat{\mathcal{F}}$. We will use P_i and $\tilde{P}_i$ to denote the ith party in the internal simulation of ρ and the external execution of $\hat{\mathcal{F}}$.

- I_1 Whenever $\mathcal{A}_1$ receives a message $(id, \tilde{P}_i, \tilde{P}_j, m)$ (from $\hat{\mathcal{F}}$) and is asked for a bit c, he internally simulates P_i to send a flow one message m to P_j in his simulated $\mathsf{uRO\text{-}Auth}$ and reports this flow one message to $\mathcal{Z}$ (intuitively, let $\mathcal{Z}$ believe he is interacting with real execution of $\mathsf{uRO\text{-}Auth}$).
- I_1' Whenever $\mathcal{A}_1$ was requested to start a session in the name of corrupted P_i to authenticate a message m to P_j, $\mathcal{A}_1$ first in the name of corrupted $\tilde{P}_i$ sends $(\mathsf{send}, \tilde{P}_j, m)$ to $\hat{\mathcal{F}}$. The remaining action of this query is to follow item I_1.
- I_2 Whenever $\mathcal{Z}$ requests $\mathcal{A}_1$ to deliver a message msg from P_i to a responder P_j (i.e., P_j plays the authentication receiver in the protocol), $\mathcal{A}_1$ does the following. $\mathcal{A}_1$ represents P_j to do so honestly in the simulation of $\mathsf{uRO\text{-}Auth}$. If msg is a flow one message, he reports the simulated flow two message back to $\mathcal{Z}$; otherwise, msg is flow three message. In this case, if the simulated P_j accepts, $c = 1$; $c = 0$ otherwise. Feedback (c, id) to $\hat{\mathcal{F}}$, where if some $(id, \tilde{P}_i, \tilde{P}_j, m)$ was received from $\hat{\mathcal{F}}$ but $(*, id)$ has not been feedback, take id as in this received tuple; otherwise $id \leftarrow \{0,1\}^\kappa$. In any case, if $c = 1$, $\mathcal{A}_1$ simulates P_j to generate an output $(\mathsf{receiv}, id, P_i, P_j, m)$. Denote the event that $c = 1$ but $(id, \tilde{P}_i, \tilde{P}_j, m)$ was never received before by Bad_0; denote the event $\tilde{P}_i$ is uncorrupted and $c = 1$ but $(c^*, *)$ on m for a bit c^* was sent to $\hat{\mathcal{F}}$ by Bad_1. Note under $\neg\mathsf{Bad}_0 \wedge \neg\mathsf{Bad}_1$, outputs of P_j and $\tilde{P}_j$ are identical.
- I_3 Whenever $\mathcal{A}_1$ is asked to deliver a Flow2 message $m||\alpha||\beta$ to P_i, $\mathcal{A}_1$ checks whether L_H has a record $((r', P_j, P_i, m), \beta)$ for some r' such that $\alpha = T_i(r')$. If the check fails, it terminates this session; otherwise (in this case r' is unique since T_i is a permutation), asks H-oracle for query (r', P_i, P_j, m). Assume the answer is γ. He then simulates to send out $m||\gamma$ to P_j and reports this message to $\mathcal{Z}$. This simulation is perfect except when β happens to be valid while (r', P_j, P_i, m) satisfying $\alpha = T_i(r')$ was not queried to H-oracle (this covers the attack by forging flow two message without query (r', P_j, P_i, m) to H-oracle). We note this event by $\mathbf{E}_1$. We know that the number of Flow2 message is upper bounded by run time of $\mathcal{Z}$ (denoted by R_z). Then, $\Pr[\mathbf{E}_1] \leq \frac{R_z}{2^\kappa}$.

I_4 Whenever $\mathcal{A}_1$ is requested to reveal a session in P_t, $\mathcal{A}_1$ represents P_t to do so honestly. No matter P_t is a sender or receiver of m, we have that before Flow2 message the session state of P_t is m while after Flow2 message the session state of P_t is $m||r'$ Note the above session state is well defined if event $\neg\mathbf{E}_1$ holds. $\mathcal{A}_1$ then reports the collected session state back to $\mathcal{Z}$.

I_5 When $\mathcal{A}_1$ is requested to corrupt P_t, he first obtains D_t from $\mathbb{T}$, then combines all the internal states in sessions in P_t. Finally report them to $\mathcal{Z}$. This simulation is perfect under event $\neg\mathbf{E}_1$.

From the above simulation, under $\neg\mathsf{Bad}_0 \wedge \neg\mathsf{Bad}_1$, the outputs of P_i and $\tilde{P}_i$ are exactly identical. In addition, the simulation of $\mathcal{A}_1$ differs from the real execution of uRO-Auth only if E_1 occurs. Thus, under $\neg\mathsf{Bad}_0 \wedge \neg\mathsf{Bad}_1 \wedge \neg\mathsf{E}_1$, the view of $\mathcal{Z}$ is identical to when interacting with G_0. So it remains to show that $\Pr[\mathsf{Bad}_0 \vee \mathsf{Bad}_1 \vee \mathsf{E}_1]$ is negligible. First, Bad_0 occurs if uncorrupted P_j successfully verifies a flow three message (m, γ) and thus attempts to feedback a bit (c, id) to $\mathcal{F}$ but he never received a $(id, \tilde{P}_i, \tilde{P}_j, m)$ from the latter. Bad_1 implies two uncorrupted sessions accepts m. Since no uncorrupted sender sends the same m twice, at lest one session has no sender session. Thus, Bad_1 occurs only if r taken in these two receiver sessions happen to be identical or otherwise if (r, P_i, P_j, m, γ) with different r are consistent for both sessions (which has a probability $\frac{R_z^2}{2^\kappa}$, as for at least one session (r, P_i, P_j, m) was not queried to H-oracle prior to receipt of γ). This gives $\Pr[\mathsf{Bad}_1] \leq \frac{2R_z^2}{2^\kappa}$. We now bound $\Pr[\mathsf{Bad}_0 \wedge \neg\mathsf{E}_1]$. Let ϵ be the probability that a trapdoor permutation adversary succeeds within run time R_z. We show that $\Pr[\mathsf{Bad}_0 \wedge \neg\mathsf{E}_1] \leq nR_z\epsilon + R_z/2^k$. Intuitively, a $\mathsf{Bad}_0 \wedge \neg\mathsf{E}_1$ implies $\mathcal{Z}$ is able to decrypt the permutation in flow two in order to forge a valid flow three message. Details will appear in the full paper.

Since $\Pr[\mathsf{Bad}_0 \vee \mathsf{E}_1 \vee \mathsf{Bad}_1] \leq \Pr[\mathsf{Bad}_0 \wedge \neg\mathsf{E}_1] + \Pr[\mathsf{E}_1] + \Pr[\mathsf{Bad}_1] \leq nR_z\epsilon + \frac{2R_z+2R_z^2}{2^\kappa}$, we have $\Pr[\mathcal{Z}(\mathsf{G}_1) = 1] - \Pr[\mathcal{Z}(\mathsf{G}_0) = 1] \leq nR_z\epsilon + \frac{2R_z+2R_z^2}{2^\kappa}$ too. This concludes our proof. ■

5 Application to Deniable Key Exchange

Key exchange is a communication procedure in which participants establish a temporarily shared secret key. To evaluate the security, several models are proposed in the literature [1,3,5]. Here we use the model in [18], a slightly revised version of [1]. In this model, an ideal process is defined. Then a real protocol λ is constructed. λ is said to be secure if for any adversary against λ, there exists an adversary against the ideal process such that the global output in these two worlds are indistinguishable. Here the ideal process as well as the security definition should be slightly modified to be consistent with that in Section 2.3. In [18], a $\mathcal{F}$-hybrid secure key exchange protocol Encr-KE was proposed (See Figure 3), where $(\mathcal{G}(1^\kappa), \mathcal{E}, \mathcal{D})$ is a semantically secure public-key encryption scheme. Notice that this protocol has an empty initialization function. It follows that Encr-KE is deniably secure in the $\mathcal{F}$-hybrid model in the sense of Definition 1 (note the

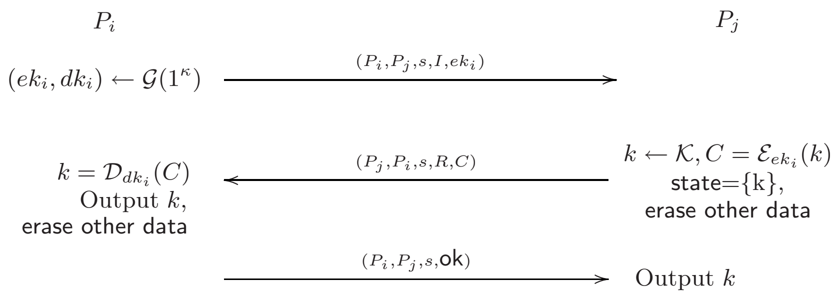

P_i
P_j
$(ek_i, dk_i) \leftarrow \mathcal{G}(1^\kappa)$
(P_i, P_j, s, I, ek_i)
$k \leftarrow \mathcal{K}, C = \mathcal{E}_{ek_i}(k)$
state={k},
erase other data
(P_j, P_i, s, R, C)
$k = \mathcal{D}_{dk_i}(C)$
Output k,
erase other data
$(P_i, P_j, s, \mathsf{ok})$
Output k

Fig. 3. AM-secure Key Exchange Protocol Encr-KE, Details see [18]

original proof needs to be slightly modified in order to cater our formalization of authenticated-link model).

Denote the key exchange protocol obtained by applying uRO-Auth to Encr-KE by uROE-KE. From the deniable authenticator theorem, we have

Theorem 3. uROE-KE *is a deniably secure key exchange protocol in the* UM*.*

Acknowledgement

Shaoquan Jiang would like to thank Prof. Hugh Williams for funding his research at University of Calgary and Prof. Rei Safavi-Naini for supporting his partial travel expense.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key exchange protocols. In: Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, pp. 419–428 (1998)
2. Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption without Random Oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, Springer, Heidelberg (2004)
3. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, Springer, Heidelberg (1994)
4. Bellare, M., Rogaway, P.: Random Oracle is Practical: A Paradigm for Designing Efficient Protocols. In: ACM CCS 1993, pp. 62–73 (1993)
5. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
6. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: FOCS 2001 (2001)
7. Dåmgard, I.: Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In: Advances in Cryptology-Eurocrypt 2000, pp. 418–430 (2005)

8. Dent, A.: The Cramer-Shoup Encryption Scheme is Plaintext Aware in the Standard Model. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, Springer, Heidelberg (2006)
9. Desmedt, Y.: Subliminal-Free Authentication and Signature (Extended Abstract). In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 23–33. Springer, Heidelberg (1988)
10. Di Raimondo, M., Gennaro, R.: New Approaches for Deniable Authentication. In: ACM CCS 2005, pp. 112–121 (2005)
11. Di Raimondo, M., Gennaro, R., Krawczyk, H.: Deniable Authentication and Key Exchange. In: ACM CCS 2006 (2006)
12. Dolev, D., Dwork, C., Naor, M.: Non-malleable Cryptography. SIAM J. Comput. 30(2), 391–437 (2000); Earlier version appeared in STOC 1991, pp. 542-552 (1991)
13. Dwork, C., Naor, M.: Zaps and Their Applications. In: FOCS 2000, pp. 283–293 (2000)
14. Dwork, C., Naor, M., Sahai, A.: Concurrent Zero-Knowledge. In: STOC 1998, pp. 409–418 (1998)
15. Dwork, C., Sahai, A.: Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 442–457. Springer, Heidelberg (1998)
16. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM J. Comput. 18(1), 186–208 (1989)
17. Groth, J., Ostrovsky, R., Sahai, A.: Perfect Non-interactive Zero Knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
18. Jiang, S., Gong, G.: Efficient Authenticators with Application to Key Exchange. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 81–91. Springer, Heidelberg (2006)
19. Katz, J.: Efficient and Non-malleable Proofs of Plaintext Knowledge and Applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 211–228. Springer, Heidelberg (2003)
20. Killian, J., Petrank, E.: Concurrent and Resettable Zero-Knowledge in Poly-Logarithmic Rounds. In: ACM STOC 2001, pp. 560–569 (2001)
21. Krawczyk, H.: SKEME: a versatile secure key exchange mechanism for Internet. In: NDSS 1996, pp. 114–127 (1996)
22. Lindell, Y.: Lower Bounds and Impossibility Results for Concurrent Self Composition. Journal of Cryptology (to appear)
23. Naor, M.: Deniable Ring Authentication. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 481–498. Springer, Heidelberg (2002)
24. Pass, R.: On the deniability in the common reference string and random oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003)
25. Prabhakaran, M., Sahai, A.: Concurrent Zero Knowledge Proofs with Logarithmic Round-Complexity. In: FOCS 2002, pp. 366–375 (2002)
26. Richardson, R., Kilian, J.: On the Concurrent Composition of Zero-Knowledge Proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 415–431. Springer, Heidelberg (1999)

Orthogonality between Key Privacy and Data Privacy, Revisited

Rui Zhang, Goichiro Hanaoka, and Hideki Imai

Research Center for Information Security (RCIS)
National Institute of Advanced Industrial Science and Technology (AIST)
{r-zhang,hanaoka-goichiro,h-imai}@aist.go.jp

Abstract. Key privacy is a notion regarding the privacy of the owner of a public key, which has important applications in building (receiver) anonymous channels, or privacy-enhanced authentication/signature schemes. Key privacy is considered to be an orthogonal (i.e., independent), notion from data privacy, while the key privacy of many public key encryption schemes has not been explored, though their data privacy is comparatively well understood. In this paper, we study key privacy of many practical encryption schemes and identify evidences that key privacy is not comparable to data privacy. We also formalize key privacy in the plaintext checking attack model and point out some generic transforms to enhance the key privacy of an encryption scheme. Interestingly, these well-known techniques have been used to enhance data security. Finally, we give detailed security analyses on the signed hashed-ElGamal encryption [27] in the random oracle model, of both key privacy and data security against chosen ciphertext attack. Again, this specific example supports our claim on the relation of two notions.

1 Introduction

Key privacy [4] is a notion regarding receiver anonymity, which captures an adversary's inability to learn any knowledge about the encryption key used for a given ciphertext. It has many applications, e.g., building an anonymous channel providing receiver anonymity. Also encryption schemes with key privacy are often used in generic constructions to build privacy-enhanced signature schemes.

Key privacy is considered as a totally different notion from data privacy, e.g., semantic security, and the authors of [4] mentioned that "*it is not hard to see that the goals of data privacy and key privacy are orthogonal*." They then explained this by showing some concrete encryption schemes which have data privacy but without key privacy, e.g., RSA-OAEP [8].

On the other hand, however, there exist some facts which seem to imply a strong relationship between key privacy and data privacy. Many natural semantically secure encryption schemes, e.g. ElGamal [12] and Cramer-Shoup [10], already have key privacy. Furthermore, Hayashi and Tanaka [18] pointed out, a well-known technique providing strong data privacy, i.e. [15], is also effective to upgrade the underlying scheme to have chosen ciphertext security, though their discussions are limited to *plaintext awareness* (PA) [5]. By "plaintext awareness", informally, it is required that an adversary should have already known corresponding plaintext before it queries a decryption oracle. PA has been a useful but very strong notion that helps to prove an encryption scheme has strong data privacy against adaptive chosen ciphertext attack (CCA) [20,25].

In this paper, we reconsider the relationship between key privacy and data privacy from a different perspective, and confirm that these two notions are actually orthogonal. However, instead of merely giving counterexamples, we give clear evidence. We show, some cryptographic techniques for data privacy happen to be able to upgrade key privacy, but they don't provide key privacy. Towards this goal, as an intermediate step, we study key privacy in the plaintext checking attack model and show that orthogonality of the two notions still holds in this model. Then we prove that (1) key privacy and data privacy are orthogonal (as pointed out by Bellare et al.), but (2) (natural) techniques (FOpkc [14], REACT [21]) for achieving data privacy against chosen-ciphertext attacks can also straightforwardly applied to obtain key privacy against the same attack (CCA). As another strong evidence for these claims, we give a security analysis of the signed hashed-ElGamal encryption [27] in the random oracle model, for both key privacy and data privacy. Existence of such schemes especially supports the second claim since this does not satisfy PA.

1.1 Our Contributions

Previous research [4] gives clear conclusions that the ElGamal encryption has key privacy against chosen plaintext attack and the Cramer-Shoup encryption has key privacy against chosen ciphertext attack, secure encryption[1], both under the decisional Diffie-Hellman (DDH) assumption. However, consider the DHIES [1] or ElGamal-REACT [21]. To prove data privacy of these schemes, one has to grant the simulator with access to a DDH oracle, furthermore, the access to the DDH oracle is essential for the proof [29]. Then it is already not obvious whether these useful schemes have key privacy in a group where DDH is easy, since no previous work has been done regarding key privacy under non-decisional versions of Diffie-Hellman assumptions.

On the other hand, consider some encryption schemes, such as the signed hashed-ElGamal encryption, in fact, this scheme is not plaintext aware, as we will prove later. Thus within known techniques, even if one proves the key privacy against chosen plaintext attack, to investigate key privacy against chosen ciphertext attack, one has to start from scratch. Then a natural question arises, *whether we can do it more intelligently when dealing with key privacy rather than these brute force treatments*? Previous studies didn't provide with us ready tools to study these schemes.

Our contribution is three-fold. We first give formal evidences that key privacy is orthogonal to data privacy. We then elaborate that well-known transforms to acquire chosen ciphertext security for data privacy, such as FOpkc [14] and REACT [22], are still effective to upgrade underlying schemes to have key privacy against chosen ciphertext attack. Finally, we give a specific security proof with exact security reductions on the data privacy and key privacy, namely indistinguishability against chosen ciphertext attack (IND-CCA) security and indistinguishability of keys against chosen ciphertext

[1] In fact, these are the only two practical schemes whose key privacy are analyzed besides those based on RSA assumptions in [4] and those acquired via the second Fujisaki-Okamoto transform [14] in [18].

attack (IK-CCA) security of the signed hashed-ElGamal encryption. We further show that the signed hashed-ElGamal encryption is not plaintext aware.

As an independent interest, we formalize the notion of key privacy in the plaintext checking attack (PCA) model [21]. A plaintext checking attack is a chosen plaintext attack plus a plaintext checking oracle that decides whether a ciphertext encrypts a given plaintext.

2 Preliminary

In this section, we review some basic definitions and security notions of public key encryption and symmetric key encryption, signature schemes, some number theoretic assumptions as well as variants of the ElGamal encryption.

Notations. Define $x \leftarrow_R X$ as x being uniformly chosen from a finite set X. If A is an algorithm, $x \leftarrow A$ means that the output of A is x. When y is not a finite set nor an algorithm, $x \leftarrow y$ is an assignment operation. A function $f(k)$ is negligible, if for any constant c there exists $k_0 \in \mathbb{N}$, such that $f(k) < (1/k)^c$ for any $k > k_0$.

2.1 Public Key Encryption

A public key encryption scheme consists of three algorithms $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$. K is the randomized key generation algorithm, which takes a security parameter k, generates a pair of key (pk, sk), where pk is a public key and sk is a secret key. Denote $\mathcal{M}$ and C as the plaintext and ciphertext spaces, respectively. E is the possibly randomized encryption algorithm, which takes pk and a plaintext $m \in \mathcal{M}$ as input, together with internal random coin r, and outputs a ciphertext c as the output, denoted as $c \leftarrow \mathsf{E}(pk, m; r)$. D is the deterministic decryption algorithm, which takes sk and a ciphertext c as input, output the corresponding m, or "$\perp$" if decryption algorithm fails, denoted as $m \leftarrow \mathsf{D}(sk, c)$. We require that Π meets standard correctness requirement, namely, for all $(pk, sk) \leftarrow \mathsf{K}(1^k)$, and all $m \in \mathcal{M}$, $\mathsf{D}(sk, \mathsf{E}(pk, m; r)) = m$.

Indistinguishability. The widely accepted security notion for public key encryption is indistinguishability against adaptive chosen ciphertext attack (IND-CCA) [16,11,20,25,5].

Definition 1 (IND-CCA). *Let $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be a public key encryption scheme. Let $\mathcal{A}$ be an adversary. Let $\mathcal{DO}$ be a decryption oracle that replies the corresponding plaintext upon a query on ciphertext c. We define the advantage of $\mathcal{A}$ as*

$$Adv_{\Pi,\mathcal{A}}^{\text{ind-cca}}(k) = \Pr\left[b' = b \,\middle|\, \begin{array}{l} (pk, sk) \leftarrow \mathsf{K}(1^k); (m_0, m_1, s) \leftarrow \mathcal{A}^{\mathcal{DO}}(pk); \\ b \leftarrow_R \{0,1\}; c^* \leftarrow \mathsf{E}(pk, m_b); b' \leftarrow \mathcal{A}^{\mathcal{DO}}(c^*, s) \end{array} \right] - \frac{1}{2}$$

*where $\mathcal{A}$ is forbidden to query c^* at $\mathcal{DO}$. A scheme is (ϵ, t)-*IND-CCA *secure if any adversary with running time t, has advantage at most ϵ in winning the above game. Furthermore, we say a public key encryption scheme is* IND-CCA *secure if for any PPT adversary, ϵ is negligible.*

Key Privacy. We now review the security notion for key privacy, i.e., indistinguishability of keys against adaptive chosen ciphertext attack (IK-CCA) [4].

Definition 2 (IK-CCA). *Let $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be a public key encryption scheme. Let $\mathcal{A}$ be an adversary. Let $\mathcal{DO}$ be a decryption oracle that replies the corresponding plaintext upon a query on ciphertext c. We define the advantage of $\mathcal{A}$ as*

$$Adv^{\text{ik-cca}}_{\Pi,\mathcal{A}}(k) = \Pr\left[b' = b \left| \begin{array}{c} (pk_0, sk_0) \leftarrow \mathsf{K}(1^k); (pk_1, sk_1) \leftarrow \mathsf{K}(1^k); \\ (m^*, s) \leftarrow \mathcal{A}^{\mathcal{DO}}(pk_0, pk_1); b \leftarrow_R \{0,1\}; \\ c^* \leftarrow \mathsf{E}(pk_b, m^*); b' \leftarrow \mathcal{A}^{\mathcal{DO}}(c^*, s) \end{array} \right. \right] - \frac{1}{2}$$

where $\mathcal{A}$ is forbidden to query c^* on $\mathcal{DO}$. A scheme is (ϵ, t)-IK-CCA secure if any adversary with running time t, has advantage at most ϵ in winning the above game. Furthermore, we say a public key encryption scheme is IK-CCA secure if for any PPT adversary, ϵ is negligible.

For indistinguishability against chosen plaintext attack (IK-CPA), everything will be the same to the above experiment, except that $\mathcal{A}$ is not given access to $\mathcal{DO}$.

Plaintext Awareness. Informally, plaintext awareness [7,5] states an intuition that if an adversary can produce a valid ciphertext, it should "have already known" the plaintext. Plaintext awareness (PA) is another important notion regarding encryption schemes secure against chosen ciphertext attack (CCA) and many encryption schemes (especially those with random oracles) are built on this idea. On the other hand, there are some attempts to formalize the notion in the standard model, e.g. [19,6], but these definitions seem quite artificial. Moreover, PA is more like a means rather than a goal, since many useful encryption schemes are not plaintext aware [13].

To formulate the intuition, an adversary $\mathcal{B}$ is given a public key pk and access to a random oracle $\mathcal{G}$. Additionally, it is provided with a ciphertext generation oracle $\mathcal{E}^{\mathcal{G}}(pk)$ which with input pk, randomly chooses $m \in \mathcal{M}$, and outputs a ciphertext y. To be plaintext aware, $\mathcal{B}$ should necessarily infer m from y and oracle queries to $\mathcal{G}$.

Definition 3 (PA). *Let $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be a public key encryption scheme, let Γ be the set of all corresponding random functions, let $\mathcal{B}$ be an adversary, U be a plaintext extractor and $\mathcal{G}$ be a random oracle. $\mathcal{B}$ is additionally given access to an ciphertext generation oracle $\mathcal{E}^{\mathcal{G}}(pk)$. Denote the list* $\mathtt{H}$ *that contains all the random oracle queries from $\mathcal{B}$ to $\mathcal{G}$ and the list* C *that contains all the replies by $\mathcal{E}^{\mathcal{G}}(pk)$. We require the queries of $\mathcal{B}$ to $\mathcal{E}^{\mathcal{G}}(pk)$ is not included in* C *and interaction between $\mathcal{E}^{\mathcal{G}}(pk)$ and $\mathcal{G}$ is not included in* $\mathtt{H}$. *For any $k \in \mathbb{N}$, define*

$$Suc^{\text{pa}}_{\Pi,\mathcal{B},U}(k) = \Pr\left[U(\mathtt{H}, \mathsf{C}, y, pk) = \mathsf{D}^{\mathcal{G}}(sk, y) \left| \begin{array}{c} \mathcal{G} \leftarrow_R \Gamma; (pk, sk) \leftarrow \mathsf{K}(1^k); \\ (\mathtt{H}, \mathsf{C}, y) \leftarrow \mathcal{B}^{\mathcal{G}, \mathcal{E}^{\mathcal{G}}(pk)} \end{array} \right. \right]$$

The above definition is only defined in the single public key model, it is easily generalized to multiparty setting by modifying the key generation and oracle access for $\mathcal{B}$ in the above experiment, since the code of the extractor U can be used for multiple instances of the same encryption algorithm. A recent work [18] did this in more details, particularly with the FOcrypto transform [15].

Plaintext Checking Attack (PCA). Okamoto and Pointcheval [21] defined this attack model called plaintext checking attack (PCA), potentially motivated by an interesting property of some elliptic curves. In such an attack model, in addition to the public key, the adversary is given access to a plaintext checking oracle $\mathcal{PCO}$, which upon inputs of plaintext/ciphertext pair (m, c) and a public key pk, returns "yes" if $\mathsf{D}(sk, c) = m$ and "no" otherwise. This attack model is important and the security proof of many useful encryption scheme is based on this formulation [21,1]. We note that in a single public key setting, usually, the public key pk need not to be part of the inputs to $\mathcal{PCO}$.

We review here the security notion of onewaysness of data privacy against plaintext checking attack. Informally, onewayness means that it is computationally infeasible to "undo" the encryption without knowing the secret key.

Definition 4 (OW-PCA). *Let $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be a public key encryption scheme. Let $\mathcal{A}$ be an adversary, and $\mathcal{PCO}$ be a plaintext checking oracle, We define the success probability of $\mathcal{A}$ as*

$$Succ^{\mathsf{ow\text{-}pca}}_{\Pi,\mathcal{A}}(k) = \Pr\left[m' = m^* \middle| \begin{array}{c} (pk, sk) \leftarrow \mathsf{K}(1^k); m^* \leftarrow_R \mathcal{M}; \\ c^* \leftarrow \mathsf{E}(pk, m^*); m' \leftarrow \mathcal{A}^{\mathcal{PCO}}(c^*, pk) \end{array} \right]$$

*A scheme is (ϵ, t)-*OW-PCA *secure if any adversary with running time t, succeeds with probability at most ϵ in winning the above game. Furthermore, we say a public key encryption scheme is* OW-PCA *secure if for any PPT adversary, ϵ is negligible.*

Remark 1. An important previous belief on PCA is that indistinguishability is *meaningless* in this model, whereas onewayness can be meaningful. To see this, since the adversary can query the plaintext checking oracle with its target ciphertext and one of its chosen message, thus easily breaks the indistinguishability of the scheme. However, we believe if ruling out this trivial attack, one can still define indistinguishability of data privacy in plaintext in the PCA model. The formulation of the security is quite straightforward and we omit here.

2.2 Digital Signature

A signature scheme Σ consists of three algorithms $\Sigma = (\mathsf{G}, \mathsf{S}, \mathsf{V})$. The randomized key generation algorithm G takes a security parameter k, and generates signing key $sigk$ and verification key vk. The possibly randomized signing algorithm S takes as inputs $sigk$ and $m \in \{0, 1\}^*$, where m is a message, and outputs a signature σ. The deterministic verification algorithm V takes as inputs vk, m and σ, and outputs a symbol $\beta \in \{\mathtt{accept}, \mathtt{reject}\}$, denoted as $\beta \leftarrow \mathsf{V}(vk, m, \sigma)$. We require that for all $(sigk, vk)(= \mathsf{G}(1^k))$, all $m \in \{0, 1\}^*$, $\mathtt{accept} = \mathsf{V}(vk, m, \mathsf{S}(sigk, m))$.

Unforgeability. Here, we mainly consider strong unforgeability, i.e., sUF-CMA [3], rather than the weaker version (UF-CMA) [17]. Let $\Sigma = (\mathsf{G}, \mathsf{S}, \mathsf{V})$ be a signature scheme. Let $\mathcal{A}$ and k be an adversary and a security parameter, respectively.

Definition 5 (sUF-CMA). *Denote $\{(\sigma_i, m_i)\}_{q_s}$ as the set contains all q_s pairs of queries and replies between $\mathcal{A}$ and $\mathcal{SO}$, where $\mathcal{SO}$ is a signing oracle which for a given message m, returns a corresponding signature σ. The success probability of $\mathcal{A}$ is defined as*

$$Succ_{\mathcal{A},\Sigma}^{\mathsf{suf\text{-}cma}}(k) = \Pr\left[\begin{array}{l|l} \mathsf{V}(vk, m^*, \sigma^*) = \mathtt{accept} & (vk, sigk) \leftarrow \mathsf{Gen}(1^k); \\ \wedge(\sigma^*, m^*) \notin \{(\sigma_i, m_i)\}_{q_s} & (\sigma^*, m^*) \leftarrow \mathcal{A}^{\mathcal{SO}}(vk) \end{array}\right],$$

*We say Σ is (t, ϵ)-*sUF-CMA *secure if for any $\mathcal{A}$ in time bound t, $\mathcal{A}$'s success probability is at most ϵ. Especially, we say that Σ is* sUF-CMA *secure if ϵ is negligible.*

2.3 Number Theoretic Assumptions

We briefly review the discrete logarithm (DL), the computational Diffie-Hellman (CDH), the decisional Diffie-Hellman (DDH), and the gap Diffie-Hellman (GDH) assumptions.

Definition 6 (Assumptions). *Let $\mathbb{G}$ be a cyclic group of order q, where q is a large prime, let g be a generator of $\mathbb{G}$ and $a, b, c \leftarrow_R \mathbb{Z}_q^*$. The (t, ϵ)-*DL *assumption holds in $\mathbb{G}$ if for given $(g, y) \in \mathbb{G}^2$, no t-time algorithm finds $x \in \mathbb{Z}_q$ such that $y = g^x$ with probability at least ϵ. The (t, ϵ)-*CDH *assumption holds in $\mathbb{G}$ if for given $(g, g^a, g^b) \in \mathbb{G}^3$, no t-time algorithm finds g^{ab} with probability at least ϵ. The (t, ϵ)-*DDH *assumption holds in $\mathbb{G}$ if no t-time algorithm has with at least ϵ advantage, where for an algorithm $\mathcal{A}$, $\mathcal{A}$'s advantage $\epsilon_{\mathcal{A}}$ is defined as $\epsilon_{\mathcal{A}} = \frac{1}{2}|\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c) = 1]|$. The (t, ϵ)-*GDH *assumption [22] holds in $\mathbb{G}$ if for given $(g, g^a, g^b) \in \mathbb{G}^3$, no t-time algorithm with polynomial bounded number of accesses to $\mathcal{O}$ finds g^{ab} with probability at least ϵ, where $\mathcal{O}$ is a decision oracle which for a given $(g, g^a, g^b, T) \in \mathbb{G}^4$, returns* 1 *if $T = g^{ab}$ or* 0 *otherwise.*

2.4 Some Variants of ElGamal Encryption

The signed ElGamal encryption [30,27] was proposed as a highly secure upgrade to the plain ElGamal encryption [12], which combines the ElGamal encryption with Schnorr signature [26]. Because of its good homomorphic property, the signed ElGamal encryption is widely used to construct threshold encryptions, with various applications like mix-net and auctions schemes. On the other hand, it has been a long-standing open problem to prove the security of the signed ElGamal encryption scheme based on reasonable assumptions. Jacobsson and Schnorr provided a security analysis of the signed ElGamal assuming both random oracles [8] and generic groups [28]. Another natural generalized version of the signed ElGamal was proposed in [27], namely the signed hashed-ElGamal, however, even the data privacy of this variant has not been well studied except brief discussions mentioned in [27,2]. We recall these schemes in Table 1. Let $\mathbb{G}$ be a group of prime order q, and let $g \in \mathbb{G}$ be a generator. Denote the public key as *pk* and the secret key as *sk*.

3 Key Privacy in the PCA Model

We sketch the intuition of the formalization before define key privacy in the PCA model, which is the first result of this work. We also give discussions on its reasonability,

Table 1. Some variants of the ElGamal encryption

	Plain ElGamal	**Hashed-ElGamal**	**Signed Hashed-ElGamal**
K	$x \leftarrow_R \mathbb{Z}_q$, $y \leftarrow g^x$. Public key: $pk \leftarrow g, y$, secret key: $sk \leftarrow x$. The message space is $\mathbb{G}$.	Let $H : \mathbb{G} \to \{0,1\}^\ell$ be a hash function. Choose $x \leftarrow_R \mathbb{Z}_q$, $y \leftarrow g^x$. Public key: $pk \leftarrow (g, y, H)$, Secret key: $sk \leftarrow x$. The message space is $\{0,1\}^\ell$.	$x \leftarrow_R \mathbb{Z}_q$, $y \leftarrow g^x$. Let $H_1 : \mathbb{G} \leftarrow \{0,1\}^\ell$ and $H_2 : \mathbb{G}^2 \times \{0,1\}^\ell \to \mathbb{Z}_q$ be two hash functions. Public key: $pk \leftarrow (g, y, H_1, H_2)$, Secret key: $sk \leftarrow x$. The message space is $\{0,1\}^\ell$.
E	For $m \in \mathbb{G}$, $r \leftarrow_R \mathbb{Z}_q$, $c_1 = g^r$, $c_2 = m \cdot y^r$. Ciphertext: $c = (c_1, c_2)$.	For $m \in \{0,1\}^\ell$, $r \leftarrow_R \mathbb{Z}_q$, $c_1 = g^r$, $c_2 = m \oplus H(y^r)$. Ciphertext: $c = (c_1, c_2)$.	For $m \in \{0,1\}^\ell$, $r, s \leftarrow_R \mathbb{Z}_q$, $c_1 = g^r$, $c_2 = m \oplus H_1(y^r)$. $d = H_2(g^s, c_1, c_2)$ and $z = s + rd$. Ciphertext: $c = (c_1, c_2, d, z)$.
D	For $c' = (c'_1, c'_2)$, output $m' = c'_2 / {c'_1}^x$ as the plaintext.	For $c' = (c'_1, c'_2)$, output $m' = c_2 \oplus H({c'_1}^x)$ as the plaintext.	For $c' = (c'_1, c'_2, d', z')$, test if $H_2(g^z {c'_1}^{-d'}, c'_1, c'_2) = d'$. If 'yes', output $m' = c'_2 \oplus H({c'_1}^x)$ as the plaintext; otherwise output "$\perp$".

because this attack model appears to be stronger than CPA and weaker than CCA in the sense of key privacy. In the end of this session, we further give further evidence that key privacy is orthogonal to data privacy.

Recall that the plaintext checking attack was only defined in the single public key setting, we first has to extend it to multiple public key setting and give the corresponding definition. The key observation is that a query to the plaintext checking oracle should be paired with a public key. In fact, this also fits the single public key model. Since there is only one public key, the public key is usually omitted in each plaintext checking query.

On the other hand, to make the indistinguishability meaningful in PCA model, for the target ciphertext queried with one of the chosen public keys, the plaintext checking oracle returns a special symbol "$\perp$" (cf. replayable chosen ciphertext attack RCCA [9]). With such limitation to an oracle access, the plaintext checking oracle may no longer trivially leak information regarding the public key.

Definition 7 (IK-PCA). *Let $\Pi = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ be a public key encryption scheme. Let $\mathcal{A}$ be an adversary, and $\mathcal{PCO}$ be a plaintext checking oracle with limitations as discussed above. We define the advantage of $\mathcal{A}$ as*

$$Adv_{\Pi,\mathcal{A}}^{\text{ik-pca}}(k) = \Pr\left[b' = b \,\middle|\, \begin{array}{c} (pk_0, sk_0) \leftarrow \mathsf{K}(1^k); (pk_1, sk_1) \leftarrow \mathsf{K}(1^k); \\ (m^*, s) \leftarrow \mathcal{A}^{\mathcal{PCO}}(pk_0, pk_1); b \leftarrow_R \{0,1\}; \\ c^* \leftarrow \mathsf{E}(pk_b, m^*); b' \leftarrow \mathcal{A}^{\mathcal{PCO}}(s) \end{array} \right] - \frac{1}{2}$$

A scheme is (ϵ, t)-IK-PCA secure if any adversary with running time t, has advantage at most ϵ in winning the above game. Furthermore, we say a public key encryption scheme is IK-PCA secure if for any PPT adversary, ϵ is negligible.

For the security goal of key privacy, an adversary will not necessarily gain any direct "help" via its access to a plaintext checking oracle, thus plaintext checking attack (PCA) may be meaningful in this setting. However, the plaintext to query the plaintext checking oracle may depend on the public key, thus it may be a stronger attack even in the sense of key privacy. Based on above discussions we prove the following theorem:

Theorem 1. *There is a public key encryption scheme that is* IK-CPA *secure but not* IK-PCA *secure.*

Proof. Consider the plain ElGamal encryption. It is known from [4] that under the DDH assumption, it is IK-CPA. It is sufficient to show it is not IK-PCA.

To see this, we construct an IK-PCA adversary that defeats the ElGamal encryption. After setups, the IK-PCA adversary chooses $m^* \leftarrow_R \mathbb{G}$, and submits to the challenger. The challenger chooses $b \leftarrow_R \{0, 1\}$, and returns the challenge ciphertext $c^* = (c_1, c_2) = (g^r, m^* \cdot y_b^r)$, where $r \leftarrow_R \mathbb{Z}_q$. Now the adversary just needs to query $(2 \cdot m^*, c')$ with $c' = (c_1, 2 \cdot c_2)$ to the plaintext checking oracle on public key y_1. If $b = 1$, the pair will be always valid and 0 otherwise, so the adversary always breaks the indistinguishability of keys of the scheme. □

One may argue that in the above attack we are tricky because if we assume the existence of plaintext checking oracle, the plain ElGamal encryption no longer has data privacy. However, we are considering key privacy, and a scheme with key privacy can in fact has no data privacy at all. To show that IK-PCA is a proper notion which lies between IK-CPA and IK-CCA, we present the following theorem:

Theorem 2. *There is a public key encryption scheme that is* IK-PCA *secure, but not* IK-CCA *secure.*

Proof. Consider the hashed-ElGamal encryption. We claim it is not IK-CCA secure. To see this, we construct an IK-CCA adversary as follows: After setup, the adversary is given the public key $pk_0 = (g, y_0, H_1)$ and $pk_1 = (g, y_1, H_2)$. [2] The adversary then chooses $m^* \leftarrow_R \{0, 1\}^\ell$, and submits it to the challenger. The challenger chooses $b \leftarrow_R \{0, 1\}$ and returns $c^* = (c_1, c_2) = (g^r, m^* \oplus H_b(y_b^r))$. Now the adversary just need to query (c', pk_1) to the decryption oracle, where $c' = (c_1, c_2 \oplus \delta)$ and $\delta \leftarrow_R \{0, 1\}^\ell$. If the decryption oracle returns $m^* \oplus \delta$, then the adversary output "1"; otherwise "0".

It is easy to verify that c' is a valid ciphertext according to pk_b and $m^* \oplus \delta$. Therefore, we conclude the adversary guess the value of b with probability "1".

Next, we shall prove the hashed-ElGamal is IK-PCA under gap Diffie-Hellman (GDH) assumption. Here for simplicity, we consider the case that H_1 and H_2 are the same random oracle. The basic idea is that since the information of the public key can only be accessed via hash queries, thus any adversary that breaks the IK-PCA security of the scheme must have queried to the random oracle regarding the chosen public key. On the other hand, given a GDH problem instance, utilizing the random self-reducibility of the GDH problem, the simulator can create two independent public keys that distributes identically as the real attack.

We write above idea in more details. A GDH adversary $\mathcal{A}$ can be constructed as follows: for a given GDH instance $(A = g, B = g^a, C = g^b)$, where a, b are unknown, $\mathcal{A}$ then interacts with the IK-PCA adversary $\mathcal{B}$ as follows:

[2] H_1 and H_2 can be of different implementations, as long as their input and output domain are the same.

$\mathcal{A}$ gives $pk_0 = (A, B^{r_0}, H)$ and $pk_1 = (A, B^{r_1}, H)$ to $\mathcal{B}$ as two public keys, where $r_0, r_1 \leftarrow_R \mathbb{Z}_q$ and H is a random oracle controlled by $\mathcal{A}$. Since r_0 and r_1 are uniformly distributed in $\mathbb{Z}_q$, then the simulated public keys pk_0 and pk_1 are indistinguishable from the real game where $pk_0 = (A, y_0, H)$ and $pk_1 = (A, y_1, H)$. Moreover, $\mathcal{A}$ simulates the following oracles:

Random Oracle queries: $\mathcal{A}$ maintains a H-list with two entries (h_i, H_i). Where a query on h_i comes, $\mathcal{A}$ queries its own decision oracle whether (A, B, y_δ, h_i) $(\delta = 0, 1)$ is a Diffie-Hellman tuple. If "yes", $\mathcal{A}$ stops simulation and output h_i^{1/r_δ} as its answer. Otherwise, $\mathcal{A}$ searches H-list, if there exists such an entry, $\mathcal{A}$ returns the corresponding H_i. Otherwise, $\mathcal{A}$ then chooses $H_i \leftarrow_R \{0, 1\}^\ell$, adds (h_i, H_i) to the list and returns H_i to $\mathcal{B}$.

Plaintext Checking queries: On a query (m, c) on public key pk_δ $(\delta \in \{0, 1\})$, where $c = (c_1, c_2)$, $\mathcal{A}$ searches in the H-list whether there exists an entry (h_i, H_i) such that (g, c_1, y_δ, h_i) is a Diffie-Hellman tuple (this can be done via querying its own decision oracle) and $c_2 = m \oplus H_i$. If both of above hold, return "yes". Otherwise "no".

Challenge query: When $\mathcal{B}$ signals a chosen message m^* to $\mathcal{A}$, on which it would like to be challenged, $\mathcal{A}$ chooses a bit $\beta \leftarrow_R \{0, 1\}$ and $R \leftarrow_R \{0, 1\}^\ell$, set $c^* = (B, m \oplus R)$ and returns the challenge c^* to $\mathcal{B}$.

Since the information of β can only be gained by querying g^{abr_β} to the random oracle. From above descriptions one easily verifies that $\mathcal{A}$ answers the random oracle, the plaintext checking and the challenge queries perfectly. Then $\mathcal{B}$'s success probability is exactly the same as the real attack. Combining all above discussions, $\mathcal{A}$ succeeds with probability at least $\epsilon_{\mathcal{B}}$. □

Furthermore, we would like to present an important claim of our paper, namely, data privacy and key privacy are totally *unrelated*.

Theorem 3. *There is a public key encryption scheme that is* IK-ATK *secure but not* IND-ATK *secure; there is a public key encryption scheme that is* IND-ATK *secure but not* IK-ATK *secure, where* ATK $\in$ {CPA, CCA}.

Proof. We only give sketch of the proof, since it is not difficult to write a long but more precise (though tedious) one. To see the first half of the statement, given an encryption scheme with both IND-ATK and IK-ATK security (in short IND-IK-ATK), e.g., the Cramer-Shoup [10], we can modify it to an IK-ATK secure encryption yet without data privacy. All will remain the same, except that for the encryption algorithm, we append the plaintext m to the ciphertext c. Now it is easy to see, the modified encryption scheme is not IND-ATK. On the other hand, since the new encryption algorithm does not produce ciphertexts that give more information on the public key than the original encryption scheme, it is still IK-ATK secure.

To see second half of the statement, we can do similar as above: for an IND-IK-ATK secure encryption scheme, just append the public key to the ciphertext. It is easily verified then this modified encryption scheme has no key privacy but is still IND-ATK secure. □

4 Generic Transforms for Key Privacy

We here also discuss some generic transforms to enhance the chosen ciphertext security of an encryption scheme regarding key privacy. Informally and briefly speaking, for an IK-CPA secure encryption, to enhance it to an IK-CCA secure one, one just needs to additionally simulate the decryption oracle. This is quite difficult in the standard model, since the simulator has to do decryptions without the secret keys. The classic Naor-Yung paradigm is sure to apply here, as long as the new components introduced to the encryption scheme do not leak any information of the public key.

In the random oracle model, however, this can be done easily and efficiently. A recent work formulates the plaintext awareness in the two public key model and use the plaintext extractor provided via plaintext awareness to simulate the decryption oracle. Based on this idea, some famous generic transforms to enhance data privacy, like (two) Fujisaki-Okamoto (FO) [14,15] and REACT [21] transforms are all capable to construct a plaintext extractor, which can be used to for simulation of the decryption oracle.

Recall the construction of such plaintext extractors in the original papers. It is not difficult to rewrite the proofs for the two public key setting just like what has been done in [18] via the FOcrypto transform [15]. In fact, the theorem attests exactly our claim that key privacy is unrelated to chosen ciphertext security.

Theorem 4. *The FOpkc and REACT transform are both capable to enhance* IK-CPA *security to* IK-CCA *security.*

For the limit of space, some related definitions and the detailed proof is left to the full version of this paper [31]. Theorem 4 tells that FOpkc and REACT are both generic for key privacy enhancement, however, it is worth noting that "unnatural" implementations may still leak information of keys. In fact, our proofs require the symmetric key encryption should have key privacy. However, unlike public key encryption, the semantic security of implies its key privacy. For completeness, we review the model and the notion of key privacy for symmetric key encryption in Appendix A.

5 Analysis on Signed Hashed-ElGamal Encryption

In this section we give analyses on data privacy and key privacy of the signed hashed-ElGamal encryption. Since the data privacy (IND-CCA) has been studied [27,2] before, however, without details, we give a brief but much specific proof. We focus on non-plaintext awareness and the key privacy properties of the scheme, namely we prove the following theorem.

Theorem 5. *The signed hashed-ElGamal is* IND-CCA *and* IK-CCA *secure and not PA.*

We split the proof into three parts, each shown by a lemma. The theorem then follows Lemma 1, 2 and 3. For data privacy, we first review a previous result on the security of the Schnorr signature in Proposition 1. Since the GDH assumption implies the DL assumption, combining Proposition 1 and Lemma 1, we conclude signed hashed-ElGamal is secure under the GDH assumption.

Proposition 1. *([24]) The Schnorr Signature is* (ϵ_s, t_1)-sUF-CMA *secure, assuming the discrete logarithm problem in* G *is* $(120686 q_h t_1, 10(q_s + 1)(q_s + q_h)/q)$*-hard, where* q_h *is number of random oracle queries,* q_s *is the number of signing queries and* q *is the order of subgroup* G.

Lemma 1. *The signed hashed-ElGamal is* $(\epsilon + \epsilon_s, (t_1 + t_2 + O(k)))$-IND-CCA *secure, assuming the* (ϵ, t_2)*-GDH assumptions holds and the Schnorr Signature is* (ϵ_s, t_1)-sUF-CMA *secure.*

Proof Ideas. To show a more general result, we present the following encryption scheme. For the hashed-ElGamal encryption, we append to it an $(\epsilon_s, 1, t_s)$-sUF-CMA secure signature scheme. The Schnorr signature satisfies such security dentition according to [23]. We describe the algorithms of such encryption scheme. The signature scheme used in signed hashed-ElGamal is a special case by using Schnorr Signature [26] and shown in Figure 1.

The key generation just runs the key generation algorithm of the hashed-ElGamal encryption, the public key and secret key are (pk, sk) as before. That is, $pk = (g, y, H)$ and $sk = (x, H)$, such that $y = g^x$. We note that additional hash functions may be used, then also include the hash functions into the public key.

The encryption algorithm first calls the key generation algorithm of the signature scheme, generates a (one-time) verification/signing key pair (vk, sk). Then it does just as the hash ElGamal encryption, except that it also bundles the public key of the signature scheme into the a hash function, namely, it produces $c = (c_1, c_2)$, where $c_1 = g^r$ is the same as before, however, $c_2 = H(vk, y^r)$. Then a signature σ is produced on c. The ciphertext is (c, σ, vk). Note that if we use Schnorr signature, then we get a very efficient scheme: the verification key is not necessary to be explicitly included into the ciphertext, since it is exactly c_1. In following proofs, we will assume vk is not compressible.

The decryption algorithm first verifies the signature σ is a correct signature on c according to vk. If this fails, it halts and output "$\perp$". Otherwise, it proceeds to undo the hashed-ElGamal encryption using the decryption algorithm of hashed-ElGamal encryption as shown in Table 1.

Proof. We next show the above encryption scheme is $(\epsilon + \epsilon_s, t_1 + t_2 + O(k))$-IND-CCA secure, if the (ϵ, q, t_1)-GDH assumption holds and the signature scheme used is $(\epsilon_s, 1, t_2)$-sUF-CMA secure. Towards the claim, suppose a GDH solver $\mathcal{A}$ interacts with an IND-CCA adversary $\mathcal{B}$ as follows.

Key Generation: $\mathcal{A}$ receives its GDH problem instance $(A = g, B = g^a, C = g^b)$, and sets $y = C^{r^*}$ and $pk = (g, y, H)$, where $r^* \leftarrow_R \{0, 1\}^\ell$ and H is a random oracle controlled by $\mathcal{A}$. $\mathcal{A}$ gives pk to $\mathcal{B}$.

Random Oracle queries: $\mathcal{A}$ maintains a H-list with four entries (v_i, c_{1i}, h_i, H_i). Where a query comes for (v_i, h_i), $\mathcal{A}$ searches H-list, if there exists such an entry $(v_i, \cdot, h_i, H_i)$, where "$\cdot$" means "not of interest", $\mathcal{A}$ returns the corresponding H_i. Otherwise, $\mathcal{B}$ chooses $H_i \leftarrow_R \{0, 1\}^\ell$ and adds (v_i, c_{1i}, h_i, H_i) to the list. Additionally, if there exists an entry with $(v_i, c_{1i}, *, H_i)$ in the list, where $*$ mains empty entry, and $(A, c_{1i}, C^{r^*}, h_i)$ is a Diffie-Hellman tuple, $\mathcal{A}$ completes the entry with (v_i, c_{1i}, h_i, H_i). In both cases, $\mathcal{A}$ returns H_i to $\mathcal{B}$.

Decryption queries: On a decryption query on (c, σ, vk), $\mathcal{A}$ first verifies σ is a valid signature on c under vk. If this fails, $\mathcal{A}$ simply returns "$\perp$". Otherwise, $\mathcal{A}$ searches for the $\mathtt{H}$-list. $\mathcal{A}$ splits $c = (c_1, c_2)$, if there exists an entry with (vk, c_1, h_i, H_i), such that (g, c_1, y, h_i) is a DH-tuple, $\mathcal{A}$ then returns $H_i \oplus c_2$ as the plaintext. Otherwise, if (vk, c_1, h_i) is not queried before, $\mathcal{A}$ chooses $H_i \leftarrow_R \{0,1\}^\ell$ adds $(vk, c_{1i}, *, H_i)$ to $\mathtt{H}$-list and returns $H_i \oplus c_2$ as the plaintext.

Challenge query: When $\mathcal{B}$ outputs a pair of chosen message (m_0, m_1), $\mathcal{A}$ chooses $\beta \leftarrow_R \{0,1\}$ and $R^* \leftarrow_R \{0,1\}^\ell$ and sets $c^* = (B, m_\beta \oplus R)$. $\mathcal{A}$ also queries its own signing oracle on c^* under vk^*. After the signing oracle $\mathcal{O}_s$ returns a signature σ^* on c^*, $\mathcal{A}$ forwards (c^*, σ^*, vk^*) to $\mathcal{B}$ as the challenge ciphertext. Additionally, $\mathcal{A}$ adds $(vk^*, B, *, R)$ to $\mathtt{H}$-list.

After $\mathcal{B}$ submits a guess on β, $\mathcal{A}$ searches $\mathtt{H}$-list for $(\cdot, B, h_i, \cdot)$ (recall $\cdot$ means "not of interest"), such that $(A, B, C, h_i^{1/r^*})$ is a Diffie-Hellman tuple. If there does not exist such a tuple, $\mathcal{A}$ chooses $D \leftarrow_R \mathbb{G}$ and halts.

From above descriptions, we can verify the simulations of random oracle, decryption oracle and challenge oracle are perfect. Since the information of β is perfectly hiding without querying g^{abr^*} to the random oracle. Then if $\mathcal{B}$ gainst any advantage on guessing β correctly, $\mathcal{B}$ must have queried $h_i = g^{abr^*}$ already, except that $\mathcal{B}$ queries a decryption query on (c^*, σ') with $\sigma' \neq \sigma^*$, where σ' is a valid signature on c^* regarding vk^*, however, in this case, $\mathcal{B}$ has already broken the signature scheme, thus this event happens at most ϵ_s. Note that in above proof we only require $\mathcal{A}$ queries the signing oracle exactly once. This proves our claim on success probability of $\mathcal{A}$. The running time of $\mathcal{A}$ is verifiable from the above descriptions. □

Lemma 2. *The signed hashed-ElGamal is* $(\epsilon + \epsilon_s, (t_1 + t_2 + O(k)))$-IK-CCA *secure, assuming the* (ϵ, q, t_2)*-GDH assumptions holds and the Schnorr signature is* (ϵ_s, t_1)-sUF-CMA *secure.*

Proof Ideas. The idea here works as a combination of those of IK-PCA security of hashed-ElGamal and IND-CCA of signed hashed-ElGamal shown before. Basically $\mathcal{A}$ randomizes its GDH challenge into two public keys and interacts with a public key distinguisher $\mathcal{B}$. As the signature scheme does not leak information on the public key, the only way to distinguish which public key was used to compute the challenge is by directly querying some necessary information at the random oracle. Thus $\mathcal{A}$ can successfully extracts the answer for its own GDH problem.

Proof. $\mathcal{A}$ interacts with $\mathcal{B}$ in the following manner:

Key Generation: $\mathcal{A}$ receives its GDH problem instance $(A = g, B = g^a, C = g^b)$. $\mathcal{A}$ sets $pk_0 = (g, C^{r_0^*}, H)$ and $pk_1 = (g, C^{r_1^*}, H)$, where $r_0^*, r_1^* \leftarrow_R \{0,1\}^\ell$ and H is a random oracle controlled by $\mathcal{A}$ and gives pk_0 and pk_1 to $\mathcal{B}$.

Random Oracle queries: $\mathcal{A}$ maintains a $\mathtt{H}$-list with four entries (v_i, c_{1i}, h_i, H_i). Where a query comes for (v_i, h_i), $\mathcal{A}$ searches $\mathtt{H}$-list, if there exists such an entry with $(v_i, \cdot, h_i, H_i)$, $\mathcal{A}$ returns the corresponding H_i. Again "$\cdot$" means "not of interest". Otherwise, $\mathcal{B}$ chooses $H_i \leftarrow_R \{0,1\}^\ell$ and adds $(v_i, *, h_i, H_i)$ to the list. Here $*$ mean empty. Additionally, if there exists an entry with $(v_i, c_{1i}, *, H_i)$ in the list, where $*$

mains empty entry, and $(A, c_{1i}, C^{r_0^*}, h_i)$ or $(A, c_{1i}, C^{r_1^*}, h_i)$ is a Diffie-Hellman tuple, $\mathcal{A}$ completes the entry with (v_i, c_{1i}, h_i, H_i). In both cases, $\mathcal{A}$ returns H_i to $\mathcal{B}$.

Decryption queries: On a decryption query on (c, σ, vk) regarding pk_δ ($\delta \in \{0, 1\}$), $\mathcal{A}$ first verifies σ is a valid signature on c under vk. If this fails, $\mathcal{A}$ simply returns "$\perp$". Otherwise, $\mathcal{A}$ splits $c_i = (c_{1i}, c_{2i})$, searches in the H-list. If there exists an entry with (vk, c_{1i}, h_i, H_i), such that $(A, c_{1i}, C^{r_\delta^*}, h_i)$ is a DH-tuple, $\mathcal{A}$ then returns $H_i \oplus c_2$ as the plaintext. Otherwise, if (vk, c_{1i}, h_i) is not queried before, $\mathcal{A}$ chooses $H_i \leftarrow_R \{0, 1\}^\ell$, adds $(vk, c_{1i}, *, H_i)$ to H-list and returns $H_i \oplus c_2$ as the plaintext.

Challenge query: When $\mathcal{B}$ outputs a chosen message m^*, $\mathcal{A}$ chooses $\beta \leftarrow_R \{0, 1\}$ and $R^* \leftarrow_R \{0, 1\}^\ell$ and sets $c^* = (B, m \oplus R)$. $\mathcal{A}$ also queries its own signing oracle on c^* under vk^*. After the signing oracle O_s returns a signature σ^* on c^*, $\mathcal{A}$ forwards (c^*, σ^*, vk^*) to $\mathcal{B}$ as the challenge ciphertext.

After $\mathcal{B}$ submits a guess on β, $\mathcal{A}$ searches H-list for $(\cdot, B, h_i, \cdot)$, such that $(A, B, C, h_i^{1/r_\beta^*})$ is a Diffie-Hellman tuple. If there does not exist such a tuple, $\mathcal{A}$ returns $D \leftarrow_R \mathbb{G}$ and halts.

With a similar discussion as before, we remark that the simulations of random oracle, decryption oracle and challenge oracle are again perfect. However, $\mathcal{A}$ may not utilize the hash queries submitted by $\mathcal{B}$, if (c^*, σ', vk^*) is submitted for decryption query and σ' is a valid signature on c^* under verification key vk^*. However, this event happens with probability at most ϵ_s, because the signature scheme is (ϵ_s, t_2)-sUF-CMA secure. (In above proof we only require $\mathcal{A}$ queries the signing oracle exactly once.) Thus $\mathcal{A}$ succeeds solving the GDH problem with probability at least $\epsilon - \epsilon_s$. The running time of $\mathcal{A}$ is $(t_1 + t_2 + O(k))$, which can be verified from the description of $\mathcal{A}$. □

Lemma 3. *The signed hashed-ElGamal is not PA.*

Proof Sketch. We give the idea of the proof here. An adversary $\mathcal{A}$ can produce a valid ciphertext $(c, \sigma) = (c_1, c_2, \sigma)$ where $c_1 = g^r$ and $c_2 \leftarrow_R \{0, 1\}^\ell$, σ is a valid signature on c. Especially, $\mathcal{A}$ construct the ciphertext without querying $(vk, c_1, *)$ to the random oracle H. To extract the plaintext m, even if the plaintext extractor has all the random oracle queries of $\mathcal{A}$, since $(vk, c_1, *)$ hasn't been queried before, if the plaintext extractor can be constructed, i.e., $\mathsf{D}^H((sk, c_1, c_2, \sigma)$ can be constructed on the transcript given, but the existence of this knowledge extractor will contradicts the unpredictability of H.

Proof. For the signed hashed-ElGamal encryption, if the plaintext extractor can be constructed, that is, we can set $H(\cdot)$ to be $m \oplus c_2$. As mentioned above, H is a random oracle whose output is uniformly random, and c_2 is fixed already, thus m can only be extracted with probability at most $2^{-\ell}$, which is negligible. contradicting the definition of plaintext extractor. □

References

1. Abdalla, M., Bellare, M., Rogaway, P.: DHIES: An Encryption Scheme Based on the Diffie-Hellman Problem. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)
2. Abe, M.: Combining Encryption and Proof of Knowledge in the Random Oracle Model. The Computer Journal 47(1), 58–70 (2004)

3. An, J.H., Dodis, Y., Rabin, T.: On the Security of Joint Signature and Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)
4. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-Privacy in Public-Key Encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001)
5. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998)
6. Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption without Random Oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)
7. Bellare, M., Rogaway, P.: Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In: CCS 1993, pp. 62–73. ACM Press, New York (1993)
8. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption – How to encrypt with RSA. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
9. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing Chosen-Ciphertext Security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003)
10. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
11. Dolev, D., Dwork, C., Naor, M.: Non-Malleable Cryptography. In: STOC 1991, pp. 542–552. ACM, New York (1991)
12. ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEE Transactions on Information Theory 31(4), 469–472 (1985)
13. Fujisaki, E.: Plaintext Simulatability. IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences E-89A(1), 55–65 (2006)
14. Fujisaki, E., Okamoto, T.: How to Enhance the Security of Public-Key Encryption at Minimum Cost. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 53–68. Springer, Heidelberg (1999)
15. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–544. Springer, Heidelberg (1999)
16. Goldwasser, S., Micali, S.: Probabilistic Encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
17. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal on Computing 17(2), 281–308 (1988)
18. Hayashi, R., Tanaka, K.: PA in the Two-Key Setting and a Generic Conversion for Encryption with Anonymity. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 271–282. Springer, Heidelberg (2006)
19. Herzog, J., Liskov, M., Micali, S.: Plaintext Awareness via Key Registration. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 548–564. Springer, Heidelberg (2003)
20. Naor, M., Yung, M.: Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In: STOC 1990, pp. 427–437. ACM, New York (1990)
21. Okamoto, T., Pointcheval, D.: REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (2001)
22. Okamoto, T., Pointcheval, D.: The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)

23. Pointcheval, D., Stern, J.: Security Proofs for Signature Schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
24. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology 13(3), 361–396 (2000)
25. Rackoff, C., Simon, D.R.: Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In: Crypto 1991. LNCS, vol. 567, pp. 433–444. Springer, Heidelberg (1991)
26. Schnorr, C.P.: Efficient Signature Generation by Smart Cards. Journal of Cryptology 4(3), 161–174 (1991)
27. Schnorr, C.P., Jakobsson, M.: Security of Signed ElGamal Encryption. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, Springer, Heidelberg (2000)
28. Shoup, V.: Lower Bounds for Discrete Logarithms and Related Problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
29. Steinfeld, R., Baek, J., Zheng, Y.: On the Necessity of Strong Assumptions for the Security of a Class of Asymmetric Encryption Schemes. In: Batten, L.M., Seberry, J. (eds.) ACISP 2002. LNCS, vol. 2384, pp. 241–256. Springer, Heidelberg (2002)
30. Tsiounis, Y., Yung, M.: On the Security of El Gamal based Encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 117–134. Springer, Heidelberg (1998)
31. Zhang, R., Hanaoka, G., Imai, H.: Orthogonality between key privacy and data privacy, revisited. Full version of this paper (2007), `http://staff.aist.go.jp/r-zhang/research/papers.html#KeyPrivacy`

A Symmetric Key Encryption

A (deterministic) symmetric key encryption (SKE) consists of two algorithms $\Phi = (\mathsf{Enc}, \mathsf{Dec})$. The encryption algorithm Enc with input a secret key $w \in \{0,1\}^{\ell_0}$ and a plaintext $m \in \{0,1\}^{\ell_1}$, outputs a ciphertext c, denoted as $c \leftarrow \mathsf{Enc}(w, m)$. The decryption algorithm Dec with input $w \in \{0,1\}^{\ell_0}$ and the ciphertext c, outputs a plaintext $m \in \{0,1\}^{\ell_1}$, denoted as $m \leftarrow \mathsf{Dec}(w, c)$.

Definition 8 (Key Privacy for SKE). *Let $\Phi = (\mathsf{Enc}, \mathsf{Dec})$ be a SKE scheme. Let $\mathcal{A}$ be an adversary and $\mathcal{EO}$ be an encryption oracle that on a plaintext query (w, β) where $\beta \in \{0,1\}$, returns a corresponding ciphertext c_β under the secret key sk_β. Denote the advantage of $\mathcal{A}$ in the following game:*

$$Adv^{\mathsf{ik\text{-}cpa}}_{\Phi,\mathcal{A}}(k) = \Pr\left[b' = b \left| \begin{array}{c} w_0 \leftarrow_R \{0,1\}^{\ell_0}; w_1 \leftarrow_R \{0,1\}^{\ell_0}; \\ (m^*, s) \leftarrow \mathcal{A}^{\mathcal{EO}}(1^{\ell_1}); b \leftarrow_R \{0,1\}; \\ c^* \leftarrow \mathsf{Enc}(w_b, m^*); b' \leftarrow \mathcal{A}^{\mathcal{EO}}(c^*, s) \end{array} \right. \right] - \frac{1}{2}$$

Φ is (ϵ, t)-$\mathsf{IK\text{-}CPA}$ secure if any adversary with running time t, gains advantage at most ϵ in winning the above game. Furthermore, we say a symmetric key encryption scheme is $\mathsf{IK\text{-}CPA}$ secure if for any PPT adversary, ϵ is negligible.

For a deterministic symmetric key encryption scheme, key privacy is quite related to the length of the secret key rather than the space of the randomness for a public key encryption scheme.

Unlinkable Randomizable Signature and Its Application in Group Signature[⋆]

Sujing Zhou[1,2] and Dongdai Lin[1]

[1] SKLOIS Lab, Institute of Software, Chinese Academy of Sciences, Beijing, P.R. China, 100080
[2] Graduate School of the Chinese Academy of Sciences, Beijing, P.R. China, 100039
{zhousujing,ddlin}@is.iscas.ac.cn

Abstract. We formalize a generic method of constructing efficient group signatures, specifically, we define new notions of unlinkable randomizable signature, indirectly signable signature and Σ-protocol friendly signature.

We conclude that designing efficient secure group signatures can be boiled down to designing ordinary signatures satisfying the above three properties, which is supported by observations that almost all currently known secure efficient group signatures have alternative constructions in this line without deteriorating the efficiency.

Keywords: Digital Signature, Group Signature, Randomizable Signature, Sigma-protocol.

1 Introduction

In brief, a group signature scheme is composed of the following steps: (1) GM, the group manager, along with some third trusted party, chooses the security parameters as well as a group secret key and a group public key. (2) Any group member candidate is required to choose his *member secret key*, and run an interactive protocol with GM to join in the group, during which GM generates a signature on the member secret key blindly, i.e., not knowing the secret key value, the signature is also called *member certificate*. (3) Any group member can generate group signatures using his *group signing key* which includes member secret key and member certificate.

A common paradigm of constructing group signatures [1,2,3,4] is as follows: GM adopts an ordinary signature scheme to generate membership certificate for group members, i.e., sign on some secret key known only to members. The group signature is in fact a non-interactive zero-knowledge proof of knowledge of member certificate and member secret key, transformed in Fiat-Shamir's heuristic method [5] from interactive proofs.

⋆ This work is partly supported by the National Natural Science Foundation of China(No. 60673069), National 863 Project of China(No. 2007AA01Z447) and National 973 Project(No. 2004CB318004).

Recently, a kind of randomizable signatures (given a signature of a message, someone other than the signer can get a new signature with respect to the same message) have been adopted in some schemes [6,7,8,9] to generate membership certificates. The following construction of group signature has been widely recognized: to sign on a message, a member firstly randomizes his member certificate, then generates a proof of knowledge of member secret key and part of the randomized member certificate. This method might result in more efficient group signature because the relation between member secret key and other items is much simplified due to concealing only part of the randomized member certificate instead of concealing it all in previous constructions.

We formalize the characteristics of randomizable signatures that are required to build secure efficient group signatures. Specifically, we define new notions of unlinkable randomizable signature, indirectly signable signature, Σ-protocol friendly signature.

We conclude that designing efficient secure group signatures can be boiled down to designing ordinary signatures satisfying the above three properties, which is supported by observations that almost all currently known secure efficient group signatures (except [10]) have alternative constructions in this line without deteriorating the efficiency, i.e., the signature schemes used to generate member certificates in the group signature can be modified into randomizable signatures with unlinkability, indirectly signability and Σ-protocol friendliness. For example, the scheme in [7] can be seen as the randomizable version of the well known ACJT scheme [4], satisfying the above three characteristics.

Apart from pointing out the obvious alternative constructions of some current group signatures, we propose the more complicated alternative constructions of others. They include the alternative construction of the scheme [11] from randomizable signatures (denoted as NSN04*). We propose two new randomizable signatures (denoted Wat05+, ZL06+) resulting in new efficient group signatures. We also slightly improve the scheme with concurrent join [12] by replacing the member certificate generation signature with an randomizable signature (denoted as BBS04+).

Organization. The new notions of unlinkable randomizable signature, indirectly signable signature, Σ-protocol friendly signature are presented in Section 3, where you can also find the new randomizable signatures satisfying the above three properties: NSN04*, Wat05+, ZL06+. A generic construction of group signatures from the above randomizable signature is described in Section 4.1 as well as its security analysis (Section 4.2). We present the slight improvement to the group signature with concurrent join [12] in Section 4.3.

2 Preliminary

Notations. If (P, V) is a non-interactive proof for relation ρ, $P(x, w, R)$ denotes the operation of generating a proof for $(x, w) \in \rho$ under the common reference string R, $V(x, \pi, R)$ denotes the operation of verifying a proof π.

Definition 1 (wUF-ATK[13]). *A signature scheme DS=(Gen, Sig, Ver) is wUF-ATK secure (ATK ∈ {CMA, ACMA}), i.e., weakly unforgeable against ATK attack, if for every probabilistic polynomial-time algorithm $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{wUF-ATK}_{DS,\mathcal{A}} = \mathsf{Pr}\{(pk, sk)\leftarrow Gen(1^k), (m, \sigma)\leftarrow \mathcal{A}^{\mathcal{O}_{Sig(sk,.)}}(pk, ATK) : Ver(pk, m, \sigma) = 1, m \notin Q\} < \epsilon(k)$$

where $\epsilon(k)$ is a negligible function, the probability is taken over the coin tosses of algorithms Gen, Sig and $\mathcal{A}$. Q denotes the set of queries to oracle $\mathcal{O}_{Sig(sk,.)}$ made by $\mathcal{A}$.

3 The New Notions

3.1 Unlinkable Randomizable Signature (URS)

Definition 2 (Randomizable Signature). *A randomizable signature scheme is a digital signature scheme that has an efficient signature randomization algorithm Rnd besides algorithms (Gen,Sig,Ver):*

- *Gen: $N\rightarrow K$: a probabilistic polynomial-time algorithm with input k (called security parameter), output $(pk, sk) \in K$, where K is a finite set of possible keys; pk is called public key, sk is secret key kept to the signer, i.e., the owner of the instance of the signature scheme.*
- *Sig: $K\times M\rightarrow S$: a probabilistic polynomial-time algorithm with input (sk, m), where sk is the same output from K above, $m \in M$, M is a finite set of possible messages. Output is $\sigma = (\Upsilon, \Xi) \in S$, where Υ is randomly chosen and independent from m, Ξ is calculated from Υ, m and sk.*
- *Ver: $K\times M\times S\rightarrow\{0, 1\}$: a deterministic polynomial-time algorithm with input (pk, m, σ), output 1 if σ is valid, i.e., σ is really computed by the owner of the signature instance, output 0 otherwise.*
- *Rnd: $M \times S \rightarrow S$: a probabilistic polynomial-time algorithm with a message m and a signature (Υ, Ξ) on it, output a $(\Upsilon', \Xi') \neq (\Upsilon, \Xi)$ that is also a signature on m.*

$\underline{\mathsf{Exp}^{unlink-b}_{\mathcal{A}}(k)}$, $b \in \{0, 1\}$: $(pk, sk) \overset{\$}{\leftarrow}$ Gen(1^k), $(m_0, \Upsilon_0, \Xi_0, m_1, \Upsilon_1, \Xi_1) \overset{\$}{\leftarrow} \mathcal{A}(sk, pk)$, If Ver$(pk, m_0, \langle\Upsilon_0, \Xi_0\rangle) = 0$ or Ver$(pk, m_1, \langle\Upsilon_1, \Xi_1\rangle) = 0$, return 0. $(\Upsilon', \Xi') \overset{\$}{\leftarrow}Rnd(m_b, \Upsilon_b, \Xi_b)$, $b' \leftarrow \mathcal{A}(sk, pk, \Xi')$. return b'.

Definition 3 (Perfectly Unlinkable). *A randomizable signature rDS = (Gen, Sig, Ver, Rnd) is perfectly unlinkable if for any algorithm $\mathcal{A}$, the distribution of output of $\mathsf{Exp}^{unlink-b}_{\mathcal{A}}(k)$ (defined above) are the same for $b \in \{0, 1\}$, that is*

$$\mathsf{Pr}\{\mathsf{Exp}^{unlink-1}_{\mathcal{A}}(k) = 1\} = \mathsf{Pr}\{\mathsf{Exp}^{unlink-0}_{\mathcal{A}}(k) = 1\}.$$

The above equation is identical to

$$\mathsf{Pr}\{\Xi' \overset{\$}{\leftarrow} Rnd(m_1, \Upsilon_1, \Xi_1)|(pk, sk) \overset{\$}{\leftarrow} Gen(1^k), (\langle m_0, \Upsilon_0, \Xi_0\rangle, \langle m_1, \Upsilon_1, \Xi_1\rangle) \overset{\$}{\leftarrow} \mathcal{A}(sk)\}$$
$$= \mathsf{Pr}\{\Xi' \overset{\$}{\leftarrow} Rnd(m_0, \Upsilon_0, \Xi_0)|(pk, sk) \overset{\$}{\leftarrow} Gen(1^k), (\langle m_0, \Upsilon_0, \Xi_0\rangle, \langle m_1, \Upsilon_1, \Xi_1\rangle) \overset{\$}{\leftarrow} \mathcal{A}(sk)\}.$$

Definition 4 (Statistically Unlinkable). *A randomizable signature rDS = (Gen, Sig, Ver, Rnd) is statistically unlinkable if for any algorithm $\mathcal{A}$, the statistical distance between output of $\mathsf{Exp}_{\mathcal{A}}^{unlink-b}(k)$ (defined above) for $b \in \{0,1\}$ is negligible, that is*

$$\sum |\mathsf{Pr}\{\mathsf{Exp}_{\mathcal{A}}^{unlink-1}(k) = 1\} - \mathsf{Pr}\{\mathsf{Exp}_{\mathcal{A}}^{unlink-0}(k) = 1\}| < \epsilon(k),$$

where the sum is over all random choices of Gen, $\mathcal{A}$ and Rnd.

Definition 5 (Computationally Unlinkable). *A randomizable signature rDS = (Gen, Sig, Ver, Rnd) is computationally unlinkable if for any probabilistic polynomial time algorithm $\mathcal{A}$, the probability between output of $\mathsf{Exp}_{\mathcal{A}}^{unlink-b}(k)$ (defined above) for $b \in \{0,1\}$ is negligible, that is*

$$\mathsf{Pr}\{\mathsf{Exp}_{\mathcal{A}}^{unlink-1}(k) = 1\} - \mathsf{Pr}\{\mathsf{Exp}_{\mathcal{A}}^{unlink-0}(k) = 1\} < \epsilon(k)$$

The above definitions of unlinkability can be further weakened by not allowing the adversary obtain the secret key, but granting access to signing oracle $\mathcal{O}_{sig}(sk, .)$ as in experiment $\mathsf{Exp}_{\mathcal{A}}^{w-unlink-b}(k)$ defined below. Then we get weak perfectly unlinkability, weak statistically unlinkability, weak computationally unlinkability analogously.

$\underline{\mathsf{Exp}_{\mathcal{A}}^{w-unlink-b}(k),\, b \in \{0,1\}}$: $(pk, sk) \overset{\$}{\leftarrow}$ Gen(1^k), $(m_0,\, \Upsilon_0,\, \Xi_0,\, m_1,\, \Upsilon_1,\, \Xi_1)$ $\overset{\$}{\leftarrow} \mathcal{A}^{\mathcal{O}_{sig}(sk,.)}(pk)$, If Ver$(pk,\, m_0,\, \langle\Upsilon_0,\, \Xi_0\rangle) = 0$ or Ver$(pk,\, m_1,\, \langle\Upsilon_1,\, \Xi_1\rangle) = 0$, return 0. $(\Upsilon',\, \Xi') \overset{\$}{\leftarrow}Rnd(m_b,\, \Upsilon_b,\, \Xi_b)$, $b' \leftarrow \mathcal{A}^{\mathcal{O}_{sig}(sk,.)}(pk,\, \Xi')$. return b'.

Definition 6 (Unlinkable Randomizable Signature). *A (perfectly, statistically, computationally) URS urDS=(Gen,Sig,Ver,Rnd) is a randomizable signature that is also (perfectly, statistically, computationally) unlinkable respectively.*

3.2 Σ-Protocol Friendly Randomizable and Indirectly Signable Signature

Definition 7 (Σ-protocol Friendly Randomizable Signature). *A randomizable signature rDS=(Gen, Sig, Ver, Rnd) is Σ -protocol friendly if there exits a Σ -protocol $\mathcal{P}$ for relation $\mathcal{R} = \{(\Xi, \langle\Upsilon, m\rangle) | Ver(pk, m, \langle\Upsilon, \Xi\rangle) = 1\}$, that is [14]*

- *$\mathcal{P}$ is of 3-move form, and if Prover and Verifier follow the protocol, Verifier always accepts.*
- *From any Ξ and any pair of accepting conversations with different initial message from Prover on input the same Ξ, one can efficiently compute (Υ, m) such that $(\Xi,\, \langle\Upsilon,\, m\rangle) \in \mathcal{R}$.*
- *There exists a polynomial time simulator M, which on input Ξ, and a random second message sent from Verifier, outputs an accepting conversation with the same probability distribution as between the honest Prover, Verifier on input Ξ.*

The following concept of *indirectly signable* is actually a restatement of signatures on committed message [6].

Definition 8 (Indirectly Signable). *A signature is indirectly signable if there exists a one way function f (as defined in Chapter 9.2.4, [15] or moretechnically as in*

Chapter 2.2, [16]) and an efficient algorithm Sig_f *that* $Sig(sk,m)=Sig_f(sk,f(m))$. *That is* $\mathsf{Pr}\{(pk,sk,f)\xleftarrow{\$}Gen(1^k), m\xleftarrow{\$}M, v\leftarrow f(m), \sigma\leftarrow Sig_f(sk,v) : Ver(pk, m,\sigma)=1\}=1$, *and for any probabilistic polynomial time algorithm* $\mathcal{A}$, $\mathsf{Pr}\{(pk,sk, f)\xleftarrow{\$}Gen(1^k), m\xleftarrow{\$}M, v\leftarrow f(m), m'\leftarrow\mathcal{A}(sk,v) : m'=m\}<\epsilon(k)$.

Actually signatures with above characteristics have been proposed and adopted explicitly or implicitly [7,6,8,9], see Table 1 (the scheme on the right is the

Table 1. Comparison of signatures and URS

ACJT [4]	CL02 [7]
Let $n=pq$ be an RSA modulus. $S_e=[2^{l_e}-2^{\mu_e}, 2^{l_e}+2^{\mu_e}]$, $S_m=[2^{l_m}-2^{\mu_m}, 2^{l_m}+2^{\mu_m}]$, $S_s=[2^{l_s}-2^{\mu_s}, 2^{l_s}+2^{\mu_s}]$, $\mu_e>l_m$.	
Gen. $a, c\xleftarrow{\$}QR_n^*$, $sk=(p,q)$, $pk=(n, a, c, S_e, S_m)$. *Sig.* If $\lvert m\rvert=l_m$, $e\xleftarrow{\$}S_e\cap\mathsf{P}rime$, $A\leftarrow(a^m c)^{\frac{1}{e}}$ mod n. *Ver.* Given m, (e, A), check if $\lvert m\rvert=l_m$, $A^e=a^m c$ mod n. *Rnd.* -	*Gen.* $a,b,c\xleftarrow{\$}QR_n^*$, $sk=(p,q)$, $pk=(n, a, b, c, S_e, S_m, S_s)$. *Sig.* If $\lvert m\rvert=l_m$, $e\xleftarrow{\$}S_e\cap\mathsf{P}rime$, $s\xleftarrow{\$}S_s$, $A\leftarrow(a^m b^s c)^{\frac{1}{e}}$ mod n. $\Upsilon=(e, s)$, $\Xi=(A)$ *Ver.* Given m, $(\Upsilon, \Xi)=(e, s, A)$, check if $\lvert m\rvert=l_m$, $\lvert s\rvert=l_s$, $A^e=a^m b^s c$ mod n. *Rnd.* Given m, $(\Upsilon,\Xi)=(e,s,A)$, choose random r with length $l_r=l_s-l_e-1$, $\Upsilon'=(e, s+re)$, $\Xi'=(Ab^r)$.
CL04 [6]	CL04+.
Let $\mathbb{G}_1=\langle g\rangle$, $\mathbb{G}_2=\langle\tilde{g}\rangle$ be p order cyclic groups that there exists a bilinear map $e:\mathbb{G}_1\times\mathbb{G}_2\rightarrow\mathbb{G}_3$.	
Gen. $x,y\xleftarrow{\$}Z_p^*$, $sk=(x,y)$, $X=\tilde{g}^x$, $Y=\tilde{g}^y$, $pk=(p, g, \tilde{g}, \mathbb{G}_1, \mathbb{G}_2, e, X, Y)$.	
Sig. $d\xleftarrow{\$}\mathbb{G}_1$, $\Upsilon=$NULL, $\Xi=(d, d^y, d^{x+mxy})$. *Ver.* Given m, $(\Upsilon,\Xi)=(a, b, c)$, check if $e(a, Y)=e(b, \tilde{g})$, $e(a, X)e(b, X)^m=e(c, \tilde{g})$. *Rnd.* Given m, $(\Upsilon,\Xi)=(a, b, c)$, $r\xleftarrow{\$}Z_p^*$, $\Upsilon'=$NULL, $\Xi'=(a', b', c')=(a^r, b^r, c^r)$.	*Sig.* $d\xleftarrow{\$}\mathbb{G}_1$, $s\xleftarrow{\$}Z_p^*$, $\Upsilon=(s)$, $\Xi=(d^s, d^{sy}, d^{x+mxy})$. *Ver.* Given m, $(\Upsilon,\Xi)=(s,a,b,c)$, check if $e(a, Y)=e(b, \tilde{g})$, $e(a, X)e(b, X)^m=e(c, \tilde{g})^s$. *Rnd.* Given m, $(\Upsilon,\Xi)=(s, a, b, c)$, r_1, $r_2\xleftarrow{\$}Z_p^*$, $\Upsilon'=(s')=(r_2 s)$, $\Xi'=(a', b', c')=(a^{r_1 r_2}, b^{r_1 r_2}, c^{r_1})$.
BBS04 [8]	BBS04+
Gen. $x\xleftarrow{\$}Z_p^*$, $w=\tilde{g}^x$, $h_1\xleftarrow{\$}\mathbb{G}_1$. $sk=(x)$, $pk=(p, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, h_1, e, w)$.	
Sig. $s\xleftarrow{\$}Z_p^*$, $A\leftarrow(h_1^m g)^{\frac{1}{x+s}}$. *Ver.* Given m, (s, A), check if $e(A, w\tilde{g}^s)=e(h_1^m g, \tilde{g})$. *Rnd.* -	*Sig.* $s,t\xleftarrow{\$}Z_p^*$, $A\leftarrow(h_1^m g)^{\frac{t}{x+s}}$, $\Upsilon=(s, t)$, $\Xi=(A)$. *Ver.* Given m, $(\Upsilon,\Xi)=(s, t, A)$, check if $e(A, w\tilde{g}^s)=e(h_1^m g, \tilde{g}^t)$. *Rnd.* Given m, $(\Upsilon,\Xi)=(s, t, A)$, $r\xleftarrow{\$}Z_p^*$, $\Upsilon'=(s, rt)$, $\Xi'=(A^r)$.

corresponding URS signature with indirect signability and Σ-protocol friendliness with respect to the scheme on the left).

To illustrate the unlinkable randomness, take Scheme A in [6] as an example (shown in Table 1). If we set Υ =NULL, $\Xi = (a, b, c)$, it is not even computationally unlinkable, because anyone can check if (m_1, a', b', c') or (m_0, a', b', c') is a valid signature. That is why group signatures adopting the above signature only result in selfless anonymity (a weaker anonymity where the adversary should not know the message m)[9].

If we set $\Upsilon = (a)$, $\Xi = (b, c)$, then it is still not even computationally unlinkable, but is weak computationally unlinkable assuming DDH is hard over group $\mathbb{G}_1$.

If we further set $\Upsilon = (a, b)$, $\Xi = (c)$, then it is perfectly unlinkable. So it is rather easy to come up with an unlinkable randomizable signature, just reveal the randomized signature as less as possible. But revealing too little of the randomized signature may lose Σ-protocol friendliness.

3.3 Some New Unlinkable Randomizable Signatures

NSN04*. As we have mentioned, the ACJT scheme [4] has an alternative construction utilizing URS CL02. As for the scheme in [11], no similar alternative has been proposed. In this section, we propose a new URS NSN04*, which can be adopted to build a new efficient group signature.

[11]	**NSN04*.**
Let $\mathbb{G}$ be a p order additive cyclic group, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}'$ a bilinear map on $\mathbb{G} = \langle P \rangle$.	
Gen. $\gamma \xleftarrow{\$} Z_p^*$, $P_{pub} = \gamma P$, $P_0 \xleftarrow{\$} \mathbb{G}$, $sk = (\gamma)$, $pk = (p, \mathbb{G}, \mathbb{G}', P, P_0, P_{pub}, e)$.	
Sig. $a \xleftarrow{\$} Z_p^*$, $A = \frac{1}{\gamma+a}[mP + P_0]$. *Ver.* Given m, (a, A), check if $e(A, P_{pub} + aP) = e(mP + P_0, P)$. *Rnd.* -	*Sig.* $(a, b, c) \xleftarrow{\$} Z_p^{*3}$, $A = \frac{1}{\gamma+a}[mP + (b + \gamma c)P_{pub} + P_0]$, $\Upsilon = (a, b, c)$, $\Xi = (A)$. *Ver.* Given m, $(\Upsilon, \Xi) = (a, b, c, A)$, check if $e(A, P_{pub} + aP) = e(mP + bP_{pub} + P_0, P)e(cP_{pub}, P_{pub})$. *Rnd.* Given m, $(\Upsilon, \Xi) = (a, b, c, A)$, $r \xleftarrow{\$} Z_p^*$, $\Upsilon' = (a', b', c') = (a, b + ra, c + r)$, $\Xi' = (A') = (A + rP_{pub})$.

Lemma 1. *NSN04* is wUF-ACMA if q-SDH problem in* $\mathbb{G}$ *is hard, where* q *is polynomial in* $|p|$. *See the full paper for the proof.*

NSN04* is indirectly signable if we define $f(m) = mP$ assuming Computational Diffie-Hellman problem on $\mathbb{G}$ is hard. Obviously, NSN04* is perfectly unlinkable because each randomized Ξ' only consists of one element that is generated independently and randomly each time.

NSN04* is Σ-protocol friendly, because there exists an efficient Σ-protocol for the relation $\{(m, a, b, c) | e(A, P_{pub})e(A, P)^a = e(P, P)^m e(P_{pub}, P)^b e(P_0, P)e(P_{pub}, P_{pub})^c\}$.

Wat05+. The recently proposed signature in [17], which is provable secure under CBDH assumption (Computational Bilinear Diffie-Hellman assumption) without random oracle, is also an URS if only we change a bit on it, see the following restatement with an extra algorithm *Rnd*.

	Wat05+
	Let $\mathbb{G}$, $\mathbb{G}'$ be two p order cyclic groups, and there exists a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}'$. $\mathbb{G} = \langle g \rangle$.
Gen.	Set secret key $sk = (x)$, $pk = (e, g_1, g_2, u, u', u_i, i = 0, .., l)$, where g_1, g_2, u, u', u_i are all elements from $\mathbb{G}$, $g_1 = g^x$, l is the maximum binary length of a message to be signed.
Sig.	Given a message m with length at most l, the signature (Υ, Ξ) is $\Upsilon = (s)$, $\Xi = (a, b) = (g^r, g_2^x(u' \prod_{i=1}^{l} u_i^{m_i})^r)u^s$, where $s \xleftarrow{\$} Z_p$. Note that (a, bu^{-s}) is a signature of m according to the scheme in [17].
Ver.	Given a message m and its signature $(\Upsilon, \Xi) = (s, a, b)$, it is a valid signature on m if $e(b, g) = e(u', a)e(g_2, g_1) \prod_{i=1}^{l} e(u_i, a)^{m_i} e(u, g)^s$.
Rnd.	On input pk, message m, and a signature (Υ, Ξ), where $\Upsilon = (s)$, $\Xi = (a, b)$, choose $(r_1, r_2) \xleftarrow{\$} Z_p \times Z_p$, set $\Upsilon' = (s') = (s + r_1)$, $\Xi' = (a', b') = (ag^{r_2}, b(u' \prod_{i=1}^{l} u_i^{m_i})^{r_2} u^{r_1})$. The new randomized signature on m is (Υ', Ξ').

Wat05+ is wUF-ACMA. The proof is easy, omitted here.

Wat05+ is Σ-protocol friendly, because there exits efficient Σ-protocol for the relation $\{(m_1, ..., m_l, s)|\ e(b, g) = e(u', a)\ e(g_2, g_1) \prod_{i=1}^{l} e(u_i, a)^{m_i} e(u, g)^s\}$.

Wat05+ is indirectly signable if we define $f(m) = \prod_{i=1}^{l} u_i^{m_i}$, it is one way if $l = O(k)$, where k is the security parameter. That is because the probability of $f(m) = f(m')$ for $m \neq m'$ is about $1/p$, i.e., the solution to $f(m) = c$ for a random $c \in \mathbb{G}$ is unique non-negligibly. To obtain the unique solution, 2^l tests must be carried out.

Wat05+ signature is perfectly unlinkable, because a' and b' are obtained from independent random variables.

Note that the original scheme Wat05 [17] is already utilized in the compact group signature [18]. But Wat05+ has not been adopted anywhere.

ZL06+. ZL06+ is a new URS similar to the standard signature proposed in [19].

	ZL06+
	Let $\mathbb{G}_1$ be a p order cyclic group that exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$. $\mathbb{G}_1 = \langle g \rangle$, $\mathbb{G}_2 = \langle \tilde{g} \rangle$.
Gen.	Select $(x, y) \xleftarrow{\$} Z_p^* \times Z_p^*$, set $X = g^x$, $Y = g^y$, $\widetilde{X} = \tilde{g}^x$, $\widetilde{Y} = \tilde{g}^y$. The secret key is $sk = (x, y)$, public key is $pk = (X, Y, \widetilde{X}, \widetilde{Y}, g, \tilde{g}, e, p)$.
Sig.	Given a message $m \in Z_p$, its signature is (Υ, Ξ), where $\Upsilon = (s)$, $\Xi = (a, b) = (g^r, g^{r(x+my)+sx+xy})$, $(r, s) \xleftarrow{\$} Z_p^* \times Z_p$.
Ver.	Given a signature $(\Upsilon, \Xi) = (s, a, b)$ of m, check if $e(b, \tilde{g}) = e(a, \widetilde{X}\widetilde{Y}^m)e(X, \widetilde{Y})e(X, \tilde{g})^s$. If the equation holds, then accept (Υ, Ξ) as a valid signature of m, otherwise reject it as invalid.
Rnd.	On input pk, message m, and a signature $(\Upsilon, \Xi) = (s, a, b)$, choose random r_1, $r_2 \in Z_p \times Z_p$, output (Υ', Ξ') where $\Upsilon' = (s') = (s + r_1)$, $\Xi' = (a', b') = (ag^{r_2}, b(XY^m)^{r_2} X^{r_1})$.

ZL06+ is wUF-ACMA secure under the assumption proposed in [19]. The proof is easy, omitted here.

ZL06+ is Σ-protocol friendly, because there exits efficient Σ-protocol for the relation $\{(m, s)|\ e(b, \tilde{g}) = e(a, \widetilde{X})\ e(a, \widetilde{Y})^m\ e(X, \widetilde{Y})e(X,\ \tilde{g})^s\}$.

ZL06+ is indirectly signable if define $f(m) = g^m$ assuming Computational Diffie-Hellman problem on $\mathbb{G}_1$ is hard.

ZL06+ signature is perfectly unlinkable, because a' and b' are obtained from independent random variables.

4 Group Signature from URS

Definition 9 (Group Signature [20]). *A group signature is a signature scheme composed of the following algorithms between GM (including IA, issuing authority, and OA, opening authority), group members and verifiers.*

- ***Setup:*** *an algorithm run by GM (IA and OA) to generate group public key gpk and group secret key gsk.*
- ***Join:*** *a probabilistic interactive protocol between GM (IA) and a group member candidate. If the protocol ends successfully, the candidate becomes a new group member with a group signing key gsk_i including member secret key msk_i and member certificate $cert_i$; and GM (IA) adds an entry for i (denoted as reg_i) in its registration table reg storing the protocol transcript, e.g. $cert_i$. Sometimes the procedure is also separated into **Join** and **Iss**, where Join emphasize the part run by group members as well as Iss denotes the part run by IA.*
- ***GSig:*** *a probabilistic algorithm run by a group member, on input a message m and a group signing key $gsk_i = (msk_i, cert_i)$, returns a group signature σ.*
- ***GVer:*** *a deterministic algorithm which, on input a message-signature pair (m, σ) and GM's public key gpk, returns 1 or 0 indicating the group signature is valid or invalid respectively.*
- ***Open:*** *a deterministic algorithm which, on input a message-signature pair (m, σ), secret key gsk of GM (OA), and the registration table reg, returns identity of the group member who signed the signature, and a proof π.*
- ***Judge:*** *a deterministic algorithm with output of Open as input, returns 1 or 0, i.e., the output of Open is valid or invalid.*

4.1 Generic Construction of GS

Select an URS $DS= (K_s, Sig, Ver, Rnd)$ which is indirectly signable with a one way function f, a probabilistic public encryption $PE= (K_e,\ Enc, Dec)$.

Define the following relations:

ρ: $(x, w) \in \rho$ iff $x = f(w)$.

ρ_1: $(\langle pk_e,\ pk_s,\ C,\ \Xi\rangle,\ \langle w,\ \Upsilon,\ r\rangle) \in \rho_1$ iff $Ver(pk_s,\ w,\ (\Upsilon,\ \Xi))$=1 and C=$Enc\ (pk_e,\ f(w),\ r)$ and $(pk_s, \cdot) \leftarrow K_s$, $(pk_e, \cdot) \leftarrow K_e$.

ρ_2: $(\langle pk_e, C, m\rangle, \langle w\rangle) \in \rho_2$ iff $Dec(pk_e, w, C) = m$ and $(pk_e, .) \leftarrow K_e$.

Assume (P, V), (P_1, V_1) and (P_2, V_2) are non-interactive proofs for relation ρ, ρ_1 and ρ_2, which have access to common reference string R, R_1 and R_2 respectively. Let SIM, SIM_1, SIM_2 be their corresponding simulation algorithm. The detailed definition of non-interactive proof is referred to [20].

(P, V) is also defined to be with an online extractor (in the random oracle model), i.e., it has the following features (let k be the security parameter) [21]:

① Completeness: For any random oracle H, any $(x,\ w) \in \rho$, and any $\pi \leftarrow P^H(x,\ w,\ R)$, it satisfies $\mathsf{Pr}\ \{V^H(x,\ \pi,\ R) = 1\} \geq 1 - \epsilon_1(k)$, where $\epsilon_1(k)$ is a negligible function.

② Online Extractor: There exists a probabilistic polynomial time algorithm K, the online extractor, such that the following holds for any algorithm A. Let H be a random oracle, $Q_H(A)$ be the answer sequence of H to queries from A. Let $w \leftarrow K(x,\ \pi,\ Q_H(A))$, then as a function of k, $\mathsf{Pr}\{(x,\ w) \notin \rho,\ V^H(x,\ \pi,\ R) = 1\} < \epsilon_2(k)$, where $\epsilon_2(k)$ is a negligible function.

GS is constructed as follows, see Table 2 for the details.

Setup. Select an instance of DS and PE, let secret key of DS be the secret key of IA, secret key of PE be the secret key of OA.

Table 2. Algorithms Setup, GSig, GVer, Open, Judge of GS

Algorithm **Setup**(1^k): $R \overset{\$}{\leftarrow} \{0,\ 1\}^{P(k)}$, $R_1 \overset{\$}{\leftarrow} \{0,\ 1\}^{P_1(k)}$, $R_2 \overset{\$}{\leftarrow} \{0,\ 1\}^{P_2(k)}$, $(pk_s,\ sk_s) \leftarrow K_s(1^k)$, $(pk_e,\ sk_e) \leftarrow K_e(1^k)$, $gpk = (R,\ R_1,\ R_2,\ pk_e,\ pk_s)$, $ok = (sk_e)$, $ik = (sk_s)$. return (gpk, ok, ik).	Algorithm **Join**: User $i \xrightarrow{pk_i, \pi}$ IA: User selects sk_i, $pk_i = f(sk_i)$, $\pi = P(pk_i, sk_i, R)$ User $i \xleftarrow{cert_i}$ IA: IA checks if $V(pk_i,\ \pi,\ R) = 1$, calculates $cert_i = Sig_f(sk_s,\ pk_i)$, sets $reg_i = pk_i$. User i: sets $gsk_i = (pk_i,\ sk_i,\ cert_i)$.
Algorithm **GSig**(gpk, gsk_i, m): Parse $cert_i$ as $(\Upsilon,\ \Xi)$, Parse gpk as $(R,\ R_1,\ R_2,\ pk_e,\ pk_s)$, $(\Upsilon',\ \Xi') = Rnd(gpk,\ sk_i,\ \Upsilon,\ \Xi)$; $C \leftarrow Enc(pk_e,\ pk_i,\ r_i)$, r_i random; $\pi_1 = P_1(\langle pk_e,\ pk_s,\ m,\ C,\ \Xi' \rangle,\ \langle sk_i,\ \Upsilon',\ r_i \rangle,\ R_1)$. $\sigma = (C,\ \Xi',\ \pi_1)$. return σ.	Algorithm **GVer**(gpk, m, σ): Parse σ as $(C,\ \Xi',\ \pi_1)$, Parse gpk as $gpk = (R,\ R_1,\ R_2,\ pk_e,\ pk_s)$, Return $V_1(\langle pk_e,\ pk_s,\ C,\ \Xi' \rangle,\ \pi_1,\ R_1)$. (Note that π_1 here denotes the signature on m transformed from the non-interactive proof.)
Algorithm **Open**$(gpk, ok, reg, m, \sigma)$: Parse gpk as $gpk = (R,\ R_1,\ R_2,\ pk_e,\ pk_s)$, Parse σ as $(C,\ \Xi',\ \pi_1)$, If GVer$(gpk,\ m,\ \sigma) = 0$, return $\perp$. $M \leftarrow Dec(sk_e,\ C)$, If $M = reg_i$, $\exists i$, $id \leftarrow i$, else $id \leftarrow 0$. $\pi_2 = P_2(\langle pk_e,\ C,\ M \rangle,\ \langle sk_e \rangle,\ R_2)$, return $(id,\ \tau)$, where $\tau = (M,\ \pi_2)$.	Algorithm **Judge**$(gpk, reg, m, \sigma, i, M, \pi_2)$: Parse gpk as $gpk = (R,\ R_1,\ R_2,\ pk_e,\ pk_s)$, Parse σ as $(C,\ \Xi',\ \pi_1)$, If GVer$(gpk,\ m,\ \sigma) = 0$, return $\perp$. return $V_2(\langle pk_e,\ C,\ M \rangle,\ \pi_2,\ R_2)$.

Join. User i selects its member secret key sk_i in message space of DS, computes $pk_i = f(sk_i)$, generates π, a non-interactive zero-knowledge proof of knowledge of sk_i for relation ρ. IA checks the correctness of π and generates a DS signature on sk_i: $cert_i = Sig_f(sk_s, pk_i) = Sig(sk_s, sk_i)$, sets $reg_i = pk_i$. The group signing key of i is $gsk_i = (cert_i, sk_i)$.

GSig. On input (gpk, gsk_i, m), parse $cert_i$ into (Υ, Ξ), firstly derive a new certification $(\Upsilon', \Xi') = Rnd(gpk, sk_i, \Upsilon, \Xi)$; then encrypt pk_i with PE: $C = Enc(pk_e, pk_i, r_i)$ where r_i is random; then generate π_1, a non-interactive zero-knowledge of proof of knowledge of (sk_i, Υ', r_i) for relation ρ_1; in the end, transform π_1 into a signature on m using any method of transforming a non-interactive zero-knowledge proof into a signature [5,22,23,24], we also use π_1 to note the transformed signature for simplicity. The group signature on m is $\sigma = (C, \Xi', \pi_1)$.

GVer. On input (gpk, m, σ), parse σ as (C, Ξ', π_1), check the correctness of π_1, return 1 if it is correct, return 0 otherwise.

Open. On input $(gpk, ok, reg, m, \sigma)$, parse σ as (C, Ξ', π_1). OA firstly checks the validity of the group signature σ on m, if it is not valid, stops; otherwise decrypts C to get M, and generates π_2, a proof of knowledge of decryption key ok for relation ρ_2. If $M = pk_i$ for some pk_i in reg, return the corresponding index or identity and π_2, else returns zero and π_2.

Judge. Check the correctness of π_2, return 1 if it is correct, return 0 otherwise.

Comparison. The above generic construction can be seen as a particular case of the construction in [20]:

In [20], the group signature is $\sigma = (C, \pi_1) = (Enc(pk_e, \langle i, pk_i, \Upsilon, \Xi, s\rangle, r_i), \pi_1)$, where $s = S(sk_i, m)$ and π_1 is a proof of knowledge of $(pk_i, \Upsilon, \Gamma, s, r_i)$ satisfying $Ver(pk_s, \langle i, pk_i\rangle, (\Upsilon, \Xi)) = 1$, $C = Enc(pk_e, \langle i, pk_i, \Upsilon, \Xi, s\rangle, r_i)$, and $V(pk_i, m, s) = 1$. (S, V) is the signature generation and verification algorithms of an independent signature scheme.

However in this construction, the group signature is $\sigma = (C, \Xi', \pi_1) = (Enc(pk_e, pk_i, r_i), \Xi', \pi_1)$, where π_1 is a transformed signature of the proof of knowledge of (sk_i, Υ', r_i) satisfying $Ver(pk_s, sk_i, (\Upsilon', \Xi')) = 1$ and $C = Enc(pk_e, f(sk_i), r_i)$.

The construction is more efficient in that less items are encrypted in C and the relation between member secret key, member certificate and other items is much simplified, thus efficient proof of knowledge of encrypted context is obtained.

4.2 Security Proofs

The above generic group signature utilizing unlinkable randomizable signature can be proved secure according to the proof methods for the security results in [20] under a variant model (see the full paper).

Lemma 2. *The above GS is anonymous if DS is computationally unlinkable, PE is IND-CCA2,* (P_1, V_1) *is a simulation sound, computational zero-knowledge proof,* (P_2, V_2) *is a computational zero-knowledge proof.*

Lemma 3. *The above GS is traceable if* DS *is wUF-ACMA,* (P_1, V_1)*,* (P_2, V_2) *are sound proofs of knowledge and* (P, V) *is a proof of knowledge with online extractor (in random oracle model).*

Lemma 4. *The above GS is non-frameable if* $f(\cdot)$ *is one way function,* (P, V) *is a computational zero-knowledge proof,* (P_1, V_1) *and* (P_2, V_2) *are sound proofs of knowledge.*

Note that there is a gap between the generic construction GS and the realization of it by adopting the Σ-protocol friendly URS' we have described earlier (the reason we require Σ-protocol friendliness is from efficiency consideration), because Σ-protocols (after they are transformed into non-interactive forms [5]) are not guaranteed simulation sound. It can be fixed in proof by utilizing rewinding techniques [25,26] so that an adversary, even after it has been given simulated group signatures, can not generate a valid group signature unless the ciphertext therein is correctly constructed.

4.3 Improvement to a Group Signature

Review of KY05's Scheme

Setup. At first, select the following public parameters:

- two groups $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$ of order p(length is l_p bits), and there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.
- an RSA modulus n of l_n bits.
- three integer ranges S, S', S where $S' \subset S \subset Z_{\phi(n)}$, the upper bound of S'' is smaller than the lower bound of S.
- an RSA modulus N of l_N bits, choose $G \in QR_{N^2}$ so that $\langle G \rangle$ is also N-th residues, $\sharp\langle G \rangle = \phi(N)/4$.

Then IA selects ① $\gamma, \delta \overset{\$}{\leftarrow} Z_p$, set $w = g_2^{\gamma}$, $v = g_2^{\delta}$; ② $\alpha, \beta \overset{\$}{\leftarrow} Z_p$, $u \overset{\$}{\leftarrow} \mathbb{G}_1$, set $u' = u^{\alpha/\beta}$, $h = u^{\alpha}(u')^{\beta} = u^{2\alpha}$; ③$g, f_1, f_2, f_3 \overset{\$}{\leftarrow} QR_n$; ④ a collision resistant hash function HASH.

OA selects ① $a_1, a_2, a_3 \overset{\$}{\leftarrow} Z_{\lfloor N/4 \rfloor}$, set $H_1 = G^{a_1}$, $H_2 = G^{a_2}$, $H_3 = G^{a_3}$; ② a universal one-way hash function family UOHF, and a hash key hk.

Group public key $gpk = \{g_1, g_2, u, u', h, w, v, g, f_1, f_2, f_3, n, N, G, H_1, H_2, H_3, hk, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathsf{UOHF}\}$. Group secret key $gsk = \{\gamma, \delta, a_1, a_2, a_3\}$.

Join. A user selects $x = x_1 x_2$, $x_1 \overset{\$}{\leftarrow} S''$, sends x to IA; IA checks whether $x \in S'$, if that is the case, selects $r \overset{\$}{\leftarrow} Z_p^*$, $s \overset{\$}{\leftarrow} Z_p^*$, calculates $\sigma \leftarrow g_1^{\frac{s}{\gamma+x+\delta r}}$, sends (r, s, σ) to the user; the user checks if $e(\sigma, w g_2^x v^r) = e(g_1, g_2)^s$, if so, sets $cert = (x, r, s, \sigma)$, $msk = (x_1, x_2)$.

GSig. If a user with member certificate (x, σ, r) and member secret key (x_1, x_2) wants to generate a group signature on m, he firstly computes T_1, T_2, T_3, T_4, T_5, C_0, C_1, C_2 as follows.

$T_1 = u^z$, $z \xleftarrow{\$} Z_p$ in $\mathbb{G}_1$; $T_2 = (u')^{z'}$, $z' \xleftarrow{\$} Z_p$ in $\mathbb{G}_1$; $T_3 = h^{z+z'}\sigma$ in $\mathbb{G}_1$; $T_4 = g^y f_1^{x_1}$, $y \xleftarrow{\$} S(1, 2^{l_n-2})$ in QR_n; $T_5 = g^{y'} f_2^{x_2} f_3^t$, $y' \xleftarrow{\$} S(1, 2^{l_n-2})$ in QR_n; $C_0 = G^t$, $t \xleftarrow{\$} S(1, 2^{l_N-2})$ in $Z_{N^2}^*$; $C_1 = H_1^t(1+N)^x$ in $Z_{N^2}^*$; $C_2 = \| (H_2 H_3^{\mathcal{H}(hk, C_0, C_1)})^t \|$ in $Z_{N^2}^*$.

Then he generates a signature of knowledge by applying the Fiat-Shamir heuristic [5] on a proof of knowledge of the fourteen witnesses θ_z, $\theta_{z'}$, θ_x, θ_{xz}, $\theta_{xz'}$, θ_r, $\theta_{rz'}$, θ_{x_1}, θ_{x_2}, θ_y, $\theta_{y'}$, θ_{yx_2}, θ_t that satisfy the following relations:

$$\boxed{\begin{array}{llll}
T_1 = u^{\theta_z}, & T_2 = (u')^{\theta_{z'}}, & 1 = T_1^{-\theta_x} u^{\theta_{xz}}, & 1 = T_2^{-\theta_x}(u')^{\theta_{xz'}}, \\
1 = T_1^{-\theta_r} u^{\theta_{rz}}, & 1 = T_2^{-\theta_r}(u')^{\theta_{rz'}}, & T_4 = g^{\theta_y} f_1^{\theta_{x_1}}, & 1 = T_4^{-\theta_{x_2}} g^{\theta_{yx_2}} f_1^{\theta_x}, \\
T_5 = g^{\theta_{y'}} f_2^{\theta_{x_2}} f_3^{\theta_t}, & \theta_x \in S', & \theta_{x'} \in S'', & C_0 = G^{\theta_t}, \\
\multicolumn{4}{l}{C_1 = H_1^{\theta_t}(1+N)^{\theta_x},\ C_2^2 = (H_2 H_3^{\mathcal{H}(hk, C_0, C_1)})^{2\theta_t},} \\
\multicolumn{4}{l}{e(g_1, g_2)/e(T_3, w) = e(T_3, v)^{\theta_r} e(T_3, g_2)^{\theta_x} e(h, g_2)^{-\theta_{xz}-\theta_{xz'}} e(h, v)^{-\theta_{rz}-\theta_{rz'}} e(h, w)^{-\theta_z-\theta_{z'}}}
\end{array}}$$

The realization of the above signature of knowledge is quite standard, so we omit it here. The output is $(T_1, T_2, T_3, T_4, T_5, C_0, C_1, C_2, c, s_z, s_{z'}, s_{xz}, s_{xz'}, s_r, s_{rz}, s_{rz'}, s_x, s_{x_1}, s_{x_2}, s_y, s_{y'}, s_{yx_2}, s_t)$.

GVer. The verification is achieved by checking the above proof of knowledge, omitted here.

Open. Firstly the group signature is verified as well as the relation $C_2^2 = C_0^{2(a_2+a_3\mathcal{H}(hk, C_0, C-1))}$ is checked. If all the tests pass, OA computes $x = (C_1 C_0^{-a_1} - 1)/N$, then checks if there exists a matching member certificate in the database maintained by IA.

Group Signature KY05+
Replacing the member certificate signature with the following BB04+ signature, the scheme in [12] can be improved.

	BB04+
	Let $\mathbb{G}_1$, $\mathbb{G}_2$ be two p order cyclic groups, and there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$. $\mathbb{G}_1 = \langle g \rangle$, $\mathbb{G}_2 = \langle \tilde{g} \rangle$.
Gen.	It chooses $x \xleftarrow{\$} Z_p^*$, $y \xleftarrow{\$} Z_p^*$, and sets $sk = (x, y)$, $pk = (p, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, X, Y, e)$, where $X = \tilde{g}^x$, $Y = \tilde{g}^y$.
Sig.	On input message m, secret key sk, and public key pk, choose $(s, t) \xleftarrow{\$} Z_p^{*2}$, compute $A = g^{\frac{t}{x+m+ys}}$, output the signature (Υ, Ξ) where $\Upsilon = (s, t)$, $\Xi = (A)$. Note that $(s, A^{\frac{1}{t}})$ is a valid [27] signature on m.
Ver.	On input pk, message m, and purported signature $(\Upsilon, \Xi) = (s, t, A)$, check that $e(A, XY^s\tilde{g}^m) = e(g^t, \tilde{g})$.
Rnd.	On input pk, message m, and a signature $(\Upsilon, \Xi) = (s, t, A)$, choose $r \xleftarrow{\$} Z_p^*$, output (Υ', Ξ') where $\Upsilon' = (s', t') = (s, rt)$, $\Xi' = (A') = (A^r)$.

It is easy to prove the wUF-ACMA of BB04+, similar to the original scheme [27]. Obviously, BB04+ is perfectly unlinkable because each randomized Ξ' only consists of one element that is generated independently and randomly each time, but it is not indirectly signable because m must be known to calculate a signature on it. BB04+ is Σ-protocol friendly, because there exists an efficient Σ-protocol for the relation $\{(m, s, t)|e(A, X)e(A, Y)^s e(A, \tilde{g})^m = e(g, \tilde{g})^t\}$.

Now we turn back to the group signature of KY05+. Public parameters and algorithms Setup, Join, Open are exactly as [12], except that key-setup for linear ElGamal encryption is eliminated.

GSig. If a user with member certificate (x, σ, r) and member secret key (x_1, x_2) wants to generate a group signature on m, he firstly computes $(\sigma', s', T_4, T_5, C_0, C_1, C_2)$ as described in the following table.

$\sigma' = \sigma^{r'}, s' = r's$	$r' \overset{\$}{\leftarrow} Z_p^*$ in $\mathbb{G}_1$
$T_4 = g^y f_1^{x_1}$	$y \overset{\$}{\leftarrow} S(1, 2^{l_n - 2})$ in QR_n
$T_5 = g^{y'} f_2^{x_2} f_3^t$	$y' \overset{\$}{\leftarrow} S(1, 2^{l_n - 2})$ in QR_n
$C_0 = G^t$	$t \overset{\$}{\leftarrow} S(1, 2^{l_N - 2})$ in $Z_{N^2}^*$
$C_1 = H_1^t (1 + N)^x$	in $Z_{N^2}^*$
$C_2 = \Vert (H_2 H_3^{\mathcal{H}(hk, C_0, C_1)})^t \Vert$	in $Z_{N^2}^*$

Then he generates a signature of knowledge by applying the Fiat-Shamir heuristic [5] on a proof of knowledge of the nine witnesses $(\theta_x, \theta_{x_1}, \theta_{x_2}, \theta_y, \theta_{y'}, \theta_{yx_2}, \theta_t, \theta_r, \theta_{s'})$ that satisfy the specified relations in the following table.

$g^{\theta_y} f_1^{\theta_{x_1}} = T_4,$	$g^{\theta_{y'}} f_2^{\theta_{x_2}} f_3^{\theta_t} = T_5,$	$T_4^{-\theta_{x_2}} g^{\theta_{yx_2}} f_1^{\theta_x} = 1,$
$e(\sigma', w g_2^{\theta_x} v^{\theta_r}) = e(g_1, g_2)^{\theta_{s'}},$	$G^{\theta_t} = C_0,$	$H_1^{\theta_t} (1 + N)^{\theta_x} = C_1,$
$(H_2 H_3^{\mathcal{H}(hk, C_0, C_1)})^{2\theta_t} = C_2^2,$	$\theta_x \in S',$	$\theta_{x'} \in S''.$

Note that the number of witnesses that need proving is fewer than that of [12]. Thus a group signature of KY05+ is $(\sigma', T_4, T_5, C_0, C_1, C_2, c, s_r, s_x, s_{x_1}, s_{x_2}, s_y, s_{y'}, s_{yx_2}, s_t, s_{s'})$, about $7|p| = 1190$ bits shorter than [12].

If we view $x = x_1 x_2$ as a one way function since factoring of x is hard, KY05+ is an application of the proposed generic construction on BB04+ except that a non-interactive zero-knowledge proof of knowledge with online extractor is not adopted in Join. The security of it follows from that of proposed generic construction and [12].

References

1. Camenisch, J., Stadler, M.: Efficient group signatures schemes for large groups. In: Advances in Cryptology - CRYPTO 1997. LNCS, vol. 1296, pp. 410–424. Springer, Heidelberg (1997)
2. Camenish, J., Michels, M.: A group signature scheme with improved efficiency. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 160–174. Springer, Heidelberg (1998)
3. Camenisch, J., Michels, M.: A group signature scheme based on an RSA-variant, in Technical Report RS-98-27, BRICS, University of Aarhus (1998)
4. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)

5. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
6. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
7. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
8. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 45–55. Springer, Heidelberg (2004)
9. Ateniese, G., Camenisch, J., de Medeiros, B., Hohenberger, S.: Practical group signatures without random oracles, Cryptology ePrint Archive, Report 2005/385
10. Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
11. Nguyen, L., Safavi-Naini, R.: Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 372–386. Springer, Heidelberg (2004)
12. Kiayias, A., Yung, M.: Group signatures with efficient concurrent join. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 198–214. Springer, Heidelberg (2005)
13. Delfs, H., Knebl, H.: Introduction to Cryptography: Principles and Applications, December 2001. Springer, Heidelberg (2001)
14. Damgård, I.: On Sigma-protocols 2005, `http://www.daimi.au.dk/~ivan/CPT.html`
15. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
16. Goldreich, O.: Foundations of Cryptography, vol. Basic Tools. Cambridge University Press, Cambridge (2001)
17. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
18. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006), `http://eprint.iacr.org/2005/381`
19. Zhou, S., Lin, D.: Shorter verifier-local revocation group signatures from bilinear maps. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 126–143. Springer, Heidelberg (2006), `http://eprint.iacr.org/2006/286`
20. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005), `http://www-cse.ucsd.edu/~mihir/papers/dgs.html`
21. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractor. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005)
22. Bellare, M., Goldwasser, S.: New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 194–211. Springer, Heidelberg (1990)

23. Cramer, R., Damgård, I.: Secure signature schemes based on interactive protocols. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 297–310. Springer, Heidelberg (1995)
24. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (2006)
25. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of Cryptology 13(3), 361–396 (2000)
26. Kiayias, A., Yung, M.: Secure scalable group signature with dynamic joins and separable authorities. International Journal of Security and Networks 1(1/2), 24–45 (2006), Also titled with Group Signatures: Provable Security, Efficient Constructions and Anonymity from Trapdoor-Holders, `http://eprint.iacr.org/2004/076`
27. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

An Improved Collision Attack on MD5 Algorithm

Shiwei Chen and Chenhui Jin[⋆]

Institute of Electronic Technology, the University of Information Engineering, Zhengzhou 450004, China
chenshiwei1012@sohu.com, jinchenhui@126.com

Abstract. The research on the attack algorithm for a MD5 collision is one of the focuses in cryptology nowadays. In this paper, by analyzing the properties of the nonlinear Boolean functions used in MD5 and the differences in term of XOR and subtraction modulo 2^{32}, we prove that some sufficient conditions presented by Jie Liang and Xuejia Lai are also necessary to guarantee the differential path and give a set of necessary and sufficient conditions to guarantee the output differences of the last two steps. Then we present an improved collision attack algorithm on MD5 by using the set of necessary and sufficient conditions. Finally, we analyze the computational complexity of our attack algorithm which is 0.718692 times of that of the previous collision attack algorithms.

1 Introduction

Many cryptographists have been focusing on analyzing the hash functions since Eurocrypt'04, when Wang et al. declared that they found a collision for MD5. MD5 is one of the important hash functions which are widely implemented in digital signature and many other cryptographic protocols. To guarantee the security of the applications in cryptography, the hash functions should be one-way and collision-resistant.

In the last three years, many cryptographists have tried to study the sufficient conditions for a MD5 collision proposed by Wang et al., and the message modification techniques to improve the efficiency of attack algorithms. Philip Hawkes et al. [1] mused on the way of producing collisions of MD5 by analyzing the collision pairs declared by Wang et al. in Eurocrypt'04. In Eurocrypt'05 Wang et al. [2] presented a differential path for a MD5 collision and gave the sufficient conditions to guarantee the differential path. However, according to a number of computer simulations, Jun Yajima and Takeshi Shimoyama [3] found that the sufficient conditions given by Wang et al. in [2] are not sufficient actually and modified some of the sufficient conditions. Yuto Nakano et al. [4] pointed out that there were 16 redundant conditions in [2] according to computer simulations and then explained why they are redundant. Yu Sasaki et al. [5] gave

[⋆] The Henan Province Science Fund for Distinguished Young Scholars In P.R. China (0312001800).

the way to construct sufficient conditions in searching collisions of MD5. Additionally, Jie Liang and Xuejia Lai [6] added several conditions to the set of sufficient conditions presented in [2] to guarantee that the left shift rotation and subtraction modulo 2^{32} can be commuted in a particular way which is used in the attack for MD5 collisions, and proposed a fast attack algorithm using small range searching technique. At the same time, Zhangyi Wang et al. [7] studied the same question as [6] but they proposed a different algorithm to yield a collision of MD5. Marc Stevens [8] optimized the set of sufficient conditions and presented a new algorithm for the first block. Since the probability that all the sufficient conditions for a MD5 collision are satisfied randomly is too small, Wang et al. [2] proposed message modification to make most of the sufficient conditions satisfied deterministically, which can improve the computational complexity of the attack algorithm. Then the authors give new techniques of message modification in [6] [7] [9] [10] [11] respectively which all can make more conditions satisfied deterministically. Yu Sasaki et al.[10] pointed that there are 14 sufficient conditions in the second round could be satisfied deterministically by advanced multi-message modification techniques. So in this paper we will use the multi-message modification techniques in [10] to make the sufficient conditions from 1-22 steps satisfied deterministically. Recently, Hongbo Yu et al. [12] presented multi-collision attack on the compression functions of MD4, which presented 4-collision using two different differential paths. The 4-collision means that there are 4 different messages producing the same hash value. However, for a certain differential path for MD5 collisions, there may be more than one set of sufficient conditions to hold it, which is the object we seek for in this paper.

In this paper, we will analyze the properties of the nonlinear Boolean functions in MD5, and then prove that the sufficient conditions from 24-61 steps but step 35 presented by Jie Liang and Xuejia Lai are also necessary to guarantee the differential path and give the necessary and sufficient conditions to guarantee the output differences of the last two chaining values. Since we obtain the necessary and sufficient conditions for some steps, we propose an attack algorithm for MD5 collisions. In our improved attack algorithm, we just test whether the output differences are what we desired and don't need to check whether the sufficient conditions holds, which will increase the probability that the algorithm continues to the next step and improve the efficiency of the attack algorithm. Finally, we will compare the computational complexity of our attack algorithm with that of the previous attack algorithms.

2 Description of MD5 Algorithm and Notations

MD5 is a hash function that generates a 128-bit hash value for a message with any length. It includes the integer addition modulo 2^{32}, the left shift rotation, and nonlinear Boolean functions. The initial values are given as follows:

$$A = 0x67452301, B = 0xefcdab89, C = 0x98badcfe, D = 0x10325476.$$

The compression function of MD5 has four rounds, and each round has 16 steps. Chaining values a, b, c and d are initialized as $a_0 = A, b_0 = B, c_0 =$

$C, d_0 = D$. One of the chaining values is updated in each step and computation is continued in sequence. In each step, the operation is as follows:

$$\begin{aligned} a &= b + ((a + f(b,c,d) + m + const) \lll k) \\ d &= a + ((d + f(a,b,c) + m + const) \lll k) \\ c &= d + ((c + f(d,a,b) + m + const) \lll k) \\ b &= c + ((b + f(c,d,a) + m + const) \lll k) \end{aligned}$$

where $'+'$ is addition modulo 2^{32}, *const* and k are step-dependent constants, m is a 32-bit message word and the 512-bit message block is divided into 16 32-bit message words. $'x \lll k'$ is the left shift rotation of x by k bits. f is a round-dependent function:

$$\begin{aligned} Round1 &: f = F(x,y,z) = (x \wedge y) \vee ((\neg x) \wedge z) \\ Round2 &: f = G(x,y,z) = (x \wedge z) \vee (y \wedge (\neg z)) \\ Round3 &: f = H(x,y,z) = x \oplus y \oplus z \\ Round4 &: f = I(x,y,z) = y \oplus (x \vee (\neg z)) \end{aligned}$$

Throughout this paper, for $x = \sum_{i=1}^{32} x_i 2^{i-1} \in Z/(2^{32})$, $x_i \in \{0,1\}$, we call x_i the i-th bit of x, and we will use following symbols:

$M_0(M_0^{'})$: the first block of the input message.
$M_1(M_1^{'})$: the second block of the input message.
a_i, b_i, c_i, d_i: the i-th values of a, b, c, d for (M_0, M_1) $(1 \le i \le 16)$.
$a_i^{'}, b_i^{'}, c_i^{'}, d_i^{'}$: the i-th values of a, b, c, d for$(M_0^{'}, M_1^{'})$ $(1 \le i \le 16)$.
$a_{i,j}, b_{i,j}, c_{i,j}, d_{i,j}$: the j-th bit of a_i, b_i, c_i, d_i ($1 \le i \le 16$, $1 \le j \le 32$).
$a_{i,j}^{'}, b_{i,j}^{'}, c_{i,j}^{'}, d_{i,j}^{'}$: the j-th bit of $a_i^{'}, b_i^{'}, c_i^{'}, d_i^{'}$ ($1 \le i \le 16$, $1 \le j \le 32$).
aa_1, bb_1, cc_1, dd_1: the initial values of the second iteration.
$H_i(H_i^{'})$: the hash value before the (i+1)-st iteration $(i \ge 0)$.
m_i: the i-th 32-bit message word $(0 \le i \le 15$).
$x \lll k$: the left shift rotation of x by k bits.
$\triangle X$, δX: $\triangle X = (X^{'} - X) mod 2^{32}$, $\delta X = X^{'} \oplus X (X \in \{a_i, b_i, c_i, d_i, \phi_i\})$.
$|A|$: the number of elements included in the set A.

3 Our Attack Algorithm on MD5

Wang et al. proposed a differential path to produce a two-block collision for MD5 in [2] and gave a set of sufficient conditions to guarantee the differential path. Then Jie Liang and Xuejia Lai [6] modified the set of sufficient conditions, which is relatively accurate so far and will be used in this paper. Then the authors gave different techniques of message modification in [6] [7] [9] [10] [11] respectively which all can make more conditions satisfied deterministically. In our attack algorithm, we use the message modification presented by Yu Sasaki et al.[10] to make all the conditions from 1-22 steps satisfied deterministically and other conditions in the remaining steps will be satisfied randomly. However, we find that the sufficient conditions are not necessary to guarantee the differential path from 63-64 steps. Since step 63 and step 64 are the last two in attack algorithms, the sufficient conditions make the probability that the attack algorithms continue to the last two steps reduce, which increases the complexity of the attack algorithms greatly. In this section, we first review the differential path proposed by Wang et al.. Then using the properties of the nonlinear Boolean functions in MD5 algorithm, we prove that the sufficient conditions from 24-61 steps except for step 35 are also necessary to guarantee the differential path, and give the necessary and sufficient conditions to guarantee the differential path from step 63 to step 64. Finally, using the set of necessary and sufficient conditions for a MD5 collision, we will present an attack algorithm on MD5 and compare the computational complexity of our attack algorithm with that of the previous algorithms in detail.

3.1 The Differential Path Proposed by Wang et al.

Finding a differential path which generates collisions with high probability is the first step of the collision attack on MD5. However, so far, few papers have been published about how to find a suitable differential path. So, in this paper we still use the differential path proposed by Wang et al. in [2], which is described as follows:

$$\triangle H_0 = 0 \overset{(M_0, M_0^{'})}{\longrightarrow} \triangle H_1 \overset{(M_1, M_1^{'})}{\longrightarrow} \triangle H = 0,$$

where $M_0^{'} - M_0 = (0,0,0,0,2^{31},0,0,0,0,0,0,2^{15},0,0,2^{31},0)$, $M_1^{'} - M_1 = (0,0,0,0,2^{31},0,0,0,0,0,0,-2^{15},0,0,2^{31},0)$ and $\triangle H_1 = (2^{31}, 2^{31}+2^{25}, 2^{31}+2^{25}, 2^{31}+2^{25}) = (\triangle a_{16}, \triangle d_{16}, \triangle c_{16}, \triangle b_{16})$. Moreover, we can refer to the table 3 and table 5 in [2] to get the output difference of each step.

3.2 The Set of Necessary and Sufficient Conditions to Guarantee Differential Path

Some Properties of the Nonlinear Boolean Functions used in MD5. There are four nonlinear Boolean functions used in MD5 algorithm. In this part, we only analyze the properties of the three nonlinear Boolean functions used in the last three rounds since we will derive the necessary and sufficient conditions from 24-64 steps.

Lemma 1. *Let $x, y, z, x^{'}, y^{'}, z^{'} \in Z/(2^{32})$, $\triangle x = 0$, $\triangle y = 2^{31}$, $\triangle z = 2^{31}$ and $G(x,y,z) = (x \wedge z) \vee (y \wedge (\neg z))$. Then $\triangle G(x,y,z) = 0$ if and only if $x_{32} \oplus y_{32} \oplus z_{32} = 0$.*

Lemma 2. *Let $x, y, z, x^{'}, y^{'}, z^{'} \in Z/(2^{32})$, $\triangle x = 0$, $\triangle y = 0$, $\triangle z = 2^{31}$ and $G(x,y,z) = (x \wedge z) \vee (y \wedge (\neg z))$. Then $\triangle G(x,y,z) = 2^{31}$ if and only if $y_{32} = x_{32} \oplus 1$.*

Lemma 3. *Let $x, y, z, x^{'}, y^{'}, z^{'} \in Z/(2^{32})$ and $H(x,y,z) = x \oplus y \oplus z$. Then*
(1) If $\triangle x = \triangle y = 0$, then $\triangle z = 2^{31}$ if and only if $\triangle H(x,y,z) = 2^{31}$.
(2) If $\triangle x = \triangle y = 2^{31}$, then $\triangle z = 2^{31}$ if and only if $\triangle H(x,y,z) = 2^{31}$.

Lemma 4. *Let* $x, y, z, x^{'}, y^{'}, z^{'} \in Z/(2^{32}), \triangle x = \triangle y = \triangle z = 2^{31}$ *and* $I(x, y, z) = y \oplus (x \vee (\neg z))$. *Then*
(1) $\triangle I(x, y, z) = 2^{31}$ *if and only if* $x_{32} = z_{32}$.
(2) $\triangle I(x, y, z) = 0$ *if and only if* $z_{32} = x_{32} \oplus 1$.

The set of the necessary and sufficient conditions. Using the properties of the nonlinear Boolean functions described above, we can prove that the sufficient conditions in table 4 of [6] are also necessary to guarantee the differential path from 24-61 steps but step 35.

Proposition 1. *The sufficient conditions presented by Jie Liang and Xuejia Lai are also necessary to guarantee the differential path from 24-61 steps except for step 35 in the first iteration.*

Proof. From the table 3 in [2], we get the output differences from 24-61 steps. In step 24, we know that

$$b_6 = c_6 + ((G(c_6, d_6, a_6) + b_5 + m_4 + 0xe7d3fbc8) \lll 20)$$

and $\triangle c_6 = 0$, $\triangle d_6 = \triangle a_6 = \triangle b_5 = 2^{31}$ and $\triangle m_4 = 2^{31}$. To achieve $\triangle b_6 = 0$, it is necessary and sufficient to have $\triangle G(c_6, d_6, a_6) = 0$. By lemma 1, we know that $c_{6,32} = d_{6,32} \oplus a_{6,32}$ is necessary and sufficient condition to hold $\triangle G(c_6, d_6, a_6) = 0$. Moreover, from the table 4 of [6] we know that $d_{6,32} = a_{6,32}$ holds to guarantee the output difference in step 23, so $c_{6,32} = 0$ is necessary and sufficient condition to hold $\triangle G(c_6, d_6, a_6) = 0$.

Henceforth, in the same way we can prove that the sufficient conditions presented by Jie Liang and Xuejia Lai in [6] are also necessary to guarantee the differential path from 24-61 steps but step 35 in the first iteration. Hence, we have proved the proposition 1.

Remark 1. In step 35 and step 62, we know that the two extra conditions in table 4 of [6] are necessary and sufficient to guarantee that the left shift rotation and subtraction modulo 2^{32} can be commuted. If the two operations can be commuted, the output difference in step 35 is held and the condition $a_{16,32} = c_{15,32}$ is necessary and sufficient to hold $\triangle I(a_{16}, b_{15}, c_{15}) = 2^{31}$ by lemma 4 and thereby guarantee the output difference in step 62. Hence, the sufficient conditions in [6] are also necessary to ensure that the two operations can be commuted and the output differences are held.

In the following part, we will derive the necessary and sufficient conditions to hold the last two output differences which are $\triangle c_{16} = 2^{31} + 2^{25}$ and $\triangle b_{16} = 2^{31} + 2^{25}$. Firstly, we give out two lemmas as follows.

Lemma 5. *Let* $x_i, y_i, \alpha_i, \beta_i \in \{0, 1\}$ *and* $h(x_i, y_i) = x_i \vee (y_i \oplus 1)$. *Then*

$$h(x_i \oplus \alpha_i, y_i \oplus \beta_i) \oplus h(x_i, y_i) = \begin{cases} y_i\alpha_i, & \text{if } \beta_i = 0; \\ y_i\alpha_i \oplus x_i \oplus \alpha_i \oplus 1, & \text{if } \beta_i = 1. \end{cases}$$

Lemma 6. *Let $X^{'}, X \in Z/(2^{32})$ and $\triangle X = (X^{'} - X) mod 2^{32}$. Then $\triangle X = 2^{31} + 2^{25}$ if and only if $X^{'}_{i} = X_i$ holds for each i with $1 \leq i \leq 25$, and one of the following conditions is satisfied:*

(1) $X^{'}_{32} = X_{32}$, and $X^{'}_{26+t} = 0, X_{26+t} = 1$ for every t with $0 \leq t \leq 5$;

(2) $X^{'}_{32} \oplus X_{32} = 1$, and there exists a q with $0 \leq q \leq 5$ such that $X^{'}_{26+q} = 1, X_{26+q} = 0$, and $X^{'}_{26+t} = 0, X_{26+t} = 1$ for each t with $0 \leq t < q$, $X^{'}_{26+t} = X_{26+t}$ for each t with $q < t \leq 5$.

Proof. (1) Let $X^{'} < 2^{25}$. Hence $\triangle X = 2^{31} + 2^{25}$ if and only if $X = (X^{'} - \triangle X) mod 2^{32} = [X^{'} - (2^{31} + 2^{25})] mod 2^{32} = 2^{31} - 2^{25} + X^{'}$ which is equivalent to that $X_{32} = 0$, $X_{26+t} = 1$ for each t with $0 \leq t \leq 5$, and $X^{'}_{i} = X_i$ holds for each i with $1 \leq i \leq 25$.

Now we suppose that $X^{'} \geq 2^{25}$, then there exists an i with $0 \leq i \leq 6$, such that $X^{'}_{26+i} = 1$. Put $q = min\{i : X^{'}_{26+i} = 1, 0 \leq i \leq 6\}$, then $X^{'}_{26+t} = 0$ for every t with $0 \leq t < q$.

Let $q = 6$. Then there exists an integer b with $0 \leq b < 2^{25}$, such that $X^{'} = 2^{31} + b$. Hence $\triangle X = 2^{31} + 2^{25}$ if and only if

$$X = (X^{'} - \triangle X) mod 2^{32} = [(2^{31} + b) - (2^{31} + 2^{25})] mod 2^{32} = (2^{32} - 2^{25}) + b$$

which is equivalent to that $X_{26+t} = 1$ for each t with $0 \leq t \leq 6$ and $X^{'}_{i} = X_i$ for every i with $1 \leq i \leq 25$.

Thus, $\triangle X = 2^{31} + 2^{25}$ is equivalent to that $X^{'}_{32} = X_{32}$, and $X^{'}_{26+t} = 0$, $X_{26+t} = 1$ for every t with $0 \leq t \leq 5$, and $X^{'}_{i} = X_i$ for every i with $1 \leq i \leq 25$.

(2) Let $0 \leq q < 6$. Then there exists an integer b with $0 \leq b < 2^{25}$ and an integer d with $0 \leq d < 2^{6-q}$, such that $X^{'} = 2^{26+q}d + 2^{25+q} + b$. Hence $X^{'}_{26+t} = 0$ for each t with $0 \leq t < q$. Hence $\triangle X = 2^{31} + 2^{25}$ if and only if

$$\begin{aligned} X &= (X^{'} - \triangle X) mod 2^{32} = [(2^{26+q}d + 2^{25+q} + b) - (2^{31} + 2^{25})] mod 2^{32} \\ &= [(2^{26+q}d - 2^{31}) + (2^{25+q} - 2^{25}) + b] mod 2^{32} \\ &= \begin{cases} (2^{26+q}d - 2^{31}) + (2^{25+q} - 2^{25}) + b, & \text{if } d \geq 2^{5-q}; \\ (2^{31} + 2^{26+q}d) + (2^{25+q} - 2^{25}) + b, & \text{if } d < 2^{5-q}. \end{cases} \end{aligned}$$

Since $d \geq 2^{5-q}$ if and only if $2^{26+q}d \geq 2^{31}$, it is equivalent to that $X_{32} = X^{'}_{32} \oplus 1$, $X_{26+t} = 1$ for each t with $0 \leq t < q$, $X_{26+q} = 0$, and $X^{'}_{26+t} = X_{26+t}$ for each t with $q < t \leq 5$,and $X^{'}_{i} = X_i$ for every i with $1 \leq i \leq 25$. Now we have proved lemma 6.

Remark 2. Since $X - X^{'} = 2^{31} - 2^{25}$ if and only if $X^{'} - X = 2^{32} - (2^{31} - 2^{25}) = 2^{31} + 2^{25}$. Hence, if we exchange $X^{'}$ with X in (1) and (2) of lemma 6, then we can obtain the necessary and sufficient conditions to hold $\triangle X = X^{'} - X = 2^{31} - 2^{25}$.

Using the results in lemma 5 and lemma 6, we can get the set of necessary and sufficient condition to hold $\triangle c_{16} = 2^{31} + 2^{25}$ and $\triangle b_{16} = 2^{31} + 2^{25}$ as a consequence of theorem 1 and theorem 2.

Theorem 1. *Let $c_{15}, b_{15}, a_{16}, d_{16}, c_{16}, b_{16} \in Z/(2^{32})$ be the chaining values in MD5 algorithm, $\triangle c_{15} = \triangle b_{15} = \triangle a_{16} = 2^{31}$, $\triangle d_{16} = 2^{31} + 2^{25}$ and $\triangle m_2 = 0$. Then $\triangle c_{16} = 2^{31} + 2^{25}$ if and only if one of the following conditions is satisfied:*

(1) $d_{16,32} = b_{15,32}$, for some q with $0 \le q \le 5$ such that $d_{16,26+q} = 0$, $d_{16,26+t} = 1$ for each t with $0 \le t < q$ and $b_{15,26+t} = 0$ for each t with $0 \le t \le q$.

(2) $d_{16,26+t} = 1$ for each t with $0 \le t \le 6$ and $b_{15,26+t} = 0$ for each t with $0 \le t \le 5$.

Proof. From the 63rd step of MD5 in the first iteration, we know that

$$c_{16} = d_{16} + [I(d_{16}, a_{16}, b_{15}) + c_{15} + m_2 + \mathit{0x2ad7d2bb}] \lll 15,$$

where $I(d_{16}, a_{16}, b_{15}) = a_{16} \oplus (d_{16} \vee (\neg b_{15}))$.

From $\triangle d_{16} = 2^{31} + 2^{25}$, $\triangle c_{15} = 2^{31}$ and $\triangle m_2 = 0$, we know that $\triangle c_{16} = 2^{31} + 2^{25}$ iff $\triangle I(d_{16}, a_{16}, b_{15}) = 2^{31}$, which is equivalent to $\delta I(d_{16}, a_{16}, b_{15}) = 2^{31}$. From $\triangle b_{15} = \triangle a_{16} = 2^{31}$, we know $\delta b_{15} = \delta a_{16} = 2^{31}$. From the proof of lemma 4, we know that

$$\begin{aligned}\delta I(d_{16}, a_{16}, b_{15}) &= (\delta d_{16} \& b_{15}) \oplus (\delta b_{15} \& d_{16}) \oplus (\delta d_{16} \& \delta b_{15}) \oplus \delta a_{16} \oplus \delta b_{15} \\ &= (\delta d_{16} \& b_{15}) \oplus ((d_{16} \oplus \delta d_{16}) \& 2^{31}).\end{aligned}$$

By lemma 6, we may divide $\triangle d_{16} = 2^{31} + 2^{25}$ into two cases:

Case 1: $\delta d_{16} mod 2^{25} = 0$, $d'_{16,32} \oplus d_{16,32} = 1$ and there exist a q with $0 \le q \le 5$ such that $d'_{16,26+q} = 1, d_{16,26+q} = 0$, and $d'_{16,26+t} = 0, d_{16,26+t} = 1$ for each t with $0 \le t < q$, $d'_{16,26+t} = d_{16,26+t}$ for each t with $q < t \le 5$. Then

$$\delta d_{16} = d'_{16} \oplus d_{16} = 2^{31} \oplus \bigoplus_{t=0}^{q} 2^{25+t}.$$

Hence

$$\begin{aligned}\delta I(d_{16}, a_{16}, b_{15}) &= [(2^{31} \oplus \bigoplus_{t=0}^{q} 2^{25+t}) \& b_{15}] \oplus \{[d_{16} \oplus (2^{31} \oplus \bigoplus_{t=0}^{q} 2^{25+t})] \& 2^{31}\} \\ &= [(d_{16} \oplus b_{15}) \& 2^{31}] \oplus [(\bigoplus_{t=0}^{q} 2^{25+t}) \& b_{15}] \oplus 2^{31}.\end{aligned}$$

So $\delta I(d_{16}, a_{16}, b_{15}) = 2^{31}$ iff $[(d_{16} \oplus b_{15}) \& 2^{31}] \oplus [(\bigoplus_{t=0}^{q} 2^{25+t}) \& b_{15}] = 0$, which is equivalent to $d_{16,32} = b_{15,32}$ and $b_{15,26+t} = 0$ for each t with $0 \le t \le q$. Hence $\delta I(d_{16}, a_{16}, b_{15}) = 2^{31}$ if and only if (1) holds in this case.

Case 2: $\delta d_{16} mod 2^{25} = 0$, $d'_{16,32} = d_{16,32}$ and $d'_{16,26+t} = 0, d_{16,26+t} = 1$ for each t with $0 \le t \le 5$. Then $\delta d_{16} = d'_{16} \oplus d_{16} = \bigoplus_{t=0}^{5} 2^{25+t}$. Hence

$$\begin{aligned}\delta I(d_{16}, a_{16}, b_{15}) &= [(\bigoplus_{t=0}^{5} 2^{25+t}) \& b_{15}] \oplus \{[d_{16} \oplus (\bigoplus_{t=0}^{5} 2^{25+t})] \& 2^{31}\} \\ &= (d_{16} \& 2^{31}) \oplus [(\bigoplus_{t=0}^{5} 2^{25+t}) \& b_{15}].\end{aligned}$$

So $\delta I(d_{16}, a_{16}, b_{15}) = 2^{31}$ iff $(d_{16}\&2^{31}) \oplus [(\bigoplus_{t=0}^{5} 2^{25+t})\&b_{15}] = 2^{31}$, which is equivalent to $d_{16,32} = 1$ and $b_{15,26+t} = 0$ for each t with $0 \leq t \leq 5$. Hence $\delta I(d_{16}, a_{16}, b_{15}) = 2^{31}$ iff (2) holds in this case. Hence theorem 1 holds.

Theorem 2. *Let $b_{15}, a_{16}, d_{16}, c_{16}, b_{16} \in Z/(2^{32})$ be the chaining values in MD5 algorithm, $\triangle b_{15} = \triangle a_{16} = 2^{31}$, $\triangle d_{16} = 2^{31} + 2^{25}$, $\triangle c_{16} = 2^{31} + 2^{25}$ and $\triangle m_9 = 0$. Then $\triangle b_{16} = 2^{31} + 2^{25}$ if and only if one of the following conditions is satisfied:*

(1) *$c_{16,32} = a_{16,32}$, for some q, p with $0 \leq q \leq p \leq 5$ such that $d_{16,26+t} = 1$ for each t with $0 \leq t < q$ and $d_{16,26+q} = 0$, and $c_{16,26+t} = 1$ for each t with $0 \leq t < p$ and $c_{16,26+p} = 0$, and*

$$a_{16,26+i} = \begin{cases} 1, & \textit{if } 0 \leq i \leq q; \\ 0, & \textit{if } q < i \leq p. \end{cases}$$

(2) *for some q with $0 \leq q \leq 5$ such that $d_{16,26+t} = 1$ for each t with $0 \leq t < q$ and $d_{16,26+q} = 0$, $c_{16,26+t} = 1$ for every t with $0 \leq t \leq 6$ and*

$$a_{16,26+i} = \begin{cases} 1, & \textit{if } 0 \leq i \leq q; \\ 0, & \textit{if } q < i \leq 5. \end{cases}$$

(3) *$c_{16,32} = a_{16,32} \oplus 1$, $d_{16,26+t} = 1$ for each t with $0 \leq t \leq 5$, $c_{16,26+t} = 1$ for each t with $0 \leq t < 5$ and $c_{16,31} = 0$, and $a_{16,26+i} = 1$ for each i with $0 \leq i \leq 5$;*

(4) *$d_{16,26+t} = 1$ for each t with $0 \leq t \leq 5$, $c_{16,26+t} = 1$ for each t with $0 \leq t \leq 5$ and $c_{16,32} = 0$, and $a_{16,26+i} = 1$ for each i with $0 \leq i \leq 5$.*

Proof. From the 64^{th} step of MD5 in the first iteration, we know that

$$b_{16} = c_{16} + [I(c_{16}, d_{16}, a_{16}) + b_{15} + m_9 + \mathit{0xeb86d39}] \lll 21,$$

where $I(c_{16}, d_{16}, a_{16}) = d_{16} \oplus (c_{16} \vee (\neg a_{16}))$.

From $\triangle c_{16} = 2^{31}+2^{25}$, $\triangle b_{15} = 2^{31}$ and $\triangle m_9 = 0$, we know that $\triangle b_{16} = 2^{31} + 2^{25}$ if and only if $\triangle I(c_{16}, d_{16}, a_{16}) = 2^{31}$ which is equivalent to $\delta I(c_{16}, d_{16}, a_{16}) = 2^{31}$. From $\triangle a_{16} = 2^{31}$, we know $\delta a_{16} = 2^{31}$. From the proof of lemma 4, we know that $\delta I(c_{16}, d_{16}, a_{16}) = (\delta c_{16}\&a_{16}) \oplus (\delta a_{16}\&c_{16}) \oplus (\delta c_{16}\&\delta a_{16}) \oplus \delta d_{16} \oplus \delta a_{16}$.

By lemma 6, we may divide $\triangle c_{16} = 2^{31} + 2^{25}$ and $\triangle d_{16} = 2^{31} + 2^{25}$ into four cases:

Case 1: $\delta d_{16} mod 2^{25} = 0$, $d'_{16,32} \oplus d_{16,32} = 1$ and there exists a q with $0 \leq q \leq 5$ such that $d'_{16,26+q} = 1$, $d_{16,26+q} = 0$, and $d'_{16,26+t} = 0, d_{16,26+t} = 1$ for each t with $0 \leq t < q$, $d'_{16,26+t} = d_{16,26+t}$ for each t with $q < t \leq 5$; $\delta c_{16} mod 2^{25} = 0$, $c'_{16,32} \oplus c_{16,32} = 1$ and there exists a p with $0 \leq p \leq 5$ such that $c'_{16,26+p} = 1$, $c_{16,26+p} = 0$, and $c'_{16,26+t} = 0, c_{16,26+t} = 1$ for each t with $0 \leq t < p$, $c'_{16,26+t} = c_{16,26+t}$ for each t with $p < t \leq 5$. Then

$$\delta d_{16} = d'_{16} \oplus d_{16} = 2^{31} \oplus \bigoplus_{t=0}^{q} 2^{25+t} \quad \text{and} \quad \delta c_{16} = c'_{16} \oplus c_{16} = 2^{31} \oplus \bigoplus_{t=0}^{p} 2^{25+t}.$$

Hence

$$\begin{aligned}\delta I(c_{16}, d_{16}, a_{16}) &= (\delta c_{16} \& a_{16}) \oplus (\delta a_{16} \& c_{16}) \oplus (\delta c_{16} \& \delta a_{16}) \oplus \delta d_{16} \oplus \delta a_{16} \\ &= [(\bigoplus_{t=0}^{p} 2^{25+t}) \& a_{16}] \oplus [2^{31} \& (a_{16} \oplus c_{16})] \oplus (\bigoplus_{t=0}^{q} 2^{25+t}) \oplus 2^{31}\end{aligned}$$

So $\delta I(c_{16}, d_{16}, a_{16}) = 2^{31}$ iff $a_{16,32} = c_{16,32}$ and

$$[(\bigoplus_{t=0}^{p} 2^{25+t}) \& a_{16}] \oplus (\bigoplus_{t=0}^{q} 2^{25+t}) = 0,$$

which is equivalent to $a_{16,32} = c_{16,32}$, $q \leq p$ and

$$a_{16,26+i} = \begin{cases} 1, & \text{if } 0 \leq i \leq q; \\ 0, & \text{if } q < i \leq p. \end{cases}$$

Case 2: $\delta d_{16} mod 2^{25} = 0$, $d^{'}_{16,32} \oplus d_{16,32} = 1$ and there exists a q with $0 \leq q \leq 5$ such that $d^{'}_{16,26+q} = 1$, $d_{16,26+q} = 0$, and $d^{'}_{16,26+t} = 0, d_{16,26+t} = 1$ for each t with $0 \leq t < q$, $d^{'}_{16,26+t} = d_{16,26+t}$ for each t with $q < t \leq 5$; $\delta c_{16} mod 2^{25} = 0$, $c^{'}_{16,32} = c_{16,32}$, and $c^{'}_{16,26+t} = 0, c_{16,26+t} = 1$ for every t with $0 \leq t \leq 5$. Then

$$\delta d_{16} = d^{'}_{16} \oplus d_{16} = 2^{31} \oplus \bigoplus_{t=0}^{q} 2^{25+t} \qquad \textit{and} \qquad \delta c_{16} = c^{'}_{16} \oplus c_{16} = \bigoplus_{t=0}^{5} 2^{25+t}.$$

Hence

$$\begin{aligned}\delta I(c_{16}, d_{16}, a_{16}) &= (\delta c_{16} \& a_{16}) \oplus (\delta a_{16} \& c_{16}) \oplus (\delta c_{16} \& \delta a_{16}) \oplus \delta d_{16} \oplus \delta a_{16} \\ &= (2^{31} \& c_{16}) \oplus [(\bigoplus_{t=0}^{5} 2^{25+t}) \& a_{16}] \oplus (\bigoplus_{t=0}^{q} 2^{25+t}).\end{aligned}$$

So $\delta I(c_{16}, d_{16}, a_{16}) = 2^{31}$ if and only if $c_{16,32} = 1$ and

$$a_{16,26+i} = \begin{cases} 1, & \text{if } 0 \leq i \leq q; \\ 0, & \text{if } q < i \leq 5. \end{cases}$$

Case 3: $\delta d_{16} mod 2^{25} = 0$, $d^{'}_{16,32} = d_{16,32}$, and $d^{'}_{16,26+t} = 0, d_{16,26+t} = 1$ for every t with $0 \leq t \leq 5$; $\delta c_{16} mod 2^{25} = 0$, $c^{'}_{16,32} \oplus c_{16,32} = 1$ and there exists a p with $0 \leq p \leq 5$ such that $c^{'}_{16,26+p} = 1$, $c_{16,26+p} = 0$, and $c^{'}_{16,26+t} = 0, c_{16,26+t} = 1$ for each t with $0 \leq t < p$, $c^{'}_{16,26+t} = c_{16,26+t}$ for each t with $p < t \leq 5$. Then

$$\delta d_{16} = d^{'}_{16} \oplus d_{16} = \bigoplus_{t=0}^{5} 2^{25+t} \qquad \textit{and} \qquad \delta c_{16} = c^{'}_{16} \oplus c_{16} = 2^{31} \oplus \bigoplus_{t=0}^{p} 2^{25+t}.$$

Hence

$$\begin{aligned}\delta I(c_{16}, d_{16}, a_{16}) &= (\delta c_{16} \& a_{16}) \oplus (\delta a_{16} \& c_{16}) \oplus (\delta c_{16} \& \delta a_{16}) \oplus \delta d_{16} \oplus \delta a_{16} \\ &= [(\bigoplus_{t=0}^{p} 2^{25+t}) \& a_{16}] \oplus [2^{31} \& (a_{16} \oplus c_{16})] \oplus (\bigoplus_{t=0}^{5} 2^{25+t}).\end{aligned}$$

So $\delta I(c_{16}, d_{16}, a_{16}) = 2^{31}$ iff $a_{16,32} \neq c_{16,32}$ and

$$(\bigoplus_{t=0}^{p} 2^{25+t}) \& a_{16}] \oplus (\bigoplus_{t=0}^{5} 2^{25+t}) = 0,$$

which is equivalent to $p = 5$ and $a_{16,26+i} = 1$ for each i with $0 \leq i \leq 5$.
Case 4: $\delta d_{16} mod 2^{25} = 0$, $d'_{16,32} = d_{16,32}$, and $d'_{16,26+t} = 0, d_{16,26+t} = 1$ for every t with $0 \leq t \leq 5$; $\delta c_{16} mod 2^{25} = 0$, $c'_{16,32} = c_{16,32}$, and $c'_{16,26+t} = 0, c_{16,26+t} = 1$ for every t with $0 \leq t \leq 5$. Then

$$\delta d_{16} = d'_{16} \oplus d_{16} = \bigoplus_{t=0}^{5} 2^{25+t} \qquad and \qquad \delta c_{16} = c'_{16} \oplus c_{16} = \bigoplus_{t=0}^{5} 2^{25+t}.$$

Hence

$$\begin{aligned}\delta I(c_{16}, d_{16}, a_{16}) &= (\delta c_{16} \& a_{16}) \oplus (\delta a_{16} \& c_{16}) \oplus (\delta c_{16} \& \delta a_{16}) \oplus \delta d_{16} \oplus \delta a_{16} \\ &= [(\bigoplus_{t=0}^{5} 2^{25+t}) \& a_{16}] \oplus (\bigoplus_{t=0}^{5} 2^{25+t}) \oplus (2^{31} \& c_{16}) \oplus 2^{31}.\end{aligned}$$

So $\delta I(c_{16}, d_{16}, a_{16}) = 2^{31}$ iff $c_{16,32} = 0$ and $a_{16,26+i} = 1$ for each i with $0 \leq i \leq 5$.

It is easy to verify that (1)-(4) are equivalent to cases 1-4 respectively. Hence theorem 2 holds. Now we have proved theorem 2.

Theorem 3. *Let $c_{15}, b_{15}, a_{16}, d_{16}, c_{16}, b_{16} \in Z/(2^{32})$ be the chaining values in MD5 algorithm, $\triangle c_{15} = \triangle b_{15} = \triangle a_{16} = 2^{31}$, $\triangle d_{16} = 2^{31} + 2^{25}$, $\triangle m_2 = 0$ and $\triangle m_9 = 0$. Then $\triangle c_{16} = 2^{31} + 2^{25}$ and $\triangle b_{16} = 2^{31} + 2^{25}$ if and only if one of the following conditions is satisfied:*

(1) $d_{16,32} = b_{15,32}$, $c_{16,32} = a_{16,32}$, *for some q, p with $0 \leq q \leq p \leq 5$ such that $d_{16,26+t} = 1$ for each t with $0 \leq t < q$ and $d_{16,26+q} = 0$, and $c_{16,26+t} = 1$ for each t with $0 \leq t < p$ and $c_{16,26+p} = 0$, $b_{15,26+t} = 0$ for each t with $0 \leq t \leq q$ and*

$$a_{16,26+i} = \begin{cases} 1, & \textit{if } 0 \leq i \leq q; \\ 0, & \textit{if } q < i \leq p. \end{cases}$$

(2) $d_{16,32} = b_{15,32}$, *for some q with $0 \leq q \leq 5$ such that $d_{16,26+t} = 1$ for each t with $0 \leq t < q$ and $d_{16,26+q} = 0$, $c_{16,26+t} = 1$ for every t with $0 \leq t \leq 6$, $b_{15,26+t} = 0$ for each t with $0 \leq t \leq q$ and*

$$a_{16,26+i} = \begin{cases} 1, & \textit{if } 0 \leq i \leq q; \\ 0, & \textit{if } q < i \leq 5. \end{cases}$$

(3) $c_{16,32} = a_{16,32} \oplus 1$, $d_{16,26+t} = 1$ *for each* t *with* $0 \le t \le 5$, $c_{16,26+t} = 1$ *for each* t *with* $0 \le t < 5$ *and* $c_{16,31} = 0$, $b_{15,26+i} = 0$ *and* $a_{16,26+i} = 0$ *for each* i *with* $0 \le i \le 5$.

(4) $d_{16,26+t} = 1$ *and* $c_{16,26+t} = 1$ *for each* t *with* $0 \le t \le 5$, $c_{16,32} = 0$, $b_{15,26+i} = 0$ *and* $a_{16,26+i} = 1$ *for each* i *with* $0 \le i \le 5$.

Proof. This is an immediate consequence of theorem 1 and theorem 2.

Remark 3. By theorem 3, we obtain four sets of necessary and sufficient condition to guarantee the output differences of c_{16} and b_{16} as follows:

*Set*1: $c_{16,32} = a_{16,32}$, $d_{16,32} = b_{15,32}$ and for some q, p with $0 \le q \le p \le 5$, there hold

$$d_{16,26+t} = \begin{cases} 1, & \text{if } 0 \le t < q; \\ 0, & \text{if } t = q. \end{cases} \qquad \textit{and} \qquad c_{16,26+t} = \begin{cases} 1, & \text{if } 0 \le l < p; \\ 0, & \text{if } l = p. \end{cases}$$

and $b_{16,26+t} = 0$ for each t with $0 \le t \le q$, and

$$a_{16,26+i} = \begin{cases} 1, & \text{if } 0 \le i \le q; \\ 0, & \text{if } q < i \le p. \end{cases}$$

*Set*2: $d_{16,32} = b_{15,32}$, for some q with $0 \le q \le 5$, there hold

$$d_{16,26+t} = \begin{cases} 1, & \text{if } 0 \le t < q; \\ 0, & \text{if } t = q. \end{cases}$$

and $b_{15,26+t} = 0$ for each t with $0 \le t \le q$ and $c_{16,26+t} = 1$ for each t with $0 \le t \le 6$, and

$$a_{16,26+i} = \begin{cases} 1, & \text{if } 0 \le i \le q; \\ 0, & \text{if } q < i \le 5. \end{cases}$$

*Set*3: $c_{16,32} = a_{16,32} \oplus 1$, $d_{16,26+t} = 1$ for each t with $0 \le t \le 5$, $c_{16,26+t} = 1$ for each t with $0 \le t < 5$ and $c_{16,31} = 0$, $b_{15,26+i} = 0$ and $a_{16,26+i} = 0$ for each i with $0 \le i \le 5$.

*Set*4: $d_{16,26+t} = 1$ for every t with $0 \le t \le 5$, $b_{15,26+i} = 0$ and $a_{16,26+i} = 1$ for each i with $0 \le i \le 5$, and

$$c_{16,26+t} = \begin{cases} 1, & \text{if } 0 \le t \le 5; \\ 0, & \text{if } t = 6. \end{cases}$$

Remark 4. From the description of the above four sets, we get some characteristics of them as follows:

(1) According to the different values of q, p, the *Set*1 includes 21 sets of sufficient conditions. If $q = p = 0$, it is the set presented in [6]. Thus, using the

set of necessary and sufficient condition we will compare the complexity of our algorithm with that of the previous algorithms.

(2) By the different value of q, the $Set2$ include 6 sets of sufficient conditions.

(3) The four sets all include the two conditions $b_{15,26} = 0$ and $a_{16,26} = 1$. Henceforth, the four sets by removing $b_{15,26} = 0$ and $a_{16,26} = 1$ are denoted by $Set1^{'}$, $Set2^{'}$, $Set3^{'}$ and $Set4^{'}$ respectively.

(4) Since $Set3^{'}$ includes 23 conditions, the probability that all the conditions in $Set3^{'}$ are satisfied is $1/2^{23}$. Similarly, the probability that all the conditions in $Set4^{'}$ are satisfied is also $1/2^{23}$.

Similarly, for the second iteration, we can also obtain the set of necessary and sufficient condition to guarantee the differential path from 24-64 steps.

3.3 Our Attack Algorithm for Finding a Collision

According to proposition 1 and theorem 3, we get the sets of necessary and sufficient conditions to guarantee the output differences from 24-64 steps in the first iteration. However, the previous algorithms [6] [7] [9] only use the sufficient conditions to test the output differences, which reduce the probability that the algorithms continue to the next step. So, in our algorithm we will directly test whether the output differences are what we needed. Now, we will outline our attack algorithm for the first block and the second block respectively. (Refer to the sets of sufficient conditions presented by Jie Liang and Xuejia Lai in [6])

The Algorithm for the First Block. (1) Select random 32-bit values for each m_i with $0 \leq i \leq 15$.

(2) Using single-message modification to make the sufficient conditions from step 1 to step 16 satisfied as in [6].

(3) Randomly select a 32-bit values for a_5 and make it satisfy the sufficient conditions in table 3 of [6]. Then update m_0,d_1,m_2,m_3,m_4,m_5.

(4) Use the multi-message modification techniques described in [10] to make the sufficient conditions from 18-22 steps satisfied deterministically.

(5) Compute the remaining steps. For each step test whether the output differences holds. If the output differences are not all held, jump to step 3. Then compute the initial values of the second iteration, if the conditions of the initial values in the table 3 of [6] are not all satisfied, jump to step 3. Otherwise, output $m_0, m_1, \ldots, m_{15}$.

The Algorithm for the Second Block. (1) Select random 32-bit values for each m_i with $0 \leq i \leq 13$.

(2) Use single-message modification to make the chaining values from step 1 to the step 14 satisfy the sufficient conditions as in [6].

(3) Randomly select two 32-bit values for c_4 and b_4 respectively but make them satisfy the sufficient conditions as in [6], then compute the values of m_{14} and m_{15}.

(4) Use multi-message modification described in [10] to make the sufficient conditions from 17-22 steps satisfied deterministically.

(5) Compute the remaining steps. For each step test whether the output differences holds. If the output difference are not all held, jump to step 3. Otherwise, output $m_0, m_1, \ldots, m_{15}$.

3.4 The Comparison between the Complexity of Our Algorithm and the Previous Algorithms

Since we increase the probability that the differences of c_{16} and b_{16} hold by using the set of necessary and sufficient conditions and the c_{16} and b_{16} are the last two output differences to be held in attack algorithms, our algorithm can improve the computational complexity of the previous attack algorithms [6][7][9] greatly. The probabilities that the output differences of c_{16} and b_{16} hold are denoted by Pr_{block1} and Pr_{block2} respectively, while in the previous algorithms they are denoted by $Pr^{'}_{block1}$ and $Pr^{'}_{block2}$ respectively. Now, the probability that at least one set in $Set1^{'}$ is satisfied is denoted by Pr^1 and the probability that at least one set in $Set2^{'}$ is satisfied is denoted by Pr^2.

Theorem 4. *Let $b_{15}, a_{16}, d_{16}, c_{16} \in Z/(2^{32})$ be the chaining values in MD5 algorithm and for some q, p with $0 \le q \le p \le 5$,*

$$\begin{aligned}
D^1_q &= \{(d_{16,26}, \ldots, d_{16,31}) \mid (d_{16,26}, \ldots, d_{16,25+q}, d_{16,26+q}) = (1, \ldots, 1, 0)\},\\
C^1_p &= \{(c_{16,26}, \ldots, c_{16,31}) \mid (c_{16,26}, \ldots, c_{16,25+p}, c_{16,26+p}) = (1, \ldots, 1, 0)\},\\
A^1_{q,p} &= \{(a_{16,27}, \ldots, a_{16,31}) \mid (a_{16,27}, \ldots, a_{16,26+q}) = (1, \ldots, 1)\\
&\qquad \textit{and} \quad (a_{16,27+q}, \ldots, a_{16,26+p}) = (0, \ldots, 0)\},\\
B^1_i &= \{(b_{15,27}, \ldots, b_{15,31}) \mid (b_{15,27}, \ldots, a_{16,26+q}) = (0, \ldots, 0)\},\\
\Lambda^1_{q,p} &= \{(a, b, c, d) | a \in A^1_{q,p}, b \in B^1_q, d \in D^1_q, c \in C^1_p\}. \qquad (1)
\end{aligned}$$

Then $Pr^1 \approx 0.347692$, $Pr^{'}_{block1} = 0.25$.

Proof. For any $q^{'} \ne q$ or $p^{'} \ne p$,$0 \le q \le p \le 5$ and $0 \le q^{'} \le p^{'} \le 5$, since $D^1_{q^{'}} \cap D^1_q = \emptyset$,$C^1_{p^{'}} \cap C^1_p = \emptyset$,$A^1_{q^{'}p^{'}} \cap A^1_{qp} = \emptyset$ or $B^1_{q^{'}} \cap B^1_q = \emptyset$, we have $\Lambda^1_{q^{'}p^{'}} \cap \Lambda^1_{qp} = \emptyset$. Moreover, from equation 1 we have $|\Lambda^1_{qp}| = 2^{6-(p+1)} \times 2^{5-q} \times 2^{6-(q+1)} \times 2^{5-p} = 2^{2(10-q-p)}$. Therefore,

$$|\bigcup_{0 \le q \le p \le 5} \Lambda^1_{qp}| = \sum_{0 \le q \le p \le 5} |\Lambda^1_{qp}| = \sum_{0 \le q \le p \le 5} 2^{2(10-q-p)} = 2^{20} \times 1.390782.$$

Thus, we have

$$Pr^1 = \frac{|\bigcup_{0 \le q \le p \le 5} \Lambda^1_{qp}|}{2^{22}} = \frac{2^{20} \times 1.390782}{2^{22}} \approx 0.347692.$$

If $q = p = 0$, from (1) of remark 4 we know that it is the case in the previous algorithms, so we have $Pr^{'}_{block1} = 0.25$.

Theorem 5. *Let* $b_{15}, a_{16}, d_{16}, c_{16} \in Z/(2^{32})$ *be the chaining values in MD5 algorithm and for some* q *with* $0 \leq q \leq 5$,

$$
\begin{aligned}
D_q^2 &= \{(d_{16,26}, \ldots, d_{16,31}) \mid (d_{16,26}, \ldots, d_{16,25+q}, d_{16,26+q}) = (1, \ldots, 1, 0)\}, \\
C^2 &= \{(c_{16,26}, \ldots, c_{16,32}) \mid (c_{16,26}, \ldots, c_{16,32}) = (1, \ldots, 1)\}, \\
A_q^2 &= \{(a_{16,27}, \ldots, a_{16,31}) \mid (a_{16,27}, \ldots, a_{16,26+q}) = (1, \ldots, 1) \\
&\quad \textit{and} \quad (a_{16,27+q}, \ldots, a_{16,31}) = (0, \ldots, 0)\}, \\
B_q^2 &= \{(b_{15,27}, \ldots, b_{15,31}) \mid (b_{15,27}, \ldots, a_{16,26+q}) = (0, \ldots, 0)\} \\
\Lambda_q^2 &= \{(a, b, c, d) | a \in A_q^2, b \in B_q^2, d \in D_q^2, c \in C^2\}. \qquad (2)
\end{aligned}
$$

Then $Pr^2 \approx 1.627604 \times 10^{-4}$.

Proof. Using the same way as theorem 4, we prove theorem 5.

Since the necessary and sufficient conditions sets are independent, by theorem 4, theorem 5 and (4) of remark 4 we get $Pr_{block1} = Pr^1 + Pr^2 + 2 \times 2^{23} \approx 0.347854$. Similarly, we obtain $Pr_{block2} \approx 0.347854$.

The computational complexity that all the sufficient conditions given in [6] except for $d_{16,26} = 0$ and $c_{16,26} = 0$ for the first iteration are satisfied is denoted by $C_{block1}(M_0)$ and except for $d_{16,26} = 1$ and $c_{16,26} = 1$ for the second iteration we denote $C_{block2}(M_1)$. Assume that each step is independent and in the t-th step of our attack algorithm the output differences of the c_{16} and b_{16} hold, then the probability that the differences of c_{16} and b_{16} hold in our algorithm is $(1 - Pr_{block1})^{t-1} \times Pr_{block1}$, the mathematical expectation of which is $\frac{1}{Pr_{block1}} = \frac{1}{0.347854}$. So, the computational complexity of our algorithm is $\frac{C_{block1}(M_0)}{0.347854}$ comparing to $\frac{C_{block1}(M_0)}{0.25}$ in the previous algorithms [6] [7] [9], which implies that the computational complexity of our algorithm for the first block

Table 1. An Example of the Comparison of the Experimental Times(hexadecimal)

LL's Attack: Block1(M_0)	LL's Attack: Block2(M_1)
ba9dc4f4 d53ada82 da3101a8 aa5240bd 599a49ab f6a6ca92 702331b6 6e1028a8 05332dc1 c0b4e7f8 7b5afb1b 8fb0a974 c5be6d0b 090e4d32 29c57fe9 c6a2712f	2f6f6903 e1a8c33f 02ad55ff e3672c0f b77a9752 fdbff0b7 177bccfe b2bcdad6 357b278d 8cb756a9 7b7c02fa ddb0e47f b644d0bc f8dbbcaa 68bdbedd 9c119e29
LL's Runtime for Block1: 1807min	LL's Runtime for Block2: 253min
Our's Attack: Block1(M_0)	Our's Attack: Block1(M_1)
ba9dc4f4 d53ada82 da3101a8 aa5240bd 599a49ab f6a6ca92 702331b6 6e1028a8 05332dc1 c0b4e7f8 7b5afb1b 8fb0a974 c5be6d0b 090e4d32 29c57fe9 c6a2712f	2f6f6903 21a8c73f 00ad51df e3672c0f b73a9352 fd7fecb7 176bccfe b2bd5ad6 357b27ad 8cb756c9 7b7c031a 1db1087b a5b6b0bc f8dbbca2 760d7e9c 7d34f5c3
Our's Runtime for Block1: 1350min	Our's Runtime for Block2: 23min

is $\frac{0.25}{0.347854} = 0.718692$ times of that of the previous algorithms. Similarly, we can obtain that the computational complexity of our algorithm for the second block is also 0.718692 times of the previous algorithms. In the previous attack algorithms, there are 33 conditions for the first block and 27 for the second block satisfied by trial and error, so the average computational complexity is about 0.718692×2^{33} times MD5 operations(See Table 1 for an comparison of the experimental times).

4 Conclusion

In this paper, we have presented an improved attack algorithm on MD5. By analyzing the properties of the nonlinear Boolean functions used in MD5 and the differences in term of XOR and subtraction modulo 2^{32}, we prove that the sufficient conditions from 24-61 steps but step 35 presented by Jie Liang and Xuejia Lai are also necessary to guarantee the differential path and give the set of necessary and sufficient conditions for the last two output differences. For step 35 and step 62, the sufficient conditions are also necessary to ensure that the left shift rotation and subtraction modulo 2^{32} can be commuted and the output differences are held. Finally, we analyze the computational complexity of our attack algorithm which is 0.718692 times of that of the previous collision attack algorithms and test the result by computer simulations.

References

1. Hawkes, P., Paddon, M., Rose, G.G.: Musings on the Wang et al. MD5 collision. Cryptology ePrint Archive, Report 2004/264 (2004)
2. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
3. Yajima, J., Shimoyama, T.: Wang's sufficient conditions of MD5 are not sufficient. Cryptology ePrint Archive, Report 2005/263 (2005)
4. Nakano, Y., Kuwakado, H., Morii, M.: Redundancy of the Wang-Yu sufficient conditions. Cryptology ePrint Archive, Report 2006/406 (2006)
5. Sasaki, Y., Naito, Y., Yajima, J., et al.: How to construct sufficient condition in searching collisions of MD5. Cryptology ePrint Archive, Report 2006/074 (2006)
6. Liang, J., Lai, X.: Improved collision attack on hash function MD5. Cryptology ePrint Archive, Report 2005/425 (2005)
7. Wang, Z., Zhang, H., Qin, Z., Meng, Q.: A fast attack on the MD5 hash function. Journal of Shanghai Jiaotong University 2 (2006)
8. Stevens, M.: Fast Collision Attack on MD5. Cryptology ePrint Archive, Report 2006/104 (2006)
9. Klima, V.: Finding MD5 collisions on a notebook PC using multi-message modifications. Cryptology ePrint Archive, Report 2005/102 (2005)
10. Sasaki, Y., Naito, Y., Kunihiro, N., et al.: Improved collision attack on MD5. Cryptology ePrint Archive, 2005/400 (2005)
11. Klima, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006/105 (2006)
12. Yu, H., Wang, X.: Multicollision attack on the compression functions of MD4 and 3-pass HAVAL. Cryptology ePrint Archive, Report 2007/085 (2007)

Multivariates Polynomials for Hashing

Jintai Ding[1] and Bo-Yin Yang[2]

[1] University of Cincinnati and Technische Universität Darmstadt
ding@math.uc.edu
[2] Institute of Information Science, Academia Sinica
by@moscito.org

Abstract. We propose the idea of building a secure hash using quadratic or higher degree multivariate polynomials over a finite field as the compression function. We analyze some security properties and potential feasibility, where the compression functions are randomly chosen high-degree polynomials, and show that under some plausible assumptions, high-degree polynomials as compression functions has good properties. Next, we propose to improve on the efficiency of the system by using some specially designed polynomials generated by a small number of random parameters, where the security of the system would then relies on stronger assumptions, and we give empirical evidence for the validity of using such polynomials.

Keywords: hash function, multivariate polynomials, sparse.

1 Introduction

There is a rather pressing need to find new hash functions as of today, after the work of Wang et al culminated in collisions of some well-known and standard hash functions [13,14]. One common feature of the currently used hash functions is that it is more an area of art, where the design of the system is based on certain procedures and the security of the system is very very difficult to study from the theoretical point of view. For example, even the work of Wang et al is still not well understood and people are still trying to analyze the methods used in some systematical way. [12]

Naturally, one direction is to look for provably secure hash functions, whose security relies on well-understood hard computational problems. These hashes tend to be slower, although they have a saving grace in terms of having some measure of provable security.

In these formulations, the designer seeks a reduction of the security of the compression function to some other intractable problem. Of course, we should be careful about these "provably secure" constructions. There are two pitfalls:

– Often, the security reduction has a large looseness factor. The practical result is that these reductions end up "proving" a very low security level — the complexity of the "underlying hard problem" divided by the looseness factor — which in a practically-sized instance will be insufficient to satisfy security requirements [10,9,17].

It should be mentioned, however, that often times this kind of reductions shows that if hard problem A is exponential in a power of n, then so is cryptosystem B under suitable assumptions. So having a proof still beats not having one because it implies that there are no real serious gotchas.
- The other possibility is that the security of the hash function may only loosely depend on the hard problem. The case of NTRU's signature schemes exemplify this possibility. One can only say in this case that the scheme is inspired by the hard problem.

In this paper, we propose our own version, which is inspired by the $\mathcal{M}Q$ problem, and study how well it can work. Our paper is arranged as follows. We first study the case of random systems. Then we study the case of sparse construction and present the main theoretical challenges and its practical applications. We will then discuss other new ideas and the future research in the conclusion. Much work remains in this area in terms of security reductions and speed-ups.

2 The MQ Frame Work

Problem $\mathcal{M}Q$: Solve a polynomial system, with coefficients and variables in $K = \mathrm{GF}(q)$, where each p_i is randomly chosen and quadratic in $\mathbf{x} = (x_1, \ldots, x_n)$.

$$p_1(x_1, \ldots, x_n) = 0,\ p_2(x_1, \ldots, x_n) = 0,\ \ldots,\ p_m(x_1, ..., x_n) = 0, \qquad (1)$$

$\mathcal{M}Q$ is NP-hard generically [7]. If instead of quadratics, each p_i is of degree $d_i > 1$, the problem may be termed called $\mathcal{M}P$, which is of course no easier than $\mathcal{M}Q$, so must also be NP-hard. That a problem is NP-hard generically need not mean that its solution is of exponential time complexity in its parameters *on average*, or imply anything about the coefficients. However, today, as m and n increases to infinity, we believe that the following holds.

Conjecture 2.1. $\forall \epsilon > 0$, $\Pr(n/m = \Theta(1)$, $\mathcal{M}Q$ can be solved in $\mathrm{poly}(n)) < \epsilon$.

The conjecture above or something similar to it is the basis of multivariate public-key cryptosystems [6,15] as well as some symmetric cryptosystems [3]. The latest way of tackling such systems involve guessing at some optimal number of variables (depending on the method in the closing stage) then use a Lazard-Faugère solver (XL or $\mathbf{F_5}$, [2]). We also believe, and it is commonly accepted that the complexity to solve the set of equations is indeed exponential, if $p_i(x_1, \ldots, x_n)$ are polynomials of a fixed degree whose coefficients are randomly chosen — say that $p_i(x_1, ..., x_n)$ are randomly chosen quadratic polynomials. We will assume this statement as the security foundation.

Under this assumption, the first straightforward idea is to build an iterated hash using completely random quadratic polynomials as compression functions, namely the one way function $F : K^{2n} \to K^n$ is given by

$$F(x_1, \ldots, x_n, y_1, \ldots, y_n) = (f_1(x_1, \ldots, x_n, y_1, \ldots, y_n), ..., f_n(x_1, ..., x_n, y_1, \ldots, y_n)),$$

where each f_i is a randomly chosen quadratic polynomial over GF(q). All the coefficients are chosen randomly. We will use this as a compressor with state $\mathbf{x} = (x_1, \ldots x_n)$, and each input message block is $\mathbf{y} = (y_1, \ldots, y_n)$. In the following, we show that this cannot work and suggest the next improvements, particularly cubic and stacked (composed) quadratics, and the idea of using the least number of parameters to determine our maps.

2.1 General Remark on Solvability

A rough estimate of effort to solve these $\mathcal{MQ}$ problems: 20 quadratic equations in 20 GF(256) variables: 2^{80} cycles; 24 quadratic in 24 GF(256) variables: 2^{92} cycles. It is harder to solve higher-order equations: for reference, 20 cubic equations in 20 GF(256) variables takes about 2^{100} cycles, 24 cubics in 24 GF(256) variables about 2^{126} cycles, and 20 quartics in in 20 GF(256) variables about 2^{128} cycles.

Clearly, the problem for our hash schemes is not going to be the direct solution (algebraic attack) using a polynomial system solver. The above shows that any multivariate polynomial systems are not really susceptible to preimage or second preimage attacks.

Note: There is a special multivariate attack to solve under-defined systems of equations [5] that applies to this situation where there is a lot many more variables than equations, but

– the number of variables that it reduces is proportional to the square root of n, and
– for $q > 2$ under most optimistic estimates it has proved to be rather useless.

We would then, of course, try multivariate quadratics as the obvious idea. However, it is not secure because collisions are easy to find:

Proposition 2.1. *With randomly chosen $F := F(\mathbf{x}, \mathbf{y})$, it is easy to solve the equation.*

$$F(\mathbf{x}_1, \mathbf{y}_1) = F(\mathbf{x}_2, \mathbf{y}_2)$$

Proof

$$F(\mathbf{x} + \mathbf{b}, \mathbf{y} + \mathbf{c}) - F(\mathbf{x}, \mathbf{y}) = 0$$

is a linear equation in $2n$ variables $(\mathbf{x}, \mathbf{y})$ and n equations, so must be solvable.

However, we this points out a better way to building a secure hash.

3 The Applications of Sparsity

In Section 3 we show that the compressor and hence the hash function is likely to be collision-free. In the following sections, we propose some naive instances, which has the security level at 2^{80} or 2^{128}, but those system are very slow for practical applications. A natural idea is to use sparse polynomials. This is as in the case of the central maps of TTS signature schemes, namely we will choose each of the above polynomial to be a sparse polynomial. However, the security relies on the following stronger assumption:

Conjecture 3.1. As $n \to \infty$ and $m/n \to k \in \mathbb{R}^+$, for any fixed $0 < \epsilon < 1$ a random sparse polynomial system of any fixed degree with a randomly picked ϵ proportion of the coefficients being non-zero (and still random, if $q > 2$) will still take time exponential in n to solve.

This can be termed a version of the conjecture S$\mathcal{M}P$ (sparse multivariate polynomial). S$\mathcal{M}P$ is only one of the many conjectures that we can make, because as long as we are not dealing with packed bits, the speed of the implementation depends more on the size of the parameter set than how we use the parameters. If a set of polynomials is determined from a relatively small set of parameters, we can call them "sparsely determined".

There is no real reduction known today that reduces S$\mathcal{M}P$ to a known hard problem. However, we can justify this assumption somewhat by trying to solve these sparse systems, and by extension sparsely generated systems, using the best tools we have.

– If we solve the system with an XL-with-Wiedemann like attack [16,17], it is clear that the running time will simply be ϵ times that of the corresponding dense problem.
– There is no commercially available implementation of $\mathbf{F_5}$, however, it runs at the same degree as $\mathbf{F_4}$ and mostly differ in the fact that $\mathbf{F_5}$ avoids generating extraneous equations that are bound to be eliminated.
– The most sophisticated solver commercially available is MAGMA [4]. We ran many tests on random systems, random sparse systems, and not necessarily sparse but sparsely generated systems. In every case, solving the sparse determined systems takes roughly the same amount of time and memory as the non-sparse ones.
– We know of some specialized methods that solves sparse systems, but some have a different definition of sparsity than we have here [11], and some are not well-quantified.

Please see the appendix for the tests that we ran. Clearly, the key is to choose wisely a correct proportion ϵ of the nonzero terms.

1. How many we choose such that it is fast?
   Our answer: no more than maybe 0.1% (see below).
2. How many we choose such that it is secure?
   Our answer: a predetermined fixed percentage.
3. How do we choose the sparse terms?
   Our answer: probably randomly.

4 Cubic Construction

Suppose that $q = 2^k$ and the F above is changed to a random *cubic* $K^{2n} \to K^n$, then the following argument that this compression function is secure.

Let $\bar{K}$ be a degree n extension of K. We have a map ϕ which identifies $\bar{K}$ as K^n.

Then it is clear from the work of Kipnis and Shamir [8] we can show that F can be lifted to be a map from $\bar{K} \times \bar{K}$ to $\bar{K}$. Then we have

$$\bar{F}(X,Y) = \sum A_{ijk} X^{q^i+q^j+q^k} + \sum B_{ijk} X^{q^i+q^j} Y^{q^k} +$$
$$\sum C_{ijk} X^{q^i} Y^{q^j+q^k} + \sum D_{ijk} Y^{q^i} X^{q^j+q^k} +$$
$$\sum E_{ij} X^{q^i} Y^{q^j} + \sum F_{ij} X^{q^i} X^{q^j} + \sum G_{ij} Y^{q^i} Y^{q^j} +$$
$$\sum H_i X^{q^I} + \sum I_i Y^{q^i} + J.$$

In this case, we can view all the coefficients as random variables.

We can show that no matter how one the choose the difference of the (X, Y) coordinates that the coefficients of the difference, namely

$$\bar{F}(X + X', Y + Y') - \bar{F}(X, Y)$$

are still linearly independet random variables as a quadratic function.

By mathematical induction, we can prove the following theorem.

Proposition 4.1. *Define a function in the Kipnis-Shamir form*

$$\bar{F}(X,Y) = \sum A_{ijk} X^{q^i+q^j+q^k} + \sum B_{ijk} X^{q^i+q^j} Y^{q^k} \sum C_{ijk} X^{q^i} Y^{q^j+q^k} + \sum D_{ijk} Y^{q^i+q^j+q^k}$$
$$+ \sum E_{ij} X^{q^i} Y^{q^j} + \sum F_{ij} X^{q^i} X^{q^j} + \sum G_{ij} Y^{q^i} Y^{q^j} + \sum H_i X^{q^I} + \sum I_i Y^{q^i} + J.$$

where each coeffcients are linear independent random variables and i, j, k are less than a fixed integer r, the nonzero coefficients of the function

$$\bar{F}(X + X', Y + Y') - \bar{F}(X, Y)$$

are also linearly independ random variables.

Proof. Let us first assume that r is zero, we have that

$$\begin{aligned}
&\bar{F}(X + X', Y + Y') - \bar{F}(X, Y)\\
&= (3A_{000}X' + B_{000}Y')X^2 + (C_{000}X' + 3D_{000}Y')Y^2 + (2X'B_{000} + 2Y'C_{000})XY\\
&+ (3A_{000}X'^2 + 2X'Y'B_{000} + Y'^2C_{000} + E_{00}Y' + 2F_{00}X')X\\
&+ (3D_{000}Y'^2 + 2X'Y'C_{000} + X'^2B_{000} + 2G_{00}Y' + E_{00}X')Y\\
&+ A_{000}X'^3 + B_{000}X'^2Y' + C_{000}X'Y'^2 + D_{000}Y'^3 + E_{00}X'Y' + F_{00}X'^2 + G_{00}Y'^2 + H_0X' + I_0Y'
\end{aligned}$$

The first three terms' coefficients are clearly independent, and involves only the $A_{000}, B_{000}, C_{000}, D_{000}$, as long as (X', Y') are not both zero. Then in the fourth and the fifth term we have $E_{00}Y' + 2F_{00}X'$ in X term and $2G_{00}Y' + E_{00}X'$ in the Y term. Therefore the initial 5 coefficients are linearly independent. Finally, we know that $H_0X' + I_0Y'$ in the constant term ensures that all the coefficients are linearly independent.

Clearly we can see that we write the difference above in the form of

$$\bar{F}(X + X', Y + Y') - F(X, Y) = \bar{A} \times Q \times Z^t,$$

according to the order of degree of the (X, Y) terms, where

$$\bar{A} = (A_{000}, ..., I_0),$$

and Q is the coefficient matrix (which may depend on X', Y') and

$$Z = (X^2, Y^2, XY, X, Y, 1)$$

Then Q clearly have a blockwise triangular structure, which makes the proof possible.

Then suppose the theorem is proved when $i, j, k < r$, we can use the same trick to proceed to $i, j, k \leq r$, namely we will write down

$$\bar{F}(X + X', Y + Y') - \bar{F}(X, Y) = \bar{A} \times Q \times Z^t,$$

where Z is arranged in the form that the first block are the terms involves terms that either have X^{q^r} or Y^{q^r}, then the rest. The $\bar{A}$ is arranged in the same way that the first block contains the terms whose subindices must have r. Then we can write down the expresssion explicitly and show the independence, just as in the case $r = 0$. We will omit the tedious detail here.

Corollary 4.1. *For a quadratic map $M : K^{2n} \to K^n$ with uniformly chosen random coefficients, we can construct a cubic map $\bar{F} : K^{2n} \to K^n$ and a differential X', Y' such that $M(X, Y) = \bar{F}(X + X', Y + Y') - \bar{F}(X, Y)$, such that X', Y' and $\bar{F}$ have uniformly random coefficients or components.*

From this proposition, we can infer directly the following important conclusions.

Proposition 4.2. *If $\bar{F}$ is randomly chosen,the function $\bar{F}(X + X', Y + Y') - \bar{F}(X, Y)$ is also random as long as X' and Y' are not both zero.*

Therefore we have

Proposition 4.3. *If F is randomly chosen,the function $F(\mathbf{x}+\mathbf{b}, \mathbf{y}+\mathbf{c})-F(\mathbf{x}, \mathbf{y})$ is also random as long as $\mathbf{b}$ and $\mathbf{c}$ are not both zero.*

Theorem 4.1. *A random cubic $F : K^{2n} \to K^n$ (written as $F := F(\mathbf{x}, \mathbf{y})$, $\mathbf{x}, \mathbf{y} \in K^n$) is*

1. *impossible to invert or to find a second image for, in polynomial time.*
2. *is impossible find a collision for, in polynomial time.*

Proof. From the proposition above, we can assume a attacker knows the difference of the collision he or she intens to find. In this case, it means that the attacker have the equation $F(\mathbf{x} + \mathbf{b}, \mathbf{y} + \mathbf{c}) - F(\mathbf{x}, \mathbf{y}) = 0$ that he or she must solve if he or she can find the collision. However, we just showed that no matter how one chooses $\mathbf{b}$ and $\mathbf{c}$, the equation can be viewed as a random equation. Therefore it is impossible to solve in polynomial time.

5 Random Cubic Polynomial

We use completely random cubic polynomials, namely the one way compression function is given by

$$F(\mathbf{x}, \mathbf{y}) = (f_1(\mathbf{x}, \mathbf{y}), ..., f_n(\mathbf{x}, \mathbf{y})),$$

where f_i is a randomly chosen cubic polynomial over GF(q). Here $\mathbf{x}, \mathbf{y} \in K^n$ as before. All the coefficients are chosen randomly. We will use this as a compressor in a Merkle-Damgård iterated compression hash function. For example we may use the following Merkle-Damgård like structure:

State: $\mathbf{x} := (x_1, x_2, \ldots, x_n)$.
Incoming Data Block: $\mathbf{y} := (x_{n+1}, \ldots, x_{2n})$.
Compression: $F : (\mathrm{GF}(q))^{2n} \to (\mathrm{GF}(q))^n$.
Initial State: $F(P(a_1, \ldots, a_{n/2}), P(a_{n/2+1}, \ldots, a_n))$, P is a random given quadratic $K^{n/2} \to K^n$.

According to [3], the output of P is impossible to predict or distinguish from random.

Final Output: in $(\mathrm{GF}(q))^n$, possibly with some post-treatment.

Assuming 160-bit hashes (SHA-1), preliminary runs with a generic F:

- 40 GF(256) variables into 20: 67300 cycles/byte (6.0 cycles/mult)
- 80 GF(16) variables into 40: 4233000 cycles/byte (3.2 cycles/mult)

Assuming 256-bit hashes (SHA-2), preliminary runs with a generic F:

- 32 GF(2^{16} variables into 16: 48200 cycles/byte (19 cycles/mult)
- 64 GF(256) variables into 32: 3040000 cycles/byte (6.0 cycles/mult)
- 128 GF(16) variables into 64: 9533000 cycles/byte (3.2 cycles/mult)

Some implementation details:

- The coefficients of the map F is taken from the binary expansion of π.
- Multiplication in GF(16) and GF(256) are implemented with tables. In fact, in GF(16), we implement a 4kBytes table with a where we can multiply simultaneously one field element by two others.
- Multiplication in GF(2^{16}) is implemented via Karatsuba multiplication over GF(256).

But using a random cubic system is not very efficient in general, the new idea is the next one namely we will use sparse polynomials.

5.1 Sparse Cubic Polynomial

The idea is the same as above but we choose each of the above cubic polynomial to be sparse. Note that due to the new result by Aumasson and Meier [1], we now advise that only the *non-affine* portion of the polynomials be sparse.

The key point is to pick a the ratio ϵ of the nonzero terms. In general, storing the formula in a sparse form takes extra space and time to unscramble the storage, so it is never worthwhile to have $\epsilon > 1/10$ or so in practice. In the examples in the following, less than 0.2% of the coefficients are non-zero (one in 500). To give one example, there is around 30 terms per equation in a 40-variable cubic form.

Assuming 160-bit hashes (SHA-1), preliminary runs with a generic sparse F:

- 40 GF(256) variables into 20, 0.2%: 215 cycles/byte
- 80 GF(16) variables into 40, 0.1%: 4920 cycles/byte

Assuming 256-bit hashes (SHA-2), preliminary runs with a generic sparse F:

- 32 GF(2^{16} variables into 16, 0.2%: 144 cycles/byte
- 64 GF(256) variables into 32, 0.1%: 3560 cycles/byte
- 128 GF(16) variables into 64, 0.1%: 9442 cycles/byte

6 Stacked (Composed) Quadratics

Having noted that cubics don't work so well, another way is to have quartics which are composed of quadratics. The first quadratic maps $2n$ variables to $3n$, the second one then maps the $3n$ to n. This avoids the problem by using a set of quartic that can still be computed relatively quickly.

The first question is, whether this is a good idea.

Proposition 6.1. *Let the compression function F be equal to $F_2 \circ F_1$, with generic $F_1 : K^{2n} \to K^{3n}$, $F_2 : K^{3n} \to K^n$.*

Proof. Schematically, $F_2^{-1} = F_1 \circ F^{-1}$. Hence If we can invert F, then we can invert F_2. This is hard by assumption.

Proposition 6.2. *F is collision-resistant.*

Proof (Heuristic). We note that F_1 has no collisions on average, and if it has a pair it is hard to find. Thus, a collision on F means a collision on F_2, which poses no problem, but then we will have to solve F_1 twice, which is difficult by assumption.

Now, let us consider a direct differential attack and compute $F^{(\mathbf{b},\mathbf{c})}$. Using the chain rule, all the polynomials belong to the ideal generated by the polynomials of F_1. Since for generic polynomials we should not see a reduction to zero under the degree of regularity [2], the solution is at as hard as inverting F_1.

Assuming 160-bit hashes (SHA-1), preliminary runs with a function $F = F_2 \circ F_1$, with generic $F_1 : K^{2n} \to K^{3n}$, $F_2 : K^{3n} \to K^n$

- 40 GF(256) variables into 20: 27400 cycles/byte
- 80 GF(16) variables into 40: 101200 cycles/byte

Assuming 256-bit hashes (SHA-2), preliminary runs with $F = F_2 \circ F_1$, generic F_1, F_2:

– 32 GF(2^{16} variables into 16: 24100 cycles/byte
– 64 GF(256) variables into 32: 64200 cycles/byte
– 128 GF(16) variables into 64: 247000 cycles/byte

In this way we can explore another form of sparsity, which relies on the idea is that any quadratic map can be written as $f \circ L$, where f is a certain standard form and L is an invertible linear map. Now we will instead choose L to be sparse.

In this instance, the key question are still the same:

1. How many we choose such that it is fast?
2. How many we choose such that it is secure?
3. How do we choose the sparse terms?

In this area, we note that it is not necessary that L be sparse, but only that it be decided by a relatively small number of parameters, and that the evaluation is fast. Along these lines, a new construction is the continued sparse compositions, where we use composition of sparse random linear maps. We propose some instances of these hash functions for practical applications. There are furthermore several new ideas that should be further studied.

6.1 Sparse Composition Factor: "Rotated" Quadratic Sets

The idea is that any quadratic map can be written as $f \circ L$, where f is a certain standard form and L is an invertible affine map. Now we will choose the linear part of L to be sparse. The obvious standard form for characteristic 2 fields is to start with the standard form ("rank form").

$$f_1(\mathbf{x}) = x_1x_2 + x_3x_4 + \cdots x_{n-1}x_n.$$

Let us explain a little why. Let k be a field of characteristic 2. A quadratic form in n variables over k is defined by $Q = \sum_{1 \le i \le j \le n} p_{ij}x_ix_j$, $p_{ij} \in F$.

Theorem 6.1. *Any quadratic form over k is equivalent to*

$$Q' = \sum_{i=1}^{\nu} x_iy_i + \sum_{j=\nu+1}^{\tau+\nu} \left(a_jx_j^2 + x_jy_j + b_jy_j^2\right) + \sum_{k=1}^{d} c_kz_k^2$$

with $c_k \neq 0$ and $2\nu + 2\tau + d \le n$.

When $2\nu + 2\tau + d = n$, the form Q' is regular. The number d is the deficiency of the form and the form q' is completely regular, if Q' is regular and its deficiency is zero, which corresponding the case that the corresponding symmetric form is non-degenerate. A randomly chosen is in general expected to completely regular.

Furthermore, we have

Theorem 6.2. *If two quadratic forms over k, $\psi + q_1$ and $\psi + q_2$, are equivalent and ψ is completely regular, then q_2 and q_1 are equivalent over F.*

We will use this to give a general characterization of a quadratic function. Any quadratic function $f(x_1, ..., x_{2n})$ can be written in the form

$$f(x_1, .., x_n) = \sum_{1 \le i \le j \le 2n} a_{ij} x_i x_j + \sum_{1 \le i \le 2n} b_i x_i + c$$

where a_{ij}, b_i, c are in k. We know that through a linear transformation of the form $L(x_i) = x_i + d_i$, if the quadratic part is non degenerate, we can have that

$$f(L_1(x_1, .., x_n)) = \sum_{1 \le i \le j \le 2n} a'_{ij} x_i x_j + c'.$$

From the theorem above we know that there is a linear map L_2 such that

$$f((L_2 \circ L_1)(x_1, .., x_n)) = \sum_{1 \le i \le n} x_{2i-1} x_{2i} + \sum_{i \in S} x_i^2 + c',$$

where S is a set consisting of pairs in the form of $(2j-1, 2j)$. The simplest form this kind of function is surely

$$f((L_2 \circ L_1)(x_1, .., x_n)) = \sum_{1 \le i \le n} x_{2i-1} x_{2i} + c',$$

and its difference from others in some sense are something of the linear nature, which can be neglected in some way. From this we conclude that a general quadratic function can be represented as: $F \circ L$, where

$$F = \sum_{1 \le i \le n} x_{2i-1} x_{2i} + c',$$

which is the basis we will use to build our hash function.

Knowing that any quadratic functions from $\mathrm{GF}(q)^k \to \mathrm{GF}(q)$ can be written this way. The key question are similar now: *How do we choose L such that it is fast? How many we choose such that it is secure? How do we choose the sparse terms?*

In this particular instance, there is something that leaps out at us. starting with $\mathbf{x}_1 := \mathbf{x}$, we compute $f_1(\mathbf{x}_1)$, then transform $\mathbf{x}_1 \mapsto \mathbf{x}_2 := L_2 \mathbf{x}_1 + \mathbf{c}_2$, where L_2 has three randomly chosen entries in each row, and $f_2(\mathbf{x}) := f_1(\mathbf{x}_2)$. Continue in this vein and do $\mathbf{x}_2 \mapsto \mathbf{x}_3 := L_3 \mathbf{x}_2 + \mathbf{c}_3$, $f_3(\mathbf{x}) := f_1(\mathbf{x}_3)$, and so on and so forth.

A set of quadratic polynomials like this we call "rotated". "Rotated" quadratics are obviously a kind of sparsely generated polynomials, and they behave like random ones under $\mathbf{F_4}$. *In Fig. 1 in the appendix, data points denoted "non-random sparse" polynomials are rotated.*

Assuming 160-bit hashes (SHA-1), preliminary runs with a composed rotated F:

– 40 GF(256) variables into 20: 1720 cycles/byte
– 80 GF(16) variables into 40: 3220 cycles/byte

Assuming 256-bit hashes (SHA-2), preliminary runs with a composed rotated F:

– 64 GF(256) variables into 32: 8100 cycles/byte
– 128 GF(16) variables into 64: 24100 cycles/byte

Again, we note that the transformation L has random constant terms. This leads to random affine parts in the structure throughout, to defend against the attack below.

7 Further Discussion: Variant Ideas and Other Attacks

We see that our hash schemes are roughly on a par speedwise with other schemes that depends on hard problems. It is true that there may be better formulations of the same idea, but $\mathcal{M}Q$ is a known hard problem, which should lend a measure of confidence.

7.1 Other Attacks

There are many specialized attacks in multivariate public key cryptography, especially the true multivariates, that one may think to use to attack our systems. But one should realize that due to the property of random polynomials, it is hard to imagine the usual attacks of linear and differential cryptanalysis working since cubic and quartic equations are so far removed from linearity. From what we can see, all but two related ones are now inapplicable to attack our hash.

These attacks are proposed by Aumasson and Meier [1]. The points are:

– If the affine part is as sparse as the rest of the polynomials, then there is a high probability to construct a collision or partial collision.
– There is a way to solve the differential of the hash if we use a cubic construction where the cubic terms are sparse, but this involves searching for a short vector in a code, and the exact time complexity is unknown.

Both of these ideas apply over GF(2) only, hence our use of GF(16) and larger. Of course, the specific points still bear more investigation.

7.2 Other Constructions

The key idea here is once we have a sparse construction, we would like to add some kind of internal perturbation to make system even more complicated. The key question is about how we add the perturbation. Another idea is to use a Feistel structure, which might speed up evaluation a lot with random maps, but requires more set-up and makes any putative pre-image resistance harder to show. Since the principle is not much different we don't go in that direction.

8 Conclusion

In this paper, we present the idea of using randomly polynomials, and randomly polynomials with sparse property to build hash functions. What is new here are: the cubic construction, the amplify-compress composed quadratic construction, and the specially construced systems using compositions of sparse linear functions.

We present arguments for the security of the system and for the case of sparse construction. We can improve our ideas with internal perturbation by adding additional noise into our sparse system. One more idea is to use composite field, where we explore further field structure to make our system more secure.

It is clear that some of these programming is very preliminary. The idea is mainly to point out a new direction in developing hash function whose security relies on a clear hard problems and therefore easier to study and understand, our work is just the beginning of this new direction and much work need to be done. We believe the multivariate hash has a very strong potential in practical applications. Much more work is needed in finding the right constructions and optimal parameters, and in rigorous proofs of security.

Acknowledgements

The authors would like to thank the Taiwan Information Security Center at Nat'l Taiwan U. of Sci. and Tech. location (NSC Project 95-2218-E-011-015) in 2006, the discussions following which led to this work.

In addition, BY and JD would like to thank the Taiwan's National Science Council and the Humboldt Foundation respectively for support (NSC 95-2115-M-001-021).

A version of this paper is also on http://eprint.iacr.org/2007/137.

References

1. Aumasson, J.-P., Meier, W.: Analysis of multivariate hash functions. In: Proc. ICISC. LNCS (to appear, 2007), cf.`http://www.131002.net/files/pub/AM07.pdf`
2. Bardet, M., Faugère, J.-C., Salvy, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proceedings of the International Conference on Polynomial System Solving, pp. 71–74 (2004); Previously INRIA report RR-5049
3. Berbain, C., Gilbert, H., Patarin, J.: QUAD: A Practical Stream Cipher with Provable Security. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 109–128. Springer, Heidelberg (2006)
4. Computational Algebra Group, University of Sydney. The MAGMA Computational Algebra System for Algebra, Number Theory and Geometry, `http://magma.maths.usyd.edu.au/magma/`
5. Courtois, N., Goubin, L., Meier, W., Tacier, J.-D.: Solving Underdefined Systems of Multivariate Quadratic Equations. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, Springer, Heidelberg (2002)

6. Ding, J., Gower, J., Schmidt, D.: Multivariate Public-Key Cryptosystems. In: Advances in Information Security, Springer, Heidelberg (2006)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability — A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
8. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, Springer, Heidelberg (1999), `http://www.minrank.org/hfesubreg.ps` or `http://citeseer.nj.nec.com/kipnis99cryptanalysis.html`
9. Menezes, A., Koblitz, N.: Another Look at "Provable Security". II. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 148–175. Springer, Heidelberg (2006)
10. Koblitz, N., Menezes, A.: Another look at "provable security". Journal of Cryptology 20, 3–37 (2004)
11. Raddum, H., Semaev, I.: New technique for solving sparse equation systems. Cryptology ePrint Archive, Report 2006/475 (2006), `http://eprint.iacr.org/`
12. Sugita, M., Kawazoe, M., Imai, H.: Gröbner basis based cryptanalysis of sha-1. Cryptology ePrint Archive, Report 2006/098(2006), `http://eprint.iacr.org/`
13. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
14. Wang, X., Yu, H.: How to break md5 and other hash functions. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
15. Wolf, C., Preneel, B.: Taxonomy of public key schemes based on the problem of multivariate quadratic equations. Cryptology ePrint Archive, Report 2005/077, 64 (May, 2005) `http://eprint.iacr.org/2005/077/`
16. Yang, B.-Y., Chen, J.-M.: All in the XL Family: Theory and Practice. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 67–86. Springer, Heidelberg (2005)
17. Yang, B.-Y., Chen, O.C.-H., Bernstein, D.J., Chen, J.-M.: Analysis of QUAD. In: Biryukov, A. (ed.) Fast Software Encryption — FSE 2007. volume to appear of Lecture Notes in Computer Science, pp. 302–319. Springer, Heidelberg (2007) workshop record available

A Testing

We depict in Fig. 1 some MAGMA-2.13.6 tests we ran, with $K = \mathrm{GF}(2)$, $2n$ variables and $3n$ equations. At $n = 16$ the system started to thrash due to lack of memory.

Despite the fact that we had a very good workstation with 64 GB of main memory, this thrashing is perfectly in line with theory. I.e., it should run out of memory when $n = 16$.

The tests we did included quadratic, cubic, and a few quartic systems with number of variable n, number of equations m, and $n : m$ being roughly equal to 1, 0.8, 0.75, 2/3, and 0.5. We also did systems over GF(2), GF(16), GF(256), and we went up for as long as the memory allowed.

- Systems with every coefficient randomly chosen.
- Systems with ϵ proportion of coefficients randomly picked to be non-zero, then randomly chosen from non-zero field elements. We tried ϵ being two percent, one percent, half a percent, and 1 mil (that is 1/1000).

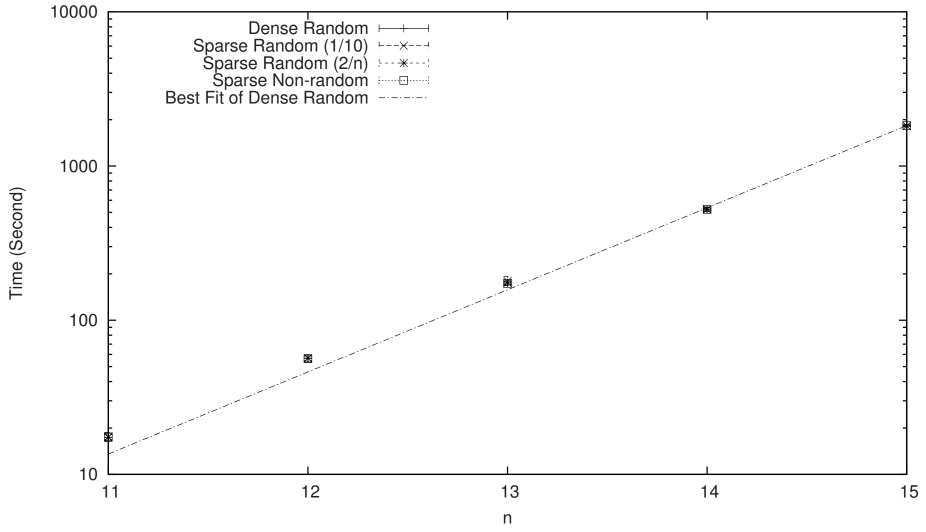


Fig. 1. "Sparsely determined random quadratics" in MAGMA

- We tried systems that have about n and $2n$ randomly picked non-zero non-linear terms in every equation (this is very sparse).
- Rotated sets (see Section 6.1) which had 3, 4, and 5 non-zero terms in every row (or every column, when we made a mistake in the program) of the transition matrices.

In every test using magma, the running time and memory used was close to each other which verifies the observation made by several people that the matrices become reasonably uniformly dense in $\mathbf{F_4}$ as we go up in degree.

Efficient Public Key Encryption with Keyword Search Schemes from Pairings[⋆]

Chunxiang Gu[1], Yuefei Zhu[1], and Heng Pan[2]

[1] Network Engineering Department, Zhengzhou Information Science and Technology Institute
P.O. Box 1001-770, Zhengzhou, 450002, P.R. China
gcxiang5209@yahoo.com.cn
[2] Zhongyuan university of technology, school of computer science

Abstract. Public key encryption with keyword search (PEKS) scheme to enable one to search encrypted keywords without compromising the security of the original data. In this paper, we propose a new PEKS scheme based on bilinear pairings. The scheme is computationally consistent. There is no pairing operation involved in the encryption procedure and so our new PEKS scheme is more efficient than the scheme of Boneh *et.al.* in [1]. Then, we provide further discussions on removing secure channel from PEKS, and present an efficient *secure channel free PEKS* scheme. Our two new schemes can be proved secure in the random oracle model, under the appropriate computational assumptions.

Keywords: public key encryption, keyword search, pairings, provable secure.

1 Introduction

In 2004, Boneh et.al [1] proposed the concept of *public key encryption with keyword search* (PEKS) scheme to enable one to search encrypted keywords without compromising the security of the original data. Later, Abdalla et.al [2] provided further discussions on the consistency properties and new constructions. They also suggest some new notions such as public-key encryption with temporary keyword search and identity based encryption with keyword search.

Suppose Bob wants to send Alice a message M with keywords $W_1, W_2, ..., W_n$. Let pk_A be Alice's public key. Bob encrypts M using a standard public key encryption $E(.)$. He then appends to the resulting ciphertext a list of PEKS ciphertext of each keyword. That is $E(M, pk_A)||PEKS(W_1, pk_A)||...||PEKS(W_n, pk_A)$. This kind of encrypted messages may be stored in a server. Alice can give the server a certain trapdoor T_W that enables the server to test whether one of the keywords associated with the message is equal to the word W of Alice's choice. Given $PEKS(W', pk_A)$ and T_W, the server can test whether $W = W'$. If $W \neq W'$ the server learns nothing more about W'.

[⋆] Research supported by NSFC (No.60473021), the Natural Science Fundation of Henan Province (No.072300410260, No.0511010900).

Such PEKS scheme can be widely used in many practical applications. For instance, Boneh et.al [1] explain that PEKS provides a mechanism that allows user Alice to have his email server extract encrypted emails that contain a particular keyword by providing a trapdoor corresponding to the keyword, while the email server and other parties excluding Alice do not learn anything else about the email. Shortly after Boneh et al.'s work, Waters et al. [3] showed that the PEKS scheme can be applied to build encrypted and searchable audit logs.

The schemes in [1,2] need secure channels to transmit trapdoors to the server. However, building a secure channel is usually expensive. Very recently, Baek et al. [4] discussed "removing secure channel", and provided a notion of *secure channel free public key encryption with keyword search* (SCF-PEKS) scheme.

In this paper, we propose a new PEKS scheme based on pairings. Its encryption procedure needs no pairing operation. So our scheme is more efficient than the schemes in [1,2]. Just as discussed in [2], our new scheme is computationally consistent and can be fine in practice. Then, we provide further discussions on the notion and security model for SCF-PEKS scheme, and present an efficient SCF-PEKS scheme. The new schemes can be proved secure in the random oracle model, under the appropriate computational assumptions.

The rest of this paper is organized as follows: In Section 2, we recall some preliminary works. In Section 3, we present a new PEKS scheme with efficiency discussions and security proof. In Section 4, we provide further discussions on the formal model for SCF-PEKS schemes, and present a new efficient SCF-PEKS scheme with provable security. Finally, we end the paper with a brief conclusion.

2 Preliminaries

2.1 Public Key Encryption with Keyword Search

Definition 1. *[1] A public key encryption with Keyword Search (PEKS) scheme consists of four polynomial-time algorithms:*

- ***KeyGen*****:** *Take as input a security parameter λ, generate a public/private key pair (pk, sk).*
- ***Trapdoor*****:** *Take as input the receiver's private key sk and a word W, produce a trapdoor T_W.*
- ***PEKS*****:** *Take as input the receiver's public key pk and a word W, produce a searchable encryption of W.*
- ***Test*****:** *Take as input the receiver's public key pk, a searchable encryption $C = PEKS(pk, W')$, and a trapdoor $T_W = Trapdoor(sk, W)$, output 1 ("yes") if $W = W'$ and 0 ("no") otherwise.*

For any keyword W and key pair $(pk, sk) = KeyGen(1^\lambda)$, $T_W = Trapdoor(sk, W)$, correctness requires that $Test(pk, PEKS(pk, W), T_W) = 1$. In [4], the authors provided further discussions on consistency and suggested the definitions of *perfectly consistent*, *statistically consistent* and *computationally consistent* according to the type of the adversaries.

In [1], Boneh et.al defined a security notion for PEKS schemes– "*indistinguishability of PEKS against chosen keyword attack*" (IND-CKA).

IND-CKA game

- **KeyGen:** The challenger runs the $KeyGen(\lambda)$ algorithm to generate (pk, sk). It gives pk to the attacker $\mathcal{A}$.
- **Phase 1:** The attacker $\mathcal{A}$ can adaptively ask the challenger for the trapdoor T_W for any keyword $W \in \{0,1\}^*$ of his choice.
- **Challenge:** At some point, the attacker $\mathcal{A}$ sends the challenger two words W_0, W_1 on which it wishes to be challenged. The only restriction is that the attacker did not previously ask for the trapdoors T_{W_0} or T_{W_1}. The challenger picks a random $b \in \{0,1\}$ and gives the attacker $C = PEKS(pk, W_b)$ as the challenge PEKS ciphertext.
- **Phase 2:** The attacker can continue to ask for trapdoors T_W for any keyword W of his choice as long as $W \neq W_0, W_1$.
- **Guess:** Eventually, the attacker $\mathcal{A}$ outputs $b' \in \{0,1\}$ and wins the game if $b = b'$.

Such an adversary $\mathcal{A}$ is called an IND-CKA adversary. $\mathcal{A}$'s advantage in attacking the scheme $\mathcal{E}$ is defined as the following function of the security parameter λ:

$$Adv_{\mathcal{E},\mathcal{A}}(\lambda) = |Pr[b = b'] - 1/2|.$$

The probability is over the random bits used by the challenger and the adversary.

Definition 2. *[1] A PEKS scheme $\mathcal{E}$ is IND-CKA secure if for any polynomially time adversary $\mathcal{A}$, $Adv_{\mathcal{E},\mathcal{A}}(\lambda)$ is negligible.*

2.2 Bilinear Pairings

Let $(G_1, +)$ and $(G_2, \cdot)$ be two cyclic groups of prime order q. $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be a map which satisfies the following properties.

1. Bilinear: $\forall P, Q \in G_1, \forall \alpha, \beta \in Z_q, \hat{e}(\alpha P, \beta Q) = \hat{e}(P, Q)^{\alpha\beta}$;
2. Non-degenerate: If P is a generator of G_1, then $\hat{e}(P, P)$ is a generator of G_2;
3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.

Such a bilinear map is called an *admissible bilinear pairing* [5]. The Weil pairings and the Tate pairings of elliptic curves can be used to construct efficient admissible bilinear pairings.

We review two complexity problems related to bilinear pairings: the Bilinear Diffie-Hellman (BDH) problem [5] and the Bilinear Diffie-Hellman Inverse (BDHI) problem [6,7]. Let P be a generator of G_1, and $a, b, c \in Z_q^*$.

- **BDH problem:** given $P, aP, bP, cP \in G_1$, output $\hat{e}(P, P)^{abc}$. An algorithm $\mathcal{A}$ solves BDH problem with the probability ε if

$$Pr[\mathcal{A}(P, aP, bP, cP) = \hat{e}(P, P)^{abc}] \geq \varepsilon,$$

where the probability is over the random choice of generator $P \in G_1^*$, the random choice of $a, b, c \in Z_q^*$ and random coins consumed by $\mathcal{A}$.

– k**-BDHI problem:** given $(P, aP, a^2P, ...a^kP) \in (G_1^*)^{k+1}$, output $\hat{e}(P,P)^{a^{-1}}$. An algorithm $\mathcal{A}$ solves k-BDHI problem with the probability ε if

$$Pr[\mathcal{A}(P, aP, a^2P, ...a^kP) = \hat{e}(P,P)^{a^{-1}}] \geq \varepsilon,$$

where the probability is over the random choice of generator $P \in G_1^*$, the random choice of $a \in Z_q^*$ and random coins consumed by $\mathcal{A}$.

We assume through this paper that BDH problem and k-BDHI problem are intractable, which means that there is no polynomial time algorithm to solve BDH problem or k-BDHI problem with non-negligible probability.

3 A New PEKS Scheme from Pairings

3.1 The Scheme

The method for obtaining trapdoors from keywords is a simplification of a method suggested by Sakai and Kasahara [12]. This leads to a more efficient performance. Let $(G_1, +)$ and $(G_2, \cdot)$ be two cyclic groups of prime order q, $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be an admissible bilinear pairing, $H_1 : \{0,1\}^* \rightarrow Z_q^*$ and $H_2 : G_2 \rightarrow \{0,1\}^{\log q}$ be two hash functions. P is a generator of G_1, $\mu = \hat{e}(P,P)$. The scheme is described as following:

– ***KeyGen*:** Pick a random $x \in Z_q^*$, compute $X = xP$, and output $pk = X$, and $sk = x$.
– ***Trapdoor*:** Take as input secret key x and keyword W, and output $T_W = (H_1(W) + x)^{-1}P$.
– ***PEKS*:** Take as input public key X and a keyword W, select randomly $r \in Z_q^*$, compute $U = rH_1(W)P + rX$, $c = H_2(\mu^r)$ and output (U, c).
– ***Test*:** For input public key X, searchable encryption cipher-text (U, c) and trapdoor T_W, test if $H_2(\hat{e}(T_W, U)) = c$. If so, output 1; otherwise, output 0.

3.2 Consistency and Efficiency

Correctness of the scheme is easily proved as follows:

$$\begin{aligned} H_2(\hat{e}(T_W, U)) &= H_2(\hat{e}((H_1(W) + x)^{-1}P, rH_1(W)P + rX)) \\ &= H_2(\hat{e}((H_1(W) + x)^{-1}P, r(H_1(W) + x)P)) \\ &= H_2(\hat{e}(P,P)^r) = c. \end{aligned}$$

With the same techniques used in the proof of Theorem 3.3 in [2], we can prove that our new scheme is computationally consistent. We do not repeat the details here.

Denote by M an ordinary scalar multiplication in $(G_1, +)$, by E an Exp. operation in $(G_2, .)$, and by $\hat{e}$ a computation of the pairing. The hash function

$H_1 : \{0,1\}^* \rightarrow G_1^*$ used by the schemes in [1,2] usually requires a "Maptopoint operation" [5] to map a keyword to an element in G_1. As discussed in [5], Maptopoint operation (denoted by P) is so inefficient that we can't neglect it. Do not take other operations into account. We compare our scheme to the schemes in [1,2] in the following table.

schemes	KeyGen	Trapdoor	PEKS	Test
scheme in [1]	$1M$	$1M + 1P$	$2M + 1P + 1\hat{e}$	$1\hat{e}$
scheme in [2]	$1M$	$1M + 1P$	$1E + 1P + 1\hat{e}$	$1\hat{e}$
proposed	$1M$	$1M$	$2M + 1E$	$1\hat{e}$

Note: The hash function used in our scheme which maps a keyword to an element in Z_q^* is so efficient that we usually can neglect it.

Although fruitful achievements [9,10] have been made in enhancing the computation of pairings, the computation of pairings is still the most time-consuming operations. Our new scheme requires no pairing operation in PEKS procedure. So our new PEKS scheme is more efficient comparatively in performance.

Some general performance enhancements can also be applied to our scheme. For pre-selected $P \in G_1$ and $\mu \in G_2$, there are efficient algorithms [11] to compute $rH_1(ID_X)P$ and μ^r for a random $r \in Z_q^*$ by pre-computing and storing.

3.3 Security Proof

Theorem 1. *Let $\mathcal{F}_0$ be an IND-CKA adversary that has advantage $\varepsilon(\lambda)$ within a time bound $T(\lambda)$. Suppose $\mathcal{F}_0$ makes at most $q_T > 0$* ***Trapdoor*** *queries, $q_1 > 0$ hash function queries to H_1 and $q_2 > 0$ hash function queries to H_2. Let $n = max\{q_1, 2q_T\}$. Then there is an algorithm $\mathcal{F}_1$ that solves the n-BDHI problem with advantage at least $\varepsilon(\lambda)/(nq_2)$ with a running time $\mathcal{O}(T(\lambda))$.*

Proof: $\mathcal{F}_1$ is given input parameters of pairing $(q, G_1, G_2, \hat{e})$ and a random instance $(P, aP, a^2P, ..., a^nP)$ of the n-BDHI problem, where P is random in G_1^* and a is a random in Z_q^*. $\mathcal{F}_1$ simulates the challenger and interacts with $\mathcal{F}_0$ as follows:

- **KeyGen:** 1. Randomly choose different $h_0, h_1, ...h_{n-1} \in Z_q^*$, and compute $f(x) = \prod_{i=1}^{n-1}(x + h_i) = \sum_{i=0}^{n-1} c_i x^i$.
  2. Compute $Q = \sum_{i=0}^{n-1} c_i a^i P = f(a)P$, $aQ = \sum_{i=0}^{n-1} c_i a^{i+1} P$, and $Q' = \sum_{i=1}^{n-1} c_i a^{i-1} P$. In the (unlikely) situation where $Q = 1_{G_1}$, there exists an $h_i = -a$, hence, $\mathcal{F}_1$ can solve the n-BDHI problem directly and abort.
  3.Compute $f_i(x) = f(x)/(x + h_i) = \sum_{j=0}^{n-2} d_j x^j$. Obviously, $(a + h_i)^{-1}Q = (a + h_i)^{-1} f(a)P = f_i(a)P = \sum_{j=0}^{n-2} d_j a^j P$ for $1 \le i \le n$.
  4. Randomly choose an index t with $1 \le t \le n$, set $v = 0$, and start by giving $\mathcal{F}_0$ the public key $Y = aQ - h_0Q$.
- **Phase 1: H_1-queries.** $\mathcal{F}_1$ maintains a H_1_list, initially empty. For a query W, if W already appears on the H_1_list in a tuple (W, g, D), $\mathcal{F}_1$ responds with g. Otherwise, sets $v = v + +$, $W_v = W$, if $v = t$, $\mathcal{F}_1$ sets $g_v = h_0$,

$D_v = \perp$ ($\perp$ means null); otherwise, $\mathcal{F}_1$ selects a random $n \geq \iota > 0$ which has not been chosen and sets $g_v = h_\iota + h_0$, $D_v = (a + h_\iota)^{-1}Q$. In both case, adds the tuple (W_v, g_v, D_v) to H_1_list and responds with g_v.

– **Phase 1: H_2-queries.** $\mathcal{F}_1$ maintains a H_2_list, initially empty. For a query e_i, $\mathcal{F}_1$ checks if e_i appears on the H_2_list in a tuple (e_i, u_i). If not, $\mathcal{F}_1$ picks a random $u_i \in \{0,1\}^{\log q}$, and adds the tuple $(e_i, u_i)$to the H_2_list. $\mathcal{F}_1$ returns u_i to $\mathcal{F}_0$.

– **Phase 1: *Trapdoor* queries:** For input W_i, without any loss of generality, we can assume that W_i has already been asked to oracle H_1. $\mathcal{F}_1$ searches in H_1_list for (W_i, g_i, D_i). If $D_i = \perp$ then $\mathcal{F}_1$ aborts. Otherwise, $\mathcal{F}_1$ responds with D_i.

– **Challenge:** Once $\mathcal{F}_0$ decides that Phase 1 is over it outputs two keywords W'_0, W'_1 on which it wishes to be challenged. $\mathcal{F}_1$ responds as follows:
1. $\mathcal{F}_1$ runs the above algorithm for responding to H_1-queries twice to obtain (W'_0, g'_0, D'_0) and (W'_1, g'_1, D'_1). If both $D'_0 \neq \perp$ and $D'_1 \neq \perp$ then $\mathcal{F}_1$ aborts. Otherwise, $\mathcal{F}_1$ responds with the challenge ciphertext (bQ, ξ) for random selected $b \in Z_q^*$ and $\xi \in \{0,1\}^{\log q}$. (Observe that if (bQ, ξ) is a ciphertext corresponding to W'_ι with $\iota \in \{0,1\}$ satisfying $D'_\iota = \perp$, by definition, the decryption of C is $\xi = H_2(\hat{e}(T_{W'_\iota}, bQ)) = H_2(\hat{e}(a^{-1}Q, bQ)) = H_2(\hat{e}(Q,Q)^{a^{-1}b})$.)

– **Phase 2: H_1-queries, H_2-queries, *Trapdoor* queries.** $\mathcal{F}_1$ responds to these queries in the same way it does in Phase 1 with the only restriction that $W_i \neq W'_0, W'_1$ for *Trapdoor* queries.

– **Guess:** Eventually $\mathcal{F}_0$ produces its guess $\iota' \in \{0,1\}$ for ι.

$\mathcal{F}_1$ keeps interacting with $\mathcal{F}_0$ until $\mathcal{F}_0$ halts or aborts. If $\mathcal{F}_0$ produces a guess ι', $\mathcal{F}_1$ picks a random tuple (e_i, u_i) from the H_2_list. $\mathcal{F}_1$ computes $\alpha = e_i^{b^{-1}}$, $\beta = \hat{e}(Q', Q + c_0 P)$ and outputs $(\alpha/\beta)^{c_0^{-2}}$ as the solution to the given instance of n-BDHI problem. (Note that if $\alpha = \hat{e}(Q,Q)^{a^{-1}}$, then $(\alpha/\beta)^{c_0^{-2}} = \hat{e}(P,P)^{a^{-1}}$.)

This completes the description of $\mathcal{F}_1$.

Suppose that in a real attack game $\mathcal{F}_0$ is given the public key $(Q, Y = aQ - h_0 Q)$ and $\mathcal{F}_0$ asks to be challenged on words W'_0 and W'_1. In response, $\mathcal{F}_0$ is given a challenge (bQ, ξ). Then, just as discussed in [1], in the real attack game $\mathcal{F}_0$ issues an H_2 query for either $H_2(\hat{e}(T_{W'_0}, bQ))$ or $H_2(\hat{e}(T_{W'_1}, bQ))$ with probability at least $2\varepsilon(\lambda)$.

Now, assuming $\mathcal{F}_1$ does not abort, we know that $\mathcal{F}_1$ simulates a real attack game perfectly up to the moment when $\mathcal{F}_0$ issues a query for either $H_2(\hat{e}(T_{W'_0}, bQ))$ or $H_2(\hat{e}(T_{W'_1}, bQ))$. Therefore, the value $H_2(\hat{e}(T_{W'_\iota}, bQ)) = H_2(\hat{e}(Q,Q)^{a^{-1}b})$ will appear in the H_2-list with probability at least $\varepsilon(\lambda)$. $\mathcal{F}_1$ will choose the correct pair with probability at least $1/q_2$.

During the simulation, the probability that $\mathcal{F}_1$ does not abort in phases 1 or 2 because of $\mathcal{F}_0$'s ***Trapdoor*** queries is $1 - q_T/n$. The probability that $\mathcal{F}_1$ does not abort during the challenge step is $2/n$. Because $n \geq 2q_T$, we know that the probability that $\mathcal{F}_1$ does not abort during the simulation is $(1 - q_T/n)2/n \geq 1/n$.

Therefore, $\mathcal{F}_1$'s success probability overall is at least $\varepsilon(\lambda)/(nq_2)$.

3.4 Public-Key Encryption with Temporary Keyword Search

Both [2] and [4] discussed the server's attack by storing trapdoors. That is, once the gateway has the trapdoor for a certain period, it can test whether this keyword was present in past ciphertexts, and can test its presence in any future ciphertexts. The possible solution is by using *public-key encryption with temporary keyword search* [2]. Abdalla et.al [2] provided a construction from hierarchical encryption. Baek et.al [4] suggested a more efficient solution by *refreshing keywords* (appending the time period to the keyword when encrypting or computing trapdoors). This skill can also be applied to our scheme.

4 PEKS Schemes without Secure Channel

PEKS schemes need secure (encrypted and authenticated) channels between users and servers. However, building a secure channel is usually expensive. In [4], Baek et.al suggested a formal model for *secure channel free public key encryption with keyword search* (SCF-PEKS) scheme, which defines SCF-PEKS scheme with six algorithms. In this section, we provide further discussions on the formal model for SCF-PEKS schemes, and present a new efficient SCF-PEKS scheme with provable security.

4.1 New Formal Model for SCF-PEKS Schemes

A SCF-PEKS scheme enables the sender to use the server's public key as well as the receiver's public key to generate PEKS ciphertexts. The receiver then can send a trapdoor to retrieve data associated with the encrypted keyword via a public channel.

Definition 3. *A SCF-PEKS scheme consists of four polynomial-time algorithms:*

- ***KeyGen*:** *Take as input a security parameter λ, generate a public/private key pair (pk, sk). This algorithm is used to generate key pairs for users (including the receiver and the server).*
- ***Trapdoor*:** *Take as input the receiver's private key sk_r and a word W, produce a trapdoor T_W.*
- ***PEKS*:** *Take as input the receiver's public key pk_r, the server's public key pk_s and a word W, produce a searchable encryption of W.*
- ***Test*:** *Take as input the server's secret key sk_s and the receiver's public key pk_r, a searchable encryption $S = PEKS(pk_r, pk_s, W')$, and a trapdoor $T_W = Trapdoor(sk_r, W)$, output 1 ("yes") if $W = W'$ and 0 ("no") otherwise.*

Consistency requires that for any keyword W, receiver's key pair $(pk_r, sk_r) = KeyGen(1^\lambda)$, server's key pair $(pk_s, sk_s) = KeyGen(1^\lambda)$, $T_W = Trapdoor(sk_r, W)$, we have $Test(sk_s, pk_r, PEKS(pk_r, pk_s, W), T_W) = 1$.

As to security, informally, we can say a SCF-PEKS scheme is secure if it can achieve the following goals:

- The attacker without the trapdoors for given keywords cannot tell the PEKS ciphertext is produced from which keyword, even if he knows the server's secret key. We call this security property "*indistinguishability against chosen keyword attack with server's secret key*" (IND-CKA-SSK).
- The attacker without the server's private key cannot make any decisions about the PEKS ciphertexts even though the attacker gets all the trapdoors for the keywords that it holds. We call this security property "*indistinguishability against chosen keyword attack with all trapdoors*" (IND-CKA-AT).

Formally, we define the following two security notions.

IND-CKA-SSK game

- **KeyGen:** The challenger runs the $KeyGen(\lambda)$ algorithm twice to generate the server's key pair (pk_s, sk_s) and the receiver's key pair (pk_r, sk_r). It gives pk_s, pk_r, sk_s to the attacker.
- **Phase 1**, **Challenge**, **Phase 2**, **Guess:** The attacker $\mathcal{A}$ does these steps almost the same as that in IND-CKA game, except that the challenge ciphertext is $C = PEKS(pk_r, pk_s, W_b)$, where $b \in_R \{0,1\}$, W_0, W_1 are the two words to be challenged.

The adversary $\mathcal{A}$ is called an IND-CKA-SSK adversary. $\mathcal{A}$'s advantage is defined as:

$$Adv_{\mathcal{E},\mathcal{A}}^{IND-CKA-SSK}(\lambda) = |Pr[b = b'] - 1/2|.$$

The probability is over the random bits used by the challenger and the adversary.

Definition 4. *A SCF-PEKS scheme $\mathcal{E}$ is IND-CKA-SSK secure if for any polynomially time adversary $\mathcal{A}$, $Adv_{\mathcal{E},\mathcal{A}}^{IND-CKA-SSK}(\lambda)$ is negligible.*

IND-CKA-AT game

- **KeyGen:** The challenger runs the $KeyGen(\lambda)$ algorithm twice to generate the server's key pair (pk_s, sk_s) and the receiver's key pair (pk_r, sk_r). It gives pk_s, pk_r to the attacker.
- **Phase 1:** The attacker can adaptively ask the challenger for the trapdoor T_W for any keyword $W \in \{0,1\}^*$ of his choice.
- **Challenge:** At some point, the attacker $\mathcal{A}$ sends the challenger two words W_0, W_1 on which it wishes to be challenged. The challenger picks a random $b \in \{0,1\}$ and gives the attacker $C = PEKS(pk_r, pk_s, W_b)$ as the challenge PEKS.
- **Phase 2:** The attacker can continue to ask for trapdoors T_W for any keyword W of his choice.
- **Guess:** Eventually, the attacker $\mathcal{A}$ outputs $b' \in \{0,1\}$ and wins the game if $b = b'$.

The adversary $\mathcal{A}$ is called an IND-CKA-AT adversary. $\mathcal{A}$'s advantage is defined as:

$$Adv_{\mathcal{E},\mathcal{A}}^{IND-CKA-AT}(\lambda) = |Pr[b = b'] - 1/2|.$$

The probability is over the random bits used by the challenger and the adversary.

Definition 5. *A SCF-PEKS scheme $\mathcal{E}$ is IND-CKA-AT secure if for any polynomially time adversary $\mathcal{A}$, $Adv_{\mathcal{E},\mathcal{A}}^{IND-CKA-AT}(\lambda)$ is negligible.*

4.2 A New SCF-PEKS Scheme from Pairings

Let $(G_1, +)$ and $(G_2, \cdot)$ be two cyclic groups of prime order q, $\hat{e} : G_1 \times G_1 \rightarrow G_2$ be an admissible bilinear pairing, $H_1 : \{0,1\}^* \rightarrow Z_q^*$ and $H_2 : G_2 \rightarrow \{0,1\}^{\log q}$ be two hash functions. P is a generator of G_1, $\mu = \hat{e}(P, P)$. The scheme is described as following:

- ***KeyGen*:** Pick a random $x \in Z_q^*$, compute $X = xP$, and output $pk = X$, and $sk = x$.
- ***Trapdoor*:** Take as input secret key x and keyword W, output $T_W = (H_1(W) + x)^{-1}P$.
- ***PEKS*:** Take as input a receiver's public key X, a server's public key Y and a keyword W, select randomly $r_1, r_2 \in Z_q^*$, compute $U = r_1 H_1(W)P + r_1 X$, $V = r_2 P$, $c = H_2(\hat{e}(r_1 P + r_2 U, Y))$ and output (U, V, c).
- ***Test*:** Take as input the receiver's public key X, the server's private key $y \in Z_q^*$, a searchable encryption cipher-text (U, V, c) and trapdoor T_W, test if $H_2(\hat{e}(yU, T_W + V)) = c$. If so, output "yes"; otherwise, output "no".

4.3 Consistency and Efficiency

Consistency of the scheme is easily proved as follows:

$$\begin{aligned} H_2(\hat{e}(yU, T_W + V)) &= H_2(\hat{e}(U, (H_1(W) + x)^{-1}P + r_2 P)^y) \\ &= H_2(\hat{e}(r_1(H_1(W) + x)P, (H_1(W) + x)^{-1}P)^y \cdot \hat{e}(U, r_2 P)^y) \\ &= H_2(\hat{e}(r_1 P, yP) \cdot \hat{e}(r_2 U, yP)) \\ &= H_2(\hat{e}(r_1 P + r_2 U, Y)) = c. \end{aligned}$$

Denote by M an ordinary scalar multiplication in $(G_1, +)$, by E an Exp. operation in $(G_2, .)$, by $\hat{e}$ a computation of the pairing and by P a Maptopoint operation [5]. Do not take other operations into account. We compare our scheme to the scheme in [4] in the following table.

schemes	KeyGen	Trapdoor	PEKS	Test
scheme in [4]	$1M$	$1M + 1P$	$1M + 1P + 1E + 2\hat{e}$	$2M + 1\hat{e}$
proposed	$1M$	$1M$	$5M + 1\hat{e}$	$1M + 1\hat{e}$

4.4 Security Proof

Theorem 2. *Let $\mathcal{F}_0$ be an IND-CKA-SSK adversary that has advantage $\varepsilon(\lambda)$ within a time bound $T(\lambda)$. Suppose $\mathcal{F}_0$ makes at most $q_T > 0$ **Trapdoor** queries, $q_1 > 0$ hash function queries to H_1 and $q_2 > 0$ hash function queries to H_2. Let $n = max\{q_1, 2q_T\}$. Then there is an algorithm $\mathcal{F}_1$ that solves the n-BDHI problem with advantage at least $\varepsilon(\lambda)/(nq_2)$ with a running time $\mathcal{O}(T(\lambda))$.*

Proof: $\mathcal{F}_1$ is given input parameters of pairing $(q, G_1, G_2, \hat{e})$ and a random instance $(P, aP, a^2P, ..., a^nP)$ of the n-BDHI problem, where P is random in G_1^* and a is a random in Z_q^*. $\mathcal{F}_1$ simulates the challenger and interacts with $\mathcal{F}_0$ as follows:

– **KeyGen:** Randomly choose different $h_0, h_1, ...h_{n-1} \in Z_q^*$, and compute $f(x)$, Q, aQ, Q', $(a + h_i)^{-1}Q$ for $1 \leq i \leq n$ the same as that in the proof of Theorem 1. In the (unlikely) situation where $Q = 1_{G_1}$, there exists an $h_i = -a$, hence, $\mathcal{F}_1$ can solve the q_1-BDHI problem directly and abort.
  2. Randomly choose an index t with $1 \leq t \leq n$, sets $v = 0$. Select a random $y \in Z_q^*$ and start by giving $\mathcal{F}_0$ the reciver's public key $X = aQ - h_0Q$ and the server's key pair (y, yQ).
– **Phase 1: H_1-queries, H_2-queries, *Trapdoor* queries.** $\mathcal{F}_1$ responds to these queries the same way as that in the proof of Lemma 1.
– **Challenge:** Once $\mathcal{F}_0$ decides that Phase 1 is over it outputs two keywords W_0', W_1' on which it wishes to be challenged. $\mathcal{F}_1$ responds as follows:
  1. $\mathcal{F}_1$ runs the above algorithm for responding to H_1-queries twice to obtain (W_0', g_0', D_0') and (W_1', g_1', D_1'). If both $D_0' \neq \perp$ and $D_1' \neq \perp$ then $\mathcal{F}_1$ aborts. Otherwise, $\mathcal{F}_1$ responds to the challenge with cipher-text $(\gamma_1 Q, \gamma_2 Q, \xi)$ for randomly selected $\gamma_1, \gamma_2 \in Z_q^*$ and $\xi \in \{0,1\}^{\log q}$. (Observe that if $(\gamma_1 Q, \gamma_2 Q, \xi)$ is a cipher-text corresponding to W_ι' with $\iota \in \{0, 1\}$ satisfying $D_\iota' = \perp$, by definition, the decryption of C is $\xi = H_2(\hat{e}(\gamma_1 Q, T_{W_\iota'} + \gamma_2 Q)^y) = H_2(\hat{e}(\gamma_1 Q, a^{-1}Q + \gamma_2 Q)^y) = H_2(\hat{e}(Q,Q)^{\gamma_1(a^{-1}+\gamma_2)y})$.)
– **Phase 2: H_1-queries, H_2-queries, *Trapdoor* queries.** $\mathcal{F}_1$ responds to these queries in the same way as it does in Phase 1 with the only restriction that $W_i \neq W_0', W_1'$ for *Trapdoor* queries.
– **Guess:** Eventually $\mathcal{F}_0$ produces its guess $\iota' \in \{0, 1\}$ for ι.

$\mathcal{F}_1$ keeps interacting with $\mathcal{F}_0$ until $\mathcal{F}_0$ halts or aborts. If $\mathcal{F}_0$ produces a guess ι', $\mathcal{F}_1$ picks a random tuple (e_i, h_i) from the H_2_list and computes $\delta = \hat{e}(Q, \gamma_2 Q)$, $\alpha = e_i^{(\gamma_1 y)^{-1}}/\delta$, $\beta = \hat{e}(Q', Q + c_0 P)$ and outputs $(\alpha/\beta)^{c_0^{-2}}$ as the solution to the given instance of n-BDHI problem. (Note that if $e_i = \hat{e}(Q,Q)^{\gamma_1(a^{-1}+\gamma_2)y}$, then $\alpha = \hat{e}(Q,Q)^{a^{-1}}$, hence, $(\alpha/\beta)^{c_0^{-2}} = \hat{e}(P,P)^{a^{-1}}$.)

This completes the description of $\mathcal{F}_1$. Just as discussed in the proof of Theorem 1, $\mathcal{F}_1$'s success probability overall is at least $\varepsilon(\lambda)/(nq_2)$.

Theorem 3. *Let $\mathcal{F}_0$ be an IND-CKA-AT adversary that has advantage $\varepsilon(\lambda)$ within a time bound $T(\lambda)$. Suppose $\mathcal{F}_0$ makes at most $q_T > 0$ **Trapdoor** queries, $q_1 > 0$ hash function queries to H_1 and $q_2 > 0$ hash function queries to H_2.*

Then there is an algorithm $\mathcal{F}_1$ that solves the BDH problem with advantage at least $2\varepsilon(\lambda)/q_2$ with a running time $\mathcal{O}(T(\lambda))$.

Proof: $\mathcal{F}_1$ is given input parameters of pairing $(q, G_1, G_2, \hat{e})$ and a random instance (P, aP, bP, cP) of the BDH problem, where P is random in G_1^* and a, b, c are random elements in Z_q^*. $\mathcal{F}_1$ simulates the challenger and interacts with $\mathcal{F}_0$ as follows:

– **KeyGen:** Select randomly $x \in Z_q^*$ and start by giving $\mathcal{F}_0$ the reciver's public key $X = xP$ and the server's public key aP.
– **Phase 1: H_1-queries.** $\mathcal{F}_1$ maintains a H_1_list, initially empty. For a query W_i, if W_i already appears on the H_1_list in a tuple (W_i, g_i), $\mathcal{F}_1$ responds with g_i. Otherwise, $\mathcal{F}_1$ selects a random $g_i \in Z_q^*$, adds the tuple (W_i, g_i) to the H_1_list and responds with g_i.
– **Phase 1: H$_2$-queries.** $\mathcal{F}_1$ maintains a H_2_list, initially empty. For a query e_i, $\mathcal{F}_1$ checks if e_i appears on the H_2_list in a tuple (e_i, h_i). If not, $\mathcal{F}_1$ picks a random $h_i \in \{0,1\}^{\log q}$, and adds the tuple $(e_i, h_i)$to the H_2_list. $\mathcal{F}_1$ returns h_i to $\mathcal{F}_0$.
– **Phase 1: *Trapdoor* queries:** For input W_i, without any loss of generality, we can assume that W_i has already been asked to oracle H_1. $\mathcal{F}_1$ searches in H_1_list for (W_i, g_i) and $\mathcal{F}_1$ responds with $D_i = (g_i + x)^{-1}P$.
– **Challenge:** Once $\mathcal{F}_0$ decides that Phase 1 is over it outputs two keywords W_0', W_1' on which it wishes to be challenged. $\mathcal{F}_1$ runs the above algorithm for responding to H_1-queries twice to obtain (W_0', g_0') and (W_1', g_1'). Selects $\iota \in \{0,1\}$ and responds with the challenge ciphertext $(g_\iota' bP + xbP, cP, \xi)$ for random selected $\xi \in \{0,1\}^{\log q}$. (Observe that if $(g_\iota' bP + xbP, cP, \xi)$ is a cipher-text corresponding to W_ι', by definition, the test procedure of C is to test $\xi = H_2(\hat{e}(g_\iota' bP + xbP, (g_\iota' + x)^{-1}P + cP)^a)$.)
– **Phase 2: H_1-queries, H_2-queries, *Trapdoor* queries.** $\mathcal{F}_1$ responds to these queries in the same way it does in Phase 1.
– **Guess:** Eventually $\mathcal{F}_0$ produces its guess $\iota' \in \{0,1\}$ for ι.

$\mathcal{F}_1$ keeps interacting with $\mathcal{F}_0$ until $\mathcal{F}_0$ halts or aborts. If $\mathcal{F}_0$ produces a guess ι', $\mathcal{F}_1$ picks a random tuple (e_i, h_i) from the H_2_list. $\mathcal{F}_1$ computes and outputs $\alpha = (e_i/\hat{e}(aP, bP))^{(g_\iota'+x)^{-1}}$ as the solution to the given instance of BDH problem. (Note that if $e_i = \hat{e}(g_\iota' bP + xbP, (g_\iota' + x)^{-1}P + cP)^a$, then $e_i = \hat{e}(bP, P)^a \hat{e}(P, P)^{bca(g_\iota'+x)}$, hence $\alpha = (e_i/\hat{e}(aP, bP))^{(g_\iota'+x)^{-1}} = \hat{e}(P, P)^{abc}$.)

This completes the description of $\mathcal{F}_1$.

We know that in the real attack game $\mathcal{F}_0$ issues an H_2 query for $H_2(\hat{e}(g_0' bP + xbP, (g_0' + x)^{-1}P + cP)^a)$ $H_2(\hat{e}(g_1' bP + xbP, (g_1' + x)^{-1}P + cP)^a)$ with probability at least $2\varepsilon(\lambda)$. $\mathcal{F}_1$simulates a real attack game perfectly up to the moment when $\mathcal{F}_0$ issues a query for $H_2(\hat{e}(g_\iota' bP + xbP, (g_\iota' + x)^{-1}P + cP)^a)$ with $\iota \in \{0,1\}$. Therefore, the value $\hat{e}(g_\iota' bP + xbP, (g_\iota' + x)^{-1}P + cP)^a$ will appear in the H_2-list with probability at least $2\varepsilon(\lambda)$. $\mathcal{F}_1$ will choose the correct pair with probability at least $1/q_2$. Therefore, $\mathcal{F}_1$'s success probability overall is at least $2\varepsilon(\lambda)/q_2$.

5 Conclusion

In this paper, first, we propose a new efficient PEKS scheme based on pairings and prove its security in the random oracle model under the hardness assumption of n-BDHI problem. Then, we provide further discussions on the notion of SCF-PEKS scheme, give a formal security model and present an efficient SCF-PEKS scheme. The security of the new SCF-PEKS scheme can be reduced to the n-BDHI problem and the BDH problem, in the random oracle model.

References

1. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
2. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
3. Waters, B., Balfanz, D., Durfee, G., Smetters, D.: Building an Encrypted and Searchable Audit Log. In: Network and Distributed System Security Symposium NDSS (2004)
4. Baek, J., Safiavi-Naini, R., Susilo, W.: Public Key Encryption with Keyword Search Revisited. Available on Cryptology ePrint Archive, Report 2005/119.
5. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, Springer, Heidelberg (2001)
6. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
7. Zhang, F., Safavi-Naini, R., Susilo, W.: An Efficient Signature Scheme from Bilinear Pairings and Its Applications. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 277–290. Springer, Heidelberg (2004)
8. Sakai, R., Kasahara, M.: ID based Cryptosystems with Pairing on Elliptic Curve. Cryptology ePrint Archive: Report 2003/054
9. Barreto, P., Kim, H., Lynn, B., Scott, M.: Efficient Algorithms for Pairing-Based Cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, Springer, Heidelberg (2002)
10. Duursma, I., Lee, H.: Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p + x + d$. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 111–123. Springer, Heidelberg (2003)
11. Sakai, Y., Sakurai, K.: Efficient Scalar Multiplications on Elliptic Curves without Repeated Doublings and Their Practical Performance. In: Clark, A., Boyd, C., Dawson, E.P. (eds.) ACISP 2000. LNCS, vol. 1841, pp. 59–73. Springer, Heidelberg (2000)
12. Sakai, R., Kasahara, M.: ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054
13. Dolev, D., Dwork, C., Naor, M.: Non-Malleable Cryptography. SIAM Journal on Computing, 2000. Early version in proceedings of STOC 1991 (2000)

Multi-Identity Single-Key Decryption without Random Oracles*

Fuchun Guo[1], Yi Mu[2], Zhide Chen[1], and Li Xu[1]

[1] Key Lab of Network Security and Cryptology
School of Mathematics and Computer Science
Fujian Normal University, Fuzhou, China
fuchunguo1982@gmail.com,
{zhidechen,xuli}@fjnu.edu.cn

[2] School of Computer Science and Software Engineering
University of Wollongong, Wollongong NSW 2522, Australia
ymu@uow.edu.au

Abstract. Multi-Identity Single-Key Decryption (MISKD) is an Identity-Based Encryption (IBE) system where a private decryption key can map multiple public keys (identities). More exactly, in MISKD, a single private key can be used to decrypt multiple ciphertexts encrypted with different public keys associated to the private key. MISKD is a variant of IBE and offers convenience to users who have to manage many private keys in a standard IBE. The notion of MISKD was recently introduced by Guo, Mu and Chen in Pairing 2007. They proposed a concrete MISKD scheme and proved its security based on the Bilinear Strong Diffie-Hellman problem (q-BSDH) in random oracle model. In this paper, we present a novel MISKD scheme that is provably secure in the selective-ID model based on the Decisional Bilinear Diffie-Hellman (DBDH) assumption. Our scheme is more efficient in decryption.

Keywords: ID-based Systems, Encryption, Pairing.

1 Introduction

In 1984, Shamir [16] introduced the notion of identity-based (or ID-based) cryptography and constructed an identity-based signature scheme. The motivation is to simplify certificate management in email systems. In identity-based cryptography, a public key can be an arbitrary string such as an email address, a user name, or an IP number. The beauty of identity-based cryptography lies in the convenience of public key handling, in the sense that a user's identity can serve as a public key without the need of a traditional Public Key Infrastructure (PKI). In 2001, Boneh and Franklin [4] for the first time successfully constructed a concrete identity-based encryption (IBE) scheme secure in the random oracle model

* Supported by National Natural Science Foundation of China (#60502047), Science and Technology of Fujian Province (2006F5036) and Open Funds of Key Lab of Fujian Province University Network Security and Cryptology (07B001).

against chosen ciphertext attack (IND-ID-CCA). Several novel IBE schemes have been proposed in various security models (e.g., [8,2,18,11]).

The notion of Multi-Identity Single Key Decryption (MISKD) was recently introduced in Pairing 2007 by Guo, Mu, and Chen [12]. We refer it to as GMC scheme. They considered a situation that a user works as a consultant for multiple (n) companies, assuming that an IBE scheme is adopted by these companies. Each company could provide him with an email address (or identity); therefore, in a traditional IBE system, he has to maintain n private keys that correspond to his n identities (apparently, no company will send an email to A's email address but using B's email address as the public key). This obviously is problematic to him when n is large. It would be nice, if he can use a single private key only and this magic private key can decrypt all ciphertexts encrypted with any of his public keys (identities). Fortunately, this has been proved feasible by Guo, Mu, and Chen [12] who proposed a MISKD scheme showing how to accumulate all private keys into a single one that can decrypt a ciphertext encrypted with any of associated public keys. Their scheme is based on the Bilinear Strong Diffie-Hellman assumption (q-BSDH for short), which is a variant of the q-SDH assumption, where q is the maximal number of public keys that a private key can map.

Although the GMC scheme is sound, there are two open problems to be solved. First, how to construct a MISKD scheme without random oracles? As we know, random oracle models are not desirable in secure proofs [7]. Second, the q-BSDH assumption is stronger than Decisional Bilinear Diffie-Hellman assumption and Cheon [6] has pointed out that q-SDH and other similar assumptions have weakness when q is the maximum number of private key owners that an adversary may corrupt in an active attack. Thus, how to construct a MISKD with a weaker assumption remains open. Moreover, the computation in their scheme is not efficient.

We should differentiate MISKD from Hierarchical Identity-Based Encryption (HIBE) and Fuzzy Identity-Based Encryption (FIBE).

The notion of hierarchical identities was introduced by Horwitz and Lynn [14]. The notion of Hierarchical Identity-Based Encryption (HIBE) was introduced by Gentry and Silverberg [13] in the random oracle model in 2002 and has been further developed in [3,10]. HIBE is an extension of IBE and achieves private key generation hierarchy. In general, HIBE can handle a similar problem as MISKD does, but they are different. In HIBE, it is required that all public keys be associated with a single identity, while in MISKD there is no such requirement. To further clarify this, let us take a look at the following example. Assume that "Alice" is an identity for Alice and she has two email addresses: alice@gmail.com and alice@hotmail.com. The public keys of Alice are then "Alice||alice@gmail.com" and "Alice||alice@hotmail.com", which are associated with "Alice" as an identity. This implies that Alice's private key is also associated with "Alice" as an identity. If Alice has another email address: me@gmail.com, which is used as her public key, then HIBE is not applicable (while this case can still be handled with MISKD).

Fuzzy IBE was introduced by Sahai and Waters [17]. A FIBE scheme allows a private key for an identity ID to decrypt a ciphertext encrypted with another

Table 1. Further comparison with the GMC's scheme. In order to present the comparison clearly, we use the same parameter n, which denotes the maximum number of private keys that can be accumulated into a single one in both the GMC scheme and ours. u is a common parameter in ciphertext in both schemes.

Scheme	Security Model	Assumption	Time cost on u
[12]	Random Oracles	n-BSDH	liner $(n-1)$
Our scheme	selective-ID	DBDH	$n-1$

identity ID' if and only if the identities ID and ID' are close to each other as measured by some metric. A FIBE scheme allows an encryption with a biometric input as the public key so that the ciphertext can still be decrypted even if the public key is partially changed. Like MISKD, FIBE is an encryption scheme where a private key can map multiple public keys, but all the public keys should satisfy some metric. E.g., when we view the identity as an n-bit vector, the difference between all public keys should not be more than d bits. However, the goal of FIBE is different from that of MISKD and it does not possess the properties of MISKD.

In this paper, we present a novel MISKD scheme that solves the open problems mentioned earlier in this section. We prove its security in the selective-ID model based on the Decisional Bilinear Diffie-Hellman assumption. Our scheme is more efficient than the scheme proposed in [12].

Road Map: In Section 2, we will provide the definitions of our scheme, including security requirements and some preliminaries. In Sections 3 and 4, we propose our MISKD schemes and its security proofs against IND-ID-CPA and IND-ID-CCA attacks. In Section 5, we give a note to our MISKD scheme. We conclude our paper in Section 6.

2 Definitions

In this section, we define our MISKD system and its security requirements.

Definition 1. *A MISKD scheme can be described as the four PPT algorithms:* Setup*,* KeyGen*,* Encrypt *and* Decrypt*.*

- Setup*: takes as input a security parameter 1^k, and outputs a master secret key α and the corresponding master public parameters params.*
- KeyGen*: takes as input the multiple public keys $IDs = \langle ID_1, ID_2, \cdots, ID_l \rangle$ $(1 \le l \le n)$ of a party, the params and the master secret key α, and outputs a single private key d_{IDs} for the party.*
- Encrypt*: takes as input a message M, the params and an $ID_i \in IDs$, and outputs ciphertext C.*
- Decrypt*: takes as input the ciphertext $\langle ID_i, C \rangle$, the private key d_{IDs} and other ID's $\{ID_1, \cdots, ID_{i-1}, ID_{i+1}, \cdots ID_l\}$, and outputs the message M.*

2.1 Security Model

Boneh and Boyen defined the model of selective-ID secure against the chosen ciphertext attack in [2], denoted by IND-sID-CCA. Our MISKD scheme has IND-sID-CCA security, defined as follows:

Init: The adversary outputs an identity ID^*, where it wishes to be challenged.

Setup: The challenger takes as input a security parameter 1^k, and then runs the algorithm Setup. It gives the adversary the resulting master public parameters denoted by $params$ and keeps the master secret key to itself.

Phase 1: The adversary makes queries $q_1, q_2, \cdots, q_m$. When the queries are for key generation, the adversary can make at most n queries at one time:

– Key generation queries $IDs = \langle ID_{i+1}, ID_{i+2}, \cdots, ID_{i+l}\rangle$ $(1 \le l \le n)$. The challenger responds by running algorithm KeyGen to generate the private key d_{IDs} corresponding to the multiple public keys IDs. It sends d_{IDs} to the adversary.
– Decryption query $\langle ID_i, C_i : IDs\rangle$. The challenger responds by running algorithm KeyGen to generate the private key d_{IDs} corresponding to IDs. It then runs algorithm $\mathsf{Decrypt}$ to decrypt the ciphertext $\langle ID_i, C_i\rangle$ using the private key d_{IDs} and IDs $(ID_i \in IDs)$ and sends the resulting plaintext to the adversary.

These queries may be asked adaptively according to the replies of queries.

Remarks: We enhance the power of adversary in the decryption query; namely, which private key to decrypt the ciphertext can be decided by the adversary. Compared to GMC's definition [12] where the decryption query is the same as that of a general IBE system, we have weakened our security model. Our definition is more realistic in terms of the property of MISKD, where any party suffering from attack may hold a single private key for multiple public keys.

Challenge: Once the adversary decides that **Phase 1** is over it outputs two equal length plaintexts M_0, M_1 on which it wishes to be challenged. The challenger picks a random bit $b_r \in \{0,1\}$ and sets $C_{ch} = \mathsf{Encrypt}(params, ID^*, M_{b_r})$. It sends C_{ch} as the challenge to the adversary.

Phase 2: It is the same as **Phase 1**. The constraint is that the adversary cannot make a key generation query on ID^* (any key generation queries on IDs should not include ID^*, i.e. $ID^* \notin IDs$) or decryption query on (ID^*, C_{ch}).

Guess: The adversary outputs a guess $b_g \in \{0,1\}$ and wins the game if $b_g = b_r$.

We refer to such an adversary $\mathcal{A}$ as an IND-sID-CCA adversary. We define the advantage of adversary $\mathcal{A}$ in attacking the scheme $\mathcal{E}$ as

$$Adv_{\mathcal{E},\mathcal{A}} = \left|\mathsf{Pr}[b_g = b_r] - \frac{1}{2}\right|.$$

The probability is over the random bits used by the challenger and the adversary.

Definition 1. *We say that a MISKD system $\mathcal{E}$ is $(t, q_{ID}, q_C, \epsilon)$-adaptively chosen ciphertext secure if for any t-time IND-sID-CCA adversary $\mathcal{A}$ that makes at most q_{ID} chosen private key queries and at most q_C chosen decryption queries we have that $Adv_{\mathcal{E}\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is $(t, q_{ID}, q_C, \epsilon)$ IND-sID-CCA secure.*

Definition 2. *We say that a MISKD system $\mathcal{E}$ is (t, q_{ID}, ϵ)-adaptively chosen plaintext secure if $\mathcal{E}$ is $(t, q_{ID}, 0, \epsilon)$ chosen ciphertext secure. As shorthand, we say that $\mathcal{E}$ is (t, q_{ID}, ϵ) IND-sID-CPA secure.*

2.2 Bilinear Pairing

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$. A map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is called a bilinear pairing (map) if this map satisfies the following properties:

- Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degeneracy: $e(g, g) \neq 1$. In other words, if g be a generator of $\mathbb{G}$, then $e(g, g)$ generates $\mathbb{G}_T$.
- Computability: There is an efficient algorithm to compute $e(u, v)$ for all $u, v \in \mathbb{G}$.

2.3 Complexity Assumption

The Bilinear Diffie-Hellman (BDH) problem in $\mathbb{G}$ is as follows: given a tuple $g, g^a, g^b, g^c \in \mathbb{G}$ as input, output $e(g, g)^{abc}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving BDH in $\mathbb{G}$, if

$$\mathsf{Pr}[\mathcal{A}(g, g^a, g^b, g^c) = e(g, g)^{abc}] \geq \epsilon,$$

where the probability is over the random choice of generator g in $\mathbb{G}^*$, the random choice of a, b, c in $\mathbb{Z}_p$, and random bits used by $\mathcal{A}$. Similarly, we say that an algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the Decisional Bilinear Diffie-Hellman (DBDH) problem in $\mathbb{G}$ if

$$\left|\mathsf{Pr}[\mathcal{B}(g, g^a, g^b, g^c, e(g, g)^{abc}) = 0] - \mathsf{Pr}[\mathcal{B}(g, g^a, g^b, g^c, Z) = 0]\right| \geq \epsilon,$$

where the probability is over the random choice of generator g in $\mathbb{G}^*$, the random choice of a, b, c in $\mathbb{Z}_p$, the random choice of $Z \in \mathbb{G}_T$, and the random bits consumed by $\mathcal{B}$.

Definition 3. *We say that the (t, ϵ)-DBDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the DBDH problem in $\mathbb{G}$.*

2.4 Identity Division

In our MISKD scheme, our identity space is $\mathbb{Z}_p^*$. We can divide the identity space $\mathbb{Z}_p^*$ into n parts by mod n. n is the number value that we can set a single private key d_{IDs} for n public keys at most. We denote by $\mathfrak{B}_i$ the sub-space of $\{x : x \in \mathbb{Z}_p^* \wedge x = i - 1 (\mathrm{mod}\ n)\}$ for $i = 1, 2, \cdots, n$. The identities for private keys should be read as follows:

- Since $n \ll p$ and ID is constructed with an identity if $ID \in \mathfrak{B}_i$, we can reconstruct it such that $ID' \in \mathfrak{B}_j$ easily. For private key revocation in an email system, the Private Key Generator (PKG) actually sets the public-key with the email address and a valid time. Then, it's easy to set a string of public key such that it lies in a right sub-space.
- A private key d_{IDs} for $ID_1, ID_2, \cdots, ID_l$ $(1 \leq l \leq n)$ can only be computed if and only if $ID_1, ID_2, \cdots, ID_l$ belong to different sub-spaces $\mathfrak{B}_1, \mathfrak{B}_2, \cdots, \mathfrak{B}_l$. We denote by $IDs = \langle ID_1, ID_2, \cdots, ID_l : ID_i \in \mathfrak{B}_i \rangle$ all public keys that satisfy the condition.
- For each sub-space $\mathfrak{B}_i$, we randomly choose a secret $\beta_i \in \mathbb{Z}_p^*$, which will be used in both algorithm KeyGen and algorithm Encrypt for any $ID \in \mathfrak{B}_i$.

3 Construction I: Chosen-Plaintext Security

We construct an efficient MISKD scheme that is IND-sID-CPA secure without random oracles. The MISKD scheme will be convenient for us to present our final scheme.

3.1 Construction

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be the bilinear map, $\mathbb{G}, \mathbb{G}_T$ be two cyclic groups of order p and g be the corresponding generator in $\mathbb{G}$. The MISKD scheme can be described as follows:

Setup: The system parameters are generated as follow. Choose at random a secret value $\alpha \in \mathbb{Z}_p$, choose g, g_2, h randomly from $\mathbb{G}$, and set the value $g_1 = g^\alpha$. In addition, compute $k_1, k_2, \cdots, k_n$ from $\beta_1, \cdots, \beta_n \in \mathbb{Z}_p$ (defined in section 2) such that $k_i = g^{\beta_i}$. The master public *params* and master secret key are

$$params = (g, g_1, g_2, h, k_1, k_2, \cdots, k_n),\ master\ secret\ key = \alpha.$$

Remarks: The PKG and the sender should judge which sub-space $\mathfrak{B}$ the public key belongs to in both algorithm KeyGen and algorithm Encrypt. According to the definition in Section 2 and the parameters $k_1, k_2, \cdots, k_n$, k_i will be chosen as the parameter in key generation and encryption when $ID \in \mathfrak{B}_i$.

KeyGen:

- To generate a private key for $ID_i \in \mathfrak{B}_i$, pick a random $r \in \mathbb{Z}_p$ and output:

$$d_{ID_i} = (d_1, d_2) = \big(g_2^\alpha (h k_i^{ID_i})^r, g^r\big).$$

  Note that this single ID case is a special case of multiple identities given below. The reason we present this simple case prior to the general case is that it will help the reader to understand the security proof given in the next section.

– To generate a private key for $IDs = \langle ID_1, ID_2, \cdots, ID_l : ID_i \in \mathfrak{B}_i \rangle$ $(1 \leq l \leq n)$, pick a random $r \in \mathbb{Z}_p$ and output:

$$d_{IDs} = (d_1, d_2) = \big(g_2^{\alpha}(hk_1^{ID_1}k_2^{ID_2} \cdots k_l^{ID_l})^r, g^r\big).$$

Encrypt: To encrypt a message $M \in \mathbb{G}_T$ under the public key $ID_i \in \mathfrak{B}_i$, pick a random $s \in \mathbb{Z}_p$ and outputs:

$$C = ((hk_i^{ID_i})^s, k_1^s, \cdots, k_{i-1}^s, k_{i+1}^s, \cdots, k_n^s, g^s, e(g_1, g_2)^s \cdot M).$$

Decrypt: Let $C = (u_{ID_i}, u_1, \cdots, u_{i-1}, u_{i+1}, \cdots, u_n, v, w)$ be a valid encryption for $ID_i \in \mathfrak{B}_i$.

– To decrypt C with d_{ID_i}, compute:

$$w \cdot \frac{e(d_2, u_{ID_i})}{e(d_1, v)} = (e(g_1, g_2)^s M) \cdot \frac{e(g^r, (hk_i^{ID_i})^s)}{e(g_2^{\alpha}(hk_i^{ID_i})^r, g^s)}.$$

$$= (e(g_1, g_2)^s M) \cdot \frac{e(g^r, (hk_i^{ID_i})^s)}{e(g_1, g_2)^s e((hk_i^{ID_i})^r, g^s)} = M.$$

– To decrypt C with d_{IDs}, where $IDs = \langle ID_1, ID_2, \cdots, ID_l : ID_{i'} \in \mathfrak{B}_{i'} \rangle$ and $ID_i \in IDs$,
  - Compute $u = u_{ID_i} u_1^{ID_1} \cdots u_{i-1}^{ID_{i-1}} u_{i+1}^{ID_{i+1}} \cdots u_l^{ID_l} = (hk_1^{ID_1} \cdots k_l^{ID_l})^s$;
  - Decrypt the ciphertext with u, v, w and d_{IDs}:

$$w \cdot \frac{e(d_2, u)}{e(d_1, v)} = (e(g_1, g_2)^s M) \cdot \frac{e(g^r, (hk_1^{ID_1} \cdots k_l^{ID_l})^s)}{e(g_2^{\alpha}(hk_1^{ID_1} \cdots k_l^{ID_l})^r, g^s)}$$

$$= (e(g_1, g_2)^s M) \cdot \frac{e(g^r, (hk_1^{ID_1} \cdots k_l^{ID_l})^s)}{e(g_1, g_2)^s e((hk_1^{ID_1} \cdots k_l^{ID_l})^r, g^s)} = M.$$

3.2 Security

We now prove the security of our MISKD scheme.

Theorem 1. *Our MISKD scheme is (t, q_{ID}, ϵ) secure assuming the (t', ϵ)-DBDH assumption holds, where $t' < t - \Theta(\tau q_{ID})$ and τ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose there exists a (t, q_{ID}, ϵ)-adversary $\mathcal{A}$ against our scheme, we construct an algorithm $\mathcal{B}$ that solves the DBDH problem. Algorithm $\mathcal{B}$ is given as input a random tuple (g, g^a, g^b, g^c, Z) that Z is either $e(g, g,)^{abc}$ or just a random value in $\mathbb{G}_T$. Set $g_1' = g^a, g_2' = g^b, g_3' = g^c$. $\mathcal{B}$'s goal is to output 1 if $Z = e(g, g)^{abc}$ and 0 otherwise. $\mathcal{B}$ works by interacting with $\mathcal{A}$ as follows:

Initialization. The adversary outputs an identity $ID^* \in \mathfrak{B}_j$, where it wishes to be challenged.

Setup: To generate the master public *params*, $\mathcal{B}$ picks $\beta_1, \beta_2, \cdots, \beta_n, \gamma$ randomly from $\mathbb{Z}_p$, define $g_1 = g_1', g_2 = g_2', k_j = g_1^{\beta_j}, k_i = g^{\beta_i} (i \neq j)$ and $h = g_1^{p-\beta_j ID^*} g^{\gamma}$. $\mathcal{B}$ sends the master public *params* to $\mathcal{A}$, where

$$params = (g, g_1, g_2, h, k_1, k_2, \cdots, k_n).$$

Phase 1: $\mathcal{A}$ makes the private key queries. To respond a private key query on $ID \in \mathbb{Z}_p$, $\mathcal{B}$ does:

– If $ID \in \mathfrak{B}_j$, $\mathcal{B}$ picks a random $r \in \mathbb{Z}_p$ and constructs the private key d_{ID} as

$$d_{ID} = (d_1, d_2) = (g_2^{\frac{-\gamma}{p-\beta_j ID^*+\beta_j ID}} (g_1^{p-\beta_j ID^*+\beta_j ID} g^{\gamma})^r, g^r g_2^{-\frac{1}{p-\beta_j ID^*+\beta_j ID}}).$$

Let $\widetilde{r} = r - \frac{b}{p-\beta_j ID^*+\beta_j ID}$. Then, we have

$$\begin{aligned} d_1 &= g_2^{\frac{-\gamma}{p-\beta_j ID^*+\beta_j ID}} \cdot (g_1^{p-\beta_j ID^*+\beta_j ID} g^{\gamma})^r \\ &= g_2^a (g_1^{p-\beta_j ID^*+\beta_j ID} g^{\gamma})^{-\frac{b}{p-\beta_j ID^*+\beta_j ID}} \cdot (g_1^{p-\beta_j ID^*+\beta_j ID} g^{\gamma})^r \\ &= g_2^a (g_1^{p-\beta_j ID^*+\beta_j ID} g^{\gamma})^{r-\frac{b}{p-\beta_j ID^*+\beta_j ID}} \\ &= g_2^a (h k_j^{ID})^{\widetilde{r}}. \end{aligned}$$

$$d_2 = g^r g_2^{-\frac{1}{p-\beta_j ID^*+\beta_j ID}} = g^{r-\frac{b}{p-\beta_j ID^*+\beta_j ID}} = g^{\widetilde{r}}.$$

So, $d_{ID} = (d_1, d_2) = (g_2^a (h k_j^{ID})^{\widetilde{r}}, g^{\widetilde{r}})$ is a valid private key, $\mathcal{B}$ gives it to $\mathcal{A}$.

– Else $(ID \in \mathfrak{B}_i, i \neq j)$, $\mathcal{B}$ picks a random $r \in \mathbb{Z}_p$ and constructs d_{ID}, as

$$d_{ID} = (d_1, d_2) = (g_2^{\frac{-(\gamma+\beta_i ID)}{p-\beta_j ID^*}} (g_1^{p-\beta_j ID^*} g^{\gamma+\beta_i ID})^r, g^r g_2^{-\frac{1}{p-\beta_j ID^*}}).$$

Let $\widetilde{r} = r - \frac{b}{p-\beta_j ID^*}$. Then, we have

$$\begin{aligned} d_1 &= g_2^{\frac{-(\gamma+\beta_i ID)}{p-\beta_j ID^*}} \cdot (g_1^{p-\beta_j ID^*} g^{\gamma+\beta_i ID})^r \\ &= g_2^a (g_1^{p-\beta_j ID^*} g^{\gamma+\beta_i ID})^{-\frac{b}{p-\beta_j ID^*}} \cdot (g_1^{p-\beta_j ID^*} g^{\gamma+\beta_i ID})^r \\ &= g_2^a (g_1^{p-\beta_j ID^*} g^{\gamma+\beta_i ID})^{r-\frac{b}{p-\beta_j ID^*}} \\ &= g_2^a (h k_i^{ID})^{\widetilde{r}}. \end{aligned}$$

$$d_2 = g^r g_2^{-\frac{1}{p-\beta_i ID^*}} = g^{r-\frac{b}{p-\beta_j ID^*}} = g^{\widetilde{r}}.$$

So, $d_{ID} = (d_1, d_2) = (g_2^a (h k_i^{ID})^{\widetilde{r}}, g^{\widetilde{r}})$ is a valid private key, $\mathcal{B}$ gives it to $\mathcal{A}$.

To generate a private key query on $IDs = \langle ID_1, ID_2, \cdots, ID_l : ID_i \in \mathfrak{B}_i \rangle$ $(1 \le l \le n)$, $\mathcal{B}$ does the following.

- If an ID lies in IDs such that $ID \in \mathfrak{B}_j$, letting $IDs' = \langle ID_{1'}, ID_{2'}, \cdots, ID_{(l-1)'} : ID_{i'} \in \mathfrak{B}_{i'} \rangle = \{IDs\} \backslash \{ID\}$, $\mathcal{B}$ constructs the single private key $d_{IDs} = (d_{IDs,1}, d_{IDs,2})$ as follows:
  - Compute the private key $d_{ID} = (d_1, d_2)$ $(ID \in \mathfrak{B}_j)$, the same as the above (with a new random $r \in \mathbb{Z}_p$).
  - Compute $d_{IDs} = (d_{IDs,1}, d_{IDs,2})$ as

$$d_{IDs} = (d_{IDs,1}, d_{IDs,2}) = (d_1 (d_2)^{ID_{1'}\beta_{1'} + \cdots + ID_{(l-1)'}\beta_{(l-1)'}}, d_2).$$

Let $d_1 = g_2^a (hk_j^{ID})^{\widetilde{r}}, d_2 = g^{\widetilde{r}}$. Then, we have

$$\begin{aligned} d_{IDs,1} &= d_1 (d_2)^{ID_{1'}\beta_{1'} + \cdots + ID_{(l-1)'}\beta_{(l-1)'}} \\ &= g_2^a (hk_j^{ID})^{\widetilde{r}} (g^{\widetilde{r}})^{ID_{1'}\beta_{1'} + \cdots + ID_{(l-1)'}\beta_{(l-1)'}} \\ &= g_2^a (hk_j^{ID})^{\widetilde{r}} (g^{ID_{1'}\beta_{1'} + \cdots + ID_{(l-1)'}\beta_{(l-1)'}})^{\widetilde{r}} \\ &= g_2^a (hk_j^{ID} k_{1'}^{ID_{1'}} \cdots k_{(l-1)'}^{ID_{(l-1)'}})^{\widetilde{r}} \\ &= g_2^a (hk_1^{ID_1} \cdots k_l^{ID_l})^{\widetilde{r}}. \end{aligned}$$

So, $d_{IDs} = (d_{IDs,1}, d_{IDs,2}) = (g_2^a (hk_1^{ID_1} \cdots k_l^{ID_l})^{\widetilde{r}}, g^{\widetilde{r}})$ is a valid private key, $\mathcal{B}$ gives it to $\mathcal{A}$.

- Else, (each $ID \in \mathfrak{B}_i, i \ne j$), let $\beta IDs = ID_1\beta_1 + \beta_2 ID_2 + \cdots + \beta_l ID_l$. $\mathcal{B}$ picks a random r and constructs the single private key $d_{IDs} = (d_{IDs,1}, d_{IDs,2})$ as

$$d_{IDs} = (d_{IDs,1}, d_{IDs,2}) = (g_2^{\frac{-(\gamma+\beta IDs)}{p-\beta_j ID^*}} (g_1^{p-\beta_j ID^*} g^{\gamma+\beta IDs})^r, g^r g_2^{-\frac{1}{p-\beta_j ID^*}}).$$

Let $\widetilde{r} = r - \frac{b}{p-\beta_j ID^*}$. Then, we have

$$\begin{aligned} d_{IDs,1} &= g_2^{\frac{-(\gamma+\beta IDs)}{p-\beta_j ID^*}} \cdot (g_1^{p-\beta_j ID^*} g^{\gamma+\beta IDs})^r \\ &= g_2^a (g_1^{p-\beta_j ID^*} g^{\gamma+\beta IDs})^{-\frac{b}{p-\beta_j ID^*}} \cdot (g_1^{p-\beta_j ID^*} g^{\gamma+\beta IDs})^r \\ &= g_2^a (g_1^{p-\beta_j ID^*} g^{\gamma+\beta IDs})^{r-\frac{b}{p-\beta_j ID^*}} \\ &= g_2^a (hg^{\beta IDs})^{\widetilde{r}} \\ &= g_2^a (hk_1^{ID_1} k_2^{ID_2} \cdots k_l^{ID_l})^{\widetilde{r}}. \end{aligned}$$

$$d_{IDs,2} = g^r g_2^{-\frac{1}{p-\beta_i ID^*}} = g^{r-\frac{b}{p-\beta_j ID^*}} = g^{\widetilde{r}}.$$

So, $d_{IDs} = (d_{IDs,1}, d_{IDs,2}) = (g_2^a (hk_1^{ID_1} k_2^{ID_2} \cdots k_l^{ID_l})^{\widetilde{r}}, g^{\widetilde{r}})$ is a valid private key. $\mathcal{B}$ gives it to $\mathcal{A}$.

Challenge: When $\mathcal{A}$ decides that phase 1 is over, it outputs two messages $M_0, M_1 \in \mathbb{G}_T$ on which it wishes to be challenged. $\mathcal{B}$ picks a random $b_r \in \{0,1\}$ and constructs the ciphertext as

$$\begin{aligned} C &= (u_{ID^*}, u_1, \cdots, u_{j-1}, u_{j+1}, \cdots, u_n, v, w) \\ &= ((g_3')^{\gamma}, (g_3')^{\beta_1} \cdots, (g_3')^{\beta_{j-1}}, (g_3')^{\beta_{j+1}}, \cdots, (g_3')^{\beta_n}, g_3', Z \cdot M_{b_r}). \end{aligned}$$

Suppose that $Z = e(g,g)^{abc}$, we have

$$(g_3')^{\gamma} = (g_1^{p-\beta_j ID^*} g_1^{\beta_j ID^*} g^{\gamma})^c = (hk_j^{ID^*})^c.$$

$$(g_3')^{\beta_i} = (k_i)^c, g_3' = g^c, \quad Z \cdot M_{b_r} = e(g_1, g_2)^c \cdot M_{b_r}.$$

Then, the ciphertext $C = ((hk_j^{ID^*})^c, k_1^c, \cdots, k_{j-1}^c, k_{j+1}^c, \cdots, k_n^c, g^c, e(g_1,g_2)^c \cdot M_{b_r})$ is valid.

Phase 2: It is the same as Phase 1. $\mathcal{B}$ repeats the process as in phase 1.

Guess: Finally, $\mathcal{A}$ outputs b_g and $\mathcal{B}$ outputs 1 if $b_g = b_r$; otherwise, it outputs 0.

When the input of $Z = e(g,g)^{abc}$, $\mathcal{A}$'s view is identical to its view in a real attack game; therefore $\mathcal{A}$ must satisfy $|\mathsf{Pr}[b_g = b_r] - 1/2| \geq \epsilon$. On the other hand, when Z is random then $\mathsf{Pr}[b_g = b_r] = 1/2$. Therefore, with g uniform in $\mathbb{G}$, a, b, c uniform in $\mathbb{Z}_p$, and Z uniform in $\mathbb{G}_T$, we have that

$$\left|\mathsf{Pr}[\mathcal{B}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 0] - \mathsf{Pr}[\mathcal{B}(g, g^a, g^b, g^c, Z) = 0]\right| \geq \left|\frac{1}{2} + \epsilon - \frac{1}{2}\right| = \epsilon$$

as required. This concludes the proof of Theorem 1. □

4 Construction II: Chosen-Ciphertext Security

Canetti *et al.* [8] provides an efficient way to construct a chosen ciphertext IBE (IND-sID-CCA) from a chosen plaintext 2-level HIBE. This work was further improved by Boneh and Katz [5]. Based on our MISKD scheme with IND-sID-CPA security presented in section 3 and the approach due to Canetti *et al.* [8], we are able to construct a MISKD scheme with chosen ciphertext security.

4.1 Construction

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be the same bilinear map as in our first scheme. Construction II of our MISKD scheme is described as follows:

Setup: The system parameters are generated as follow. Choose at random a secret $\alpha \in \mathbb{Z}_p$. Choose at random g, g_2, h_1, h_2 from $\mathbb{G}$, and set $g_1 = g^{\alpha}$. Compute $k_1, k_2, \cdots, k_n$ and a one-time unforgeable signature scheme $\mathsf{Sig} = (\mathcal{G}; \mathsf{Sign}; \mathsf{Vrfy})$ in which the verification key (in $\mathbb{Z}_p$) is generated by $\mathcal{G}(1^k)$. The master public *params* and the master secret key are

$$params = (g, g_1, g_2, h_1, h_2, k_1, k_2, \cdots, k_n, \mathsf{Sig}), \quad master\ secret\ key = \alpha.$$

KeyGen: As in the first scheme.

Encrypt: To encrypt a message $M \in \mathbb{G}_T$ under the public key $ID_i \in \mathfrak{B}_i$, do the following:

- Input the security parameter 1^k to Sig and output a pair (vk, sk), where vk is a verification key and sk is a signing key.
- Pick a random $s \in \mathbb{Z}_p$ and output

$$C_m = ((h_1 k_i^{ID_i})^s, (h_2 g_1^{vk})^s, k_1^s, \cdots, k_{i-1}^s, k_{i+1}^s, \cdots, k_n^s, g^s, e(g_1, g_2)^s \cdot M).$$

- Set the signature $\sigma = \mathsf{Sign}_{sk}(C_m)$ with sk and output the ciphertext

$$\begin{aligned} C &= (vk, \sigma, C_m) \\ &= (vk, \sigma, (h_1 k_i^{ID_i})^s, (h_2 g_1^{vk})^s, k_1^s, \cdots, k_{i-1}^s, k_{i+1}^s, \cdots, k_n^s, g^s, e(g_1, g_2)^s \cdot M). \end{aligned}$$

Decrypt: Let $C = (vk, \sigma, u_{ID_i}, u_{vk}, u_1, \cdots, u_{i-1}, u_{i+1}, \cdots, u_n, v, w)$ be a valid encryption for ID_i, where $ID_i \in \mathfrak{B}_i$.

- To decrypt C with d_{ID_i}, conduct the following tasks.
  - Check whether $\mathsf{Vrfy}_{vk}(C_m, \sigma) = 1$, if not, reject. Otherwise, continue;
  - Pick a random $r' \in \mathbb{Z}_p$ and compute the 2-level private key $d_{ID_i|vk}$ with d_{ID_i} and vk as

$$d_{ID|vk} = (d_1, d_2, d_3)_{vk} = \big(g_2^{\alpha}(h_1 k_i^{ID})^r (h_2 g_1^{vk})^{r'}, g^r, g^{r'}\big).$$

  - Decrypt the ciphertext with u_{ID_i}, u_{vk}, v, w and $d_{ID|vk}$:

$$\begin{aligned} w \cdot \frac{e(d_2, u_{ID_i}) e(d_3, u_{vk})}{e(d_1, v)} &= (e(g_1, g_2)^s M) \cdot \frac{e(g^r, (h_1 k_i^{ID_i})^s) \cdot e(g^{r'}, (h_2 g_1^{vk})^s)}{e(g_2^{\alpha}(h_1 k_i^{ID_i})^r (h_2 g_1^{vk})^{r'}, g^s)} \\ &= (e(g_1, g_2)^s M) \cdot \frac{e(g^r, (h_1 k_i^{ID_i})^s) \cdot e(g^{r'}, (h_2 g_1^{vk})^s)}{e(g_1, g_2)^s \cdot e((h_1 k_i^{ID_i})^r, g^s) \cdot e((h_2 g_1^{vk})^{r'}, g^s)} \\ &= M. \end{aligned}$$

- To decrypt C with d_{IDs}, where $IDs = \langle ID_1, ID_2, \cdots, ID_l : ID_{i'} \in \mathfrak{B}_{i'} \rangle$ and $ID_i \in IDs$, the decryption is as follow:
  - Check whether $\mathsf{Vrfy}_{vk}(C_m, \sigma) = 1$, if not, reject. Else, continue;
  - Pick a random $r' \in \mathbb{Z}_p$ and compute the 2-level private key $d_{IDs|vk}$ with d_{IDs} and vk as

$$d_{IDs|vk} = (d_1, d_2, d_3)_{vk} = \big(g_2^{\alpha}(h_1 k_1^{ID_1} k_2^{ID_2} \cdots k_l^{ID_l})^r (h_2 g_1^{vk})^{r'}, g^r, g^{r'}\big).$$

  - Compute $u = u_{ID_i} u_1^{ID_1} \cdots u_{i-1}^{ID_{i-1}} u_{i+1}^{ID_{i+1}} \cdots u_l^{ID_l} = (h_1 k_1^{ID_1} \cdots k_l^{ID_l})^s$;

- Decrypt the ciphertext with u, u_{vk}, v, w and $d_{IDs|vk}$:

$$
\begin{aligned}
& w \cdot \frac{e(d_2, u)e(d_3, u_{vk})}{e(d_1, v)} \\
&= (e(g_1, g_2)^s M) \cdot \frac{e(g^r, (h_1 k_1^{ID_1} \cdots k_l^{ID_l})^s) \cdot e(g^{r'}, (h_2 g_1^{vk})^s)}{e(g_2^\alpha (h_1 k_1^{ID_1} \cdots k_l^{ID_l})^r (h_2 g_1^{vk})^{r'}, g^s)} \\
&= (e(g_1, g_2)^s M) \cdot \frac{e(g^r, (h_1 k_1^{ID_1} \cdots k_l^{ID_l})^s) \cdot e(g^{r'}, (h_2 g_1^{vk})^s)}{e(g_1, g_2)^s \cdot e((h_1 k_1^{ID_1} \cdots k_l^{ID_l})^r, g^s) \cdot e((h_2 g_1^{vk})^{r'}, g^s)} \\
&= M.
\end{aligned}
$$

4.2 Security

We now prove the security of our MISKD scheme.

Theorem 2. *Our MISKD scheme is $(t, q_{ID}, q_C, \epsilon)$ secure assuming the (t', ϵ)-DBDH assumption holds, where $t' < t - \Theta(\tau q_{ID} + \tau q_C)$ and τ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose there exists a $(t, q_{ID}, q_C, \epsilon)$-adversary $\mathcal{A}$ against our scheme, we construct an algorithm $\mathcal{B}$ that solves the DBDH problem. Algorithm $\mathcal{B}$ is given the same tuple of (g, g^a, g^b, g^c, Z) as in the proof of the first scheme. Set $g_1' = g^a, g_2' = g^b, g_3' = g^c$. The interaction between $\mathcal{B}$ and $\mathcal{A}$ is as follows:

Initialization. The adversary outputs an identity $ID^* \in \mathfrak{B}_j$, where it wishes to be challenged.

Setup: To generate the master public *params*, $\mathcal{B}$ generates the one-time unforgeable signature (vk^*, sk^*) from algorithm Sign, picks $\beta_1, \beta_2, \cdots, \beta_n, \gamma_1, \gamma_2$ randomly from $\mathbb{Z}_p$, and defines $g_1 = g_1', g_2 = g_2', k_j = g_1^{\beta_j}, k_i = g^{\beta_i} (i \neq j)$, and $h_1 = g_1^{p - \beta_j ID^*} g^{\gamma_1}$ and $h_2 = g_1^{p - vk^*} g^{\gamma_2}$. $\mathcal{B}$ sends the master public *params* to $\mathcal{A}$, where

$$params = (g, g_1, g_2, h_1, h_2, k_1, k_2, \cdots, k_n, \mathsf{Sign}).$$

Phase 1:
$\mathcal{A}$ makes the private key queries and $\mathcal{B}$ constructs the private key as in the proof of the first scheme. $\mathcal{A}$ makes the decryption queries $\langle ID, C : IDs \rangle$ for $C = (vk, \sigma, C_m)$ where $ID \in IDs$, $\mathcal{B}$ does the following:

– If $\mathsf{Vefy}_{vk}(C_m, \sigma) \neq 1$, reject. Else, continue;
– If $ID = ID^*$ and $vk = vk^*$, abort;
– If $vk \neq vk^*$, perform the algorithm $\mathsf{Decrypt}$ by the following way:
  - Pick random $r, r' \in \mathbb{Z}_p$ and compute the 2-level private key to $IDs|vk$ directly: Let $k^{IDs} = k_1^{ID_1} k_2^{ID_2} \cdots k_l^{ID_l}$.

$$d_{IDs|vk} = (d_1, d_2, d_3)_{vk}$$

$$= (g_2^{\frac{-\gamma_2}{p-vk^*+vk}}(h_1k^{IDs})^r(g_1^{p-vk^*+vk}g^{\gamma_2})^{r'}, g^r, g^{r'}g_2^{-\frac{1}{p-vk^*+vk}}).$$

Let $\widetilde{r} = r' - \frac{b}{p-vk^*+vk}$, we have

$$\begin{aligned}
d_1 &= g_2^{\frac{-\gamma_2}{p-vk^*+vk}}(h_1k^{IDs})^r(g_1^{p-vk^*+vk}g^{\gamma_2})^{r'} \\
&= g_2^{\frac{-\gamma_2}{p-vk^*+vk}}(g_1^{p-vk^*+vk}g^{\gamma_2})^{r'} \cdot (h_1k^{IDs})^r \\
&= g_2^a(g_1^{p-vk^*+vk}g^{\gamma_2})^{-\frac{b}{p-vk^*+vk}} \cdot (g_1^{p-vk^*+vk}g^{\gamma_2})^{r'} \cdot (h_1k^{IDs})^r \\
&= g_2^a(g_1^{p-vk^*+vk}g^{\gamma_2})^{\widetilde{r}} \cdot (h_1k^{IDs})^r \\
&= g_2^a(h_1k^{IDs})^r(h_2g_1^{vk})^{\widetilde{r}}.
\end{aligned}$$

$$d_3 = g^{r'}g_2^{-\frac{1}{p-vk^*+vk}} = g^{r'-\frac{b}{p-vk^*+vk}} = g^{\widetilde{r}}.$$

Then, $d_{IDs|vk} = (g_2^a(h_1k^{IDs})^r(h_2g_1^{vk})^{\widetilde{r}}, g^r, g^{\widetilde{r}})$ is a valid private key.

- Decryption with $d_{IDs|vk}$ and IDs and algorithm $\mathsf{Decrypt}$. (omitted)

Challenge: When $\mathcal{A}$ decides that phase 1 is over, it outputs two messages $M_0, M_1 \in \mathbb{G}_T$ on which it wishes to be challenged. $\mathcal{B}$ picks a random $b_r \in \{0,1\}$ and constructs the ciphertext as

$$C = (vk^*, \sigma^*, C_m^*) = (vk^*, \sigma^*, u_{vk^*}, C_m) = (vk^*, \sigma^*, (g_3')^{\gamma_2}, C_m).$$

where C_m is constructed same as before and $\sigma^* = \mathsf{Sign}_{sk^*}(C_m)$.

Since $(g_3')^{\gamma_2} = (g^{\gamma_2})^c = (g_1^{p-vk^*}g_1^{vk^*}g^{\gamma_2})^c = (h_2g_1^{vk^*})^c$, then, $C = (vk^*, \sigma^*, C_m^*)$ is a valid ciphertext for ID^* if $Z = e(g,g)^{abc}$.

Phase 2: It is the same as Phase 1. $\mathcal{B}$ repeats the process as in phase 1.

Guess: Same as in the first scheme.

When the signature scheme $\mathsf{Sig} = (\mathcal{G}, \mathsf{Sign}, \mathsf{Vrfy})$ is secure in the sense of strong unforgeability such that an adversary is unable to forge a new signature on a previously-signed message, any ciphertext for ID with vk^* will be rejected. The probability that adversary makes a decryption query on ID^* with vk^* in Phase 1 is negligible for p is a large prime. So, for any decryption query made by $\mathcal{A}$ will be rejected or decrypted correctly by $\mathcal{B}$ except a negligible probability.

When the input of $Z = e(g,g)^{abc}$, $\mathcal{A}$'s view is identical to its view in a real attack game and therefore $\mathcal{A}$ must satisfy $|\mathsf{Pr}[b_g = b_r] - 1/2| \geq \epsilon$. On the other hand, when Z is random, then $\mathsf{Pr}[b_g = b_r] = 1/2$. Therefore, with g uniform in $\mathbb{G}$, a, b, c uniform in $\mathbb{Z}_p$, and Z uniform in $\mathbb{G}_T$, we have that

$$\left|\mathsf{Pr}[\mathcal{B}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 0] - \mathsf{Pr}[\mathcal{B}(g, g^a, g^b, g^c, Z) = 0]\right| \geq \left|\frac{1}{2} + \epsilon - \frac{1}{2}\right| = \epsilon$$

as required. This concludes the proof of Theorem 2. □

5 A Note to Arbitrary Identities

With a universal one-way hash function $H : \{0,1\}^* \to \mathbb{Z}_p$, we can extend our MISKD scheme to handle any ID such that $ID \in \{0,1\}^*$. The extension of MISKD is still provably secure in the selective-ID model against the chosen ciphertext attack. In our proofs, if the adversary cannot find any $ID \in \{0,1\}^*$ such that $H(ID^*) = H(ID)$, the simulation on a private key generation will still work, because the simulation only fails if and only if the query value is equal to $H(ID^*) \in \mathbb{Z}_p$ according to our setting. The simulation on the decryption phase holds even if the identity space changes, because any decryption query can always be simulated except that the verification key vk is equal to vk^*. There, the extension of MISKD maintains the same level of security.

6 Conclusion

We presented the first Multi-Identity Single-Key Decryption (MISKD) without random oracles. Compared to the GMC scheme, our MISKD scheme made the following improvement: Our MISKD scheme is provably secure in selective-ID model based on DBDH assumption without random oracles. Moreover, our scheme in decryption is more efficient; that is, we only need $n-1$ times of exponentiation computation for the parameter u in the ciphertext compared to that of linear $n-1$ times.

Acknowledgement. The authors would like to thank the anonymous reviewers of *Inscrypt 2007* for their comments on this work.

References

1. Benaloh, J., de Mare, M.: One-way accumulators: A decentralized al ternative to digital signatures. In: Advances in Cryptology-Eurocrypt 1993. LNCS, vol. 765, pp. 274–285. Springer-Verlag, Heidelberg (1993)
2. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
3. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
4. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, Springer, Heidelberg (2001)
5. Boneh, D., Katz, J.: Improved efficiency for cca-secure cryptosystems built using identity based encryption. In: Proceedings of RSA-CT (2005)
6. Cheon, J.H.: Security Analysis of the Strong Diffie-Hellman Problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 1–11. Springer, Heidelberg (2006)
7. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: Vitter, J. (ed.) Proc. of the 30th Annual ACM Symposium on Theory of Computing, pp. 209–218. ACM Press, New York (1998)

8. Canetti, R., Halevi, S., Katz, J.: A forward-secure public key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, Springer, Heidelberg (2003)
9. Camenisch, J., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, Springer, Heidelberg (2002)
10. Chatterjee, S., Sarkar, P.: Constant Size Ciphertext HIBE in the Augmented Selective-ID Model and its Extensions `http://eprint.iacr.org/2007/084`
11. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
12. Guo, F., Mu, Y., Chen, Z.: Identity-Based Encryption: How to Decrypt Multiple Ciphertexts Using a Single Decryption Key. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 392–406. Springer, Heidelberg (2007)
13. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
14. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, Springer, Heidelberg (2002)
15. Nguyen, L.: Accumulators from Bilinear Pairings and Applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
16. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
17. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
18. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Kipnis-Shamir Attack on HFE Revisited

Xin Jiang[1], Jintai Ding[2], and Lei Hu[1]

[1] State Key Lab of Information Security, Chinese Academy of Sciences, Beijing, China
{xjiang,hu}@is.ac.cn
[2] University of Cincinnati, USA
and
Technical University of Darmstadt, Germany
ding@math.uc.edu

Abstract. In this paper, we show the claims in the original Kipnis-Shamir attack on the HFE cryptosystems and the improved attack by Courtois that the complexity of the attacks is polynomial in terms of the number of variables are invalid. We present computer experiments and a theoretical argument using basic algebraic geometry to explain why it is so. Furthermore we show that even with the help of the powerful new Gröbner basis algorithm like F_4, the Kipnis-Shamir attack still should be exponential but not polynomial. This again is supported by our theoretical argument.

Keywords: HFE, MinRank, XL algorithm, relinearization, Gröbner basis, multivariate public key cryptosystem.

1 Introduction

The family of multivariate public key cryptosystems [19,5] is considered as one of the main candidates that have the potential to resist the future quantum computer attacks. One of the major research topics in this area is the HFE family of cryptosystems. The HFE encryption systems were presented by Jacques Patarin at Eurocrypt'96 [15], where the fundamental idea is very similar to that of Matsumoto and Imai [14], namely one first builds some polynomial system on a large field and then transforms it into a polynomial system over a vector space of a much smaller field. The first attack on HFE was presented by Kipnis and Shamir [12], where they lifted the public key back to the large field and attacked the system via a so-called MinRank problem [3]. This attack was further improved by Courtois [2] using different methods to solve the associated MinRank problem. The conclusion of these attacks is that to find the secret key and break the HFE cryptosystem is not exponential but polynomial in terms of the number of variables n once one fixes the key parameter D of HFE (or more precisely, $log(D)$). Later it was shown that if one uses new Gröbner basis methods to attack the HFE directly, it should be again not exponential but polynomial [9,11], in particular, Faugère broke one of the challenges set by Patarin. The overall conclusion seems to be that the HFE family itself is over.

However, there are still HFE variants, which we consider viable for practical applications [16,6], and resistant to the Gröbner basis attacks. The possibility of extension of Kipnis-Shamir attack seems to be quite appealing as in the case of the attack on HFEv in [6]. Therefore it seems to be a good idea to do a complete study of the original Kipnis, Shamir, and Courtois work including complete computer experiments to verify the claims and to derive a good estimate on the complexity in terms of practical attacks. To our surprise, our experiments show that the claims made by Kipnis, Shamir, and Courtois are actually invalid in the sense that the timing is far beyond what is expected. This made us to think what happened and we presented a theoretical explanation why this happens using some basic theoretical tools in algebraic geometry. Furthermore, we apply the new Gröbner basis method of Faugère by using the Magma implementations to this problem. Though the performance is clearly much better than the previous methods, it still confirms that the original Kipnis-Shamir attack is not polynomial rather it should be exponential.

The paper is arranged as follows. First we will briefly describe the original Kipnis-Shamir attack and the improvement of Courtois. Then in the next section, we will show that through experiments, the complexity of the attacks of Kipnis-Shamir are not as claimed. We present a theoretical argument why the claims of Kipnis, Shamir, and Courtois are not valid. In the next section, we will show via computer experiments using the Magma implementation of the new Gröbner basis F_4 that if we use the new Gröbner basis algorithm to improve the attack, the timing should be exponential and not polynomial. Then we will present our conclusion.

2 Kipnis-Shamir Attack on the HFE Scheme

2.1 The HFE Scheme

The HFE encryption scheme uses two finite fields. We denote the small field with q elements as $\mathbf{F}$, and $\mathbf{K}$ as its extension field of degree n over $\mathbf{F}$. A recommended choice for HFE is $q = 2$ and $n = 128$. Given a basis of $\mathbf{K}$ over $\mathbf{F}$, we can identity $\mathbf{K}$ with an n-dimensional vector space over $\mathbf{F}$ by $\varphi : \mathbf{K} \to \mathbf{F}^n$ and its inverse φ^{-1}. The design of HFE is based on a univariate polynomial $P(x)$ over $\mathbf{K}$ of the form

$$P(x) = \sum_{i=0}^{r-1}\sum_{j=0}^{r-1} p_{ij} x^{q^i+q^j}, \qquad (1)$$

where the coefficients p_{ij} are randomly chosen from $\mathbf{K}$ and r is much smaller than n so that the degree of $P(x)$ is less than some fixed parameter D. (Here for simplification reason we consider only the case of $P(x)$ being a homogeneous polynomial.) The limitation on the degree D of $P(x)$ is required to make it possible to invert $P(x)$ efficiently at decryption.

Let

$$G(x) = \varphi^{-1} \circ T \circ \varphi \circ P \circ \varphi^{-1} \circ S \circ \varphi(x), \qquad (2)$$

where T and S are two randomly chosen invertible linear transformations on $\mathbf{F}^n$, and they are part of the private key of the HFE scheme together with polynomial $P(x)$. The public key is $\varphi \circ G \circ \varphi^{-1}$, which are n homogeneous quadratic polynomials in n variables on $\mathbf{F}$.

2.2 Kipnis-Shamir Attack

The attack of Kipnis and Shamir on HFE scheme in [12] is done over the big field $\mathbf{K}$. They proved that the linear transformations S and T when lifted to the big field $\mathbf{K}$ have the form

$$S(x) = \sum_{i=0}^{n-1} s_i x^{q^i},\ T^{-1}(x) = \sum_{i=0}^{n-1} t_i x^{q^i}, \tag{3}$$

where $s_i, t_i \in \mathbf{K}$. It simplifies the expression of public key polynomial $G(x)$ to $G(x) = T(P(S(x)))$ using the univariate polynomial form over the big field, which also gives the expression $T^{-1}(G(x)) = P(S(x))$. They rewrote the public key polynomial as a matrix form:

$$G(x) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} g_{ij} x^{q^i+q^j} = \underline{x} G \underline{x}^t, \tag{4}$$

where $G = [g_{ij}]$ is a matrix over $\mathbf{K}$, and $\underline{x} = (x^{q^0}, x^{q^1}, \cdots, x^{q^{n-1}})$ is the vector over $\mathbf{K}$, and $\underline{x}^t$ is its transpose, and this implies that

$$T^{-1}(G(x)) = \sum_{k=0}^{n-1} t_k \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} (g_{i-k,j-k})^{q^k} x^{q^i+q^j}, \tag{5}$$

and

$$P(S(x)) = \underline{x} W P W^t \underline{x}^t, \tag{6}$$

where we use the same notation P to denote a matrix $[p_{ij}]$, W is a specified matrix with its (i,j)-entry $W_{ij} = s_{j-i}^{q^i}$. (Here and henceforth the subscripts are computed modulo n.)

Let G^{*k} be the matrix derived from G by raising all entries of G to the q^k-th power and cyclically rotating all rows and columns of G forwards by k steps. Then $T^{-1}(G(x)) = \underline{x} G' \underline{x}^t$, where

$$G' = \sum_{k=0}^{n-1} t_k G^{*k} = W P W^t. \tag{7}$$

It is not hard to show that both ranks of matrices P and WPW^t do not exceed r, where $r \ll n$ and are roughly $log(D)$. Kipnis and Shamir found that if one made a correct choice for the values of $t_0, t_1, \cdots, t_{n-1}$, then the rank of G' would not be more than r; otherwise for a random choice of values the expected rank would be close to n. The difference between the correct and random choices is clear, and below is a specific method to recovering $(t_0, t_1, \cdots, t_{n-1})$. Surely here in terms of explicit form of the matrix, we need to use the symmetric form of the matrix and in the case of characteristic 2, the diagonal entries shall all be 0.

The matrix G can be easily obtained from the public key of the HFE scheme, then all G^{*k} can be computed. Take $t_0, t_1, \cdots, t_{n-1}$ as n variables. The matrix $G^{'}$ can be represented by G^{*k} and $(t_0, t_1, \cdots, t_{n-1})$. Since its rank does not exceed r, its left kernel, defined as $\{\underline{x} : \underline{x}G' = 0\}$, is an (at least) $n-r$ dimensional vector subspace, and there are $n-r$ independent n-dimensional vectors $\widetilde{x}_1, \cdots, \widetilde{x}_{n-r}$ such that in the kernel. Assigning random values for these vectors in their first $n-r$ entries and taking new variables for each of the remaining r entries, one adds $r(n-r)$ new variables. Each $\widetilde{x}_i G^{'} = 0$ brings n scalar equations over $\mathbf{K}$, a total of $(n-r)n$ equations can be obtained in $n + r(n-r)$ variables $(t_0, t_1, \cdots, t_{n-1}$ and $r(n-r)$ new variables).

These equations are quadratic and form an over-defined system of about n^2 equations in about rn variables where $r \ll n$. In their attack Kipnis and Shamir propose to solve it by relinearization technique. Surely, if they had solved this over-defined system and derived the values of $t_0, t_1, \cdots, t_{n-1}$, it was easy to recover T^{-1} and T, and there is also a specific way to recover S by solving linear over-defined equations over $\mathbf{F}$. Therefore the crucial point of the attack is to recover the transformations T^{-1} and T. The later developed XL algorithm is an improved algorithm over the relinearization method.

Later Courtois pointed out that the point of the attack of Kipnis and Shamir can be viewed as a MinRank problem and he proposed some further improvement on how to find T using some of known methods for the MinRank problem.

3 Can Kipnis-Shamir Attack and Courtois' MinRank Attack Really Work?

Now we would like to do a careful analysis in theory under what condition that the Kipnis-Shamir attack will work.

3.1 Another Look at the Kipnis-Shamir Attack

If we look at the relinearization method, we know immediately that in order for it to work, the equations must satisfy the condition that the solution is actually unique because we expect to find the solution via solving a set of nondegenrate linear equations.

Originally, the part T of the private key of HFE scheme is fixed and its corresponding form, of which the coefficients are $(t_0, t_1, \cdots, t_{n-1})$, in the big field is unique too. Unfortunately, we have equivalent keys.

First, the solutions to our problem is not unique, because if $(a_0, a_1, \cdots, a_{n-1})$ is a solution for $(t_0, t_1, \cdots, t_{n-1})$, then $u(a_0, a_1, \cdots, a_{n-1})$ is still a solution for any constant u. This problem can be easily solved by fixing one variable, say t_0, to be 1. Furthermore, if r is even, we need to fix two variables, because any symmetric matrix over characteristic 2 with 0 diagonal entries of odd size is degenerate. This implies if r is even, if $(a_0, a_1, \cdots, a_{n-1})$ is a solution, then $u(a_0, a_1, \cdots, a_{n-1}) + v(a_{n-1}^q, a_0^q, \cdots, a_{n-2}^q)$ is also a solution.

Then we realize that this is not enough. If $(a_0, a_1, \cdots, a_{n-1})$ is a solution of $(t_0, t_1, \cdots, t_{n-1})$, it is easy to see that $(a_{n-1}^q, a_0^q, \cdots, a_{n-2}^q)$ is also a solution, and furthermore $(a_{n-i}^{q^i}, a_{n-i+1}^{q^i}, \cdots, a_{n-i-1}^{q^i})$ is also a solution for any i from 2 to $n-1$. This is due to the fact that we only use the condition that the rank of $G^{'}$ can not exceed r in Kipnis-Shamir attack not how it looks like, and the fact that raising the q-th powering of the entries of a matrix and rotating its rows and columns accordingly do not change the rank.

This can also be stated as follows.

Proposition 1. *Let the notation $G, T, P, S, G^{'}, G^{*k}$, and W be as defined before; Let $(a_0, a_1, \cdots, a_{n-1})$ be a solution of $(t_0, t_1, \cdots, t_{n-1})$, and the rank of matrix $G^{'} = \sum_{k=0}^{n-1} a_k G^{*k}$ does not exceed r. Given $(\alpha_0^l, \alpha_1^l, \cdots, \alpha_{n-1}^l) = (a_{n-l}^{q^l}, a_{n-l+1}^{q^l}, \cdots, a_{n-l-1}^{q^l})$, the rank of matrix $G^{'l} = \sum_{k=0}^{n-1} \alpha_k^l G^{*k}$ does not exceed r as well, and $G^{'l}$ and $G^{'}$ are actually of the same rank.*

Proof. From Section 2.2, we raise the both sides of equations (5) and (6) to q^l-th powering, and for each $0 \leq l \leq n-1$, we have

$$(T^{-1}(G(x)))^{q^l} = \sum_{k=0}^{n-1} a_k^{q^l} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (g_{i-l-k, j-l-k})^{q^{k+l}} x^{q^i + q^j}, \tag{8}$$

and

$$(P(S(x)))^{q^l} = \underline{x} W^{'} P^{(l)} W^{'t} \underline{x}^t, \tag{9}$$

where $P^{(l)}$ is derived from P by $P_{ij}^{(l)} = P_{i-l,j-l}^{q^l}$, $W^{'}$ is generated from W with that $W_{ij}^{'} = W_{i-l,j-l}^{q^l}$. Therefore, the rank of matrix $W^{'} P^{(l)} W^{'t}$ cannot exceed r as $P^{(l)}$ contains at most r nonzero rows. Equations (5) and (6) are identical, hence (8) and (9) are identical too. Then we have

$$G^{'l} = \sum_{k=0}^{n-1} a_k^{q^l} G^{*(k+l)} = W^{'} P^{(l)} W^{'t}. \tag{10}$$

Substitute k by $k+l$, we get that

$$G^{'l} = \sum_{k=0}^{n-1} a_{k-l}^{q^l} G^{*k} = \sum_{k=0}^{n-1} \alpha_k^l G^{*k}. \tag{11}$$

Obviously, the rank of $G^{'l}$ is the same as that of $P^{(l)}$ and does not exceed r, and $(a_{n-l}^{q^l}, a_{n-l+1}^{q^l}, \cdots, a_{n-l-1}^{q^l})$ is a solution. □

The above proposition states that each solution $(a_0, a_1, \cdots, a_{n-1})$ for $(t_0, t_1, \cdots, t_{n-1})$ is accompanied by $n-1$ additional solutions $(a_{n-l}^{q^l}, a_{n-l+1}^{q^l}, \cdots, a_{n-l-1}^{q^l})$, $1 \leq l \leq n-1$. These solutions are usually different. More precisely, we have the following.

Proposition 2. *Let T be a randomly chosen linear transformation over $\mathbf{F}^n$, and $(a_0, a_1, \cdots, a_{n-1})$ be a solution corresponding to T. Set $(\alpha_0^l, \alpha_1^l, \cdots, \alpha_{n-1}^l) = (a_{n-l}^{q^l}, a_{n-l+1}^{q^l}, \cdots, a_{n-l-1}^{q^l})$, $0 \leq l \leq n-1$. Then*

$$Prob(\alpha_i^j = \alpha_i^k : j \neq k, 0 \leq i, j, k \leq n-1) \leq \mathcal{O}(n^2 q^{-n}).$$

Proof. Since T is a randomly chosen linear transformation over $\mathbf{F}^n$, $(a_0, a_1, \cdots, a_{n-1})$ is a random vector with entries chosen from $\mathbf{K} = GF(q^n)$. By the birthday paradox, we have

$$Prob(a_i = a_j^{q^l} : j \neq i, 0 \leq i, j, l \leq n-1) \leq 1 - (1 - nq^{-n})^n. \tag{12}$$

Since

$$1 - (1 - nq^{-n})^n \leq \mathcal{O}(n^2 q^{-n}), \tag{13}$$

we have

$$\begin{aligned} &Prob(\alpha_i^j = \alpha_i^k : j \neq k, 0 \leq i, j, k \leq n-1) \\ = &Prob(a_i = a_j^{q^l} : j \neq i, 0 \leq i, j, l \leq n-1) \\ \leq &\mathcal{O}(n^2 q^{-n}) \end{aligned} \tag{14}$$

□

This means even if we fix one variable like t_0 to be 1 or two variables if r is even, we still expect that there should be at least n different solutions. Therefore, we can conclude that mostly each variable of the over-defined $(n-r)n$ quadratic equations system in $n + r(n-r)$ variables from Kipnis-Shamir attack has at least about n different solutions. This reminds us the case of the famous challenges of cyclic equations [22].

It is now clear that for this kind of equation system we can not find the solutions by relinearization technique [12]. Then one may ask how about the XL algorithm [13], which is the improved relinearization algorithm. We will argue that for this kind of equation system we can not find the solutions by XL algorithm easily as well.

The key point is the observation that to any system of multivariate polynomial equations, if one variable has d different solutions, we should not be able to solve this system directly by the XL algorithm with the maximum degree of this variable arisen in terms lower than d.

Proposition 3. *Let $P_0(x_0, \cdots, x_{n-1}) = 0, \cdots, P_{m-1}(x_0, \cdots, x_{n-1}) = 0$ be any set of m multivariate polynomial equations in n variables over $\mathbf{K}$; for each x_i, $0 \leq i \leq n-1$, if x_i has d different solutions $\beta_0, \cdots, \beta_{d-1}$ in $\mathbf{K}$, we can not determine the values of x_i directly from the equations generated by the XL algorithm with the maximum degree of this variable arisen in terms lower than d.*

Proof. We can prove it by contradiction. Suppose we get the exact d values of x_i by the equations generated by the XL algorithm with the maximum degree of this variable arisen and noted as $d^{'}$, and $d^{'} < d$. To get the exact values of x_i, the last step of the XL algorithm is linearization to get a univariate polynomial equation just with one variable x_i. While, we all know that the degree of univariate polynomial equation must be at most $d^{'}$ and lower than d. The contradiction is that we can not get d different values $\beta_0, \cdots, \beta_{d-1}$ of x_i by solving a univariate polynomial equation with the degree lower than d. $\square$

The first proposition in this section shows that each variable of the quadratic equations system generated by Kipnis-Shamir attack has at least n solutions; the second proposition in this section supposes that for each variable, we expect to have n different solutions in general; and this proposition shows that if we want to get the solutions of $(t_0, t_1, \cdots, t_{n-1})$ by XL algorithm, we must raise the degree of monomials at least to n in the solving process. This is quite different from what Kipnis and Shamir claimed which should be $log(D)$, which has nothing to do with n. This means the complexity of the attack should be more than what was claimed.

The statements above can be reexplained in terminology of algebraic geometry. Let V be the algebraic variety of the quadratic equations derived from the Kipnis-Shamir attack, and σ be the action of first q-th powering every component of an n-dimensional vector and then cyclically rotating all components right by one. Then V is invariant under the action of the order n cyclic group generated by σ. This variety must contain at least n distinct points (Proposition 2), and the univariate polynomial over $\mathbf{K}$ representing the variety is then of degree n.

We will confirm this with our computer experiment. Furthermore, in our experiment, we have given a toy example that even if we raise the degree of monomials by the XL algorithm to n or even larger than n, we still can not find the solutions.

3.2 What about Courtois' MinRank Attack?

Courtois tried to improve the Kipnis-Shamir attack for basic HFE [2]. From the matrix $G^{'}$ above, instead of by relinearization, he proposed to solve it by MinRank attack directly [3]. Taken $(t_0, t_1, \cdots, t_{n-1})$ as variables, he suggested that we could derive a set of equations from the fact that every $(r+1) \times (r+1)$ submatrix of $G^{'}$ has determinant 0. Therefore, there are $\binom{n}{r+1}^2$ equations with about $\binom{n}{r+1}$ monomials, and it is expected that there are more than $\binom{n}{r+1}$ equations linearly independent so that this equation system can be solved by Gaussian reduction.

However, $(t_0, t_1, \cdots, t_{n-1})$ has at least about n solutions because this MinRank attack does also use the fact that the rank of $G^{'}$ can not exceed r as in Kipnis-Shamir attack, and in the equations of MinRank attack, the degree of monomials is not larger than $r+1$. For $r+1 \ll n$, we can not solve this system by Gaussian reduction from Proposition 3, and we need to go up to degree n to find the solutions.

4 Computer Experiments

We have programmed some experiments of Kipnis-Shamir attack and MinRank attack by Magma V2.11 on a Pentium IV 2.9GHz PC with 512M memory. Our experiments works on the simplest case, where r is 2. From the theoretical argument above, we can fix the variable $t_0 = 1 \in \mathbf{K}$ to decrease the number of solutions, and also we can fix one new variable to 1 when we simulate Kipnis-Shamir attack because $r = 2$ is even. Surely, we also have the experiments without fixing any variable, and they behave essentially in the same way.

4.1 Experiment on Kipnis-Shamir Attack

We choose $q = 2$, $n \in \{5, 6, \cdots, 12\}$, $r = 2$, so $\mathbf{F} = GF(2)$ and $\mathbf{K} = GF(2^n)$; choose $P(x) = ax^3$ and two random invertible linear transformations T and S, where $a \neq 0$ is randomly chosen from $\mathbf{K}$. Following the description in Section 2.2, we derive the quadratic equation system and then try to solve it. In [12] Kipnis and Shamir intended to solve this system by the relinearization technique, while we just use the XL algorithm to simulate it. For each n, select the degree of the parameter [20] needed for the XL algorithm to be $D = 4$ and record the result of experiments in Table 1.

Table 1. Experiment of Kipnis-Shamir Attack with $r = 2$, $D = 4$, and $n \in \{5, 6, \cdots, 12\}$

	$n=5$	$n=6$	$n=7$	$n=8$	$n=9$	$n=10$	$n=11$	$n=12$
equations $n(n-r)$	15	24	35	48	63	80	99	120
variables $r(n-r)+n-2$	9	12	15	18	21	24	27	30
monomials of degree $\leq D$	715	1820	3876	7315	12650	20475	31465	46376
monomials not emergence	105	280	621	1211	2150	3555	5560	8316
number of XL monomials	610	1540	3255	6104	13995	16920	25905	38060
number of XL equations	825	2184	4760	9120	10500	26000	40194	59520
rank of the matrix	556	1408	2983	5605	9658	15586	23893	35143

As the same for each $n \in \{5, 6, 7, 8\}$, select the parameter $D = 5$ and record the experimental result in Table 2.

Table 2. Experiment of Kipnis-Shamir Attack with $r = 2$, $D = 5$, and $n \in \{5, 6, 7, 8\}$

	$n=5$	$n=6$	$n=7$	$n=8$
equations $n(n-r)$	15	24	35	48
variables $r(n-r)+n-2$	9	12	15	18
monomials of degree $\leq D$	2002	6188	15504	33649
monomials not emergence	182	588	1539	3465
number of XL monomials	1820	5600	13965	30184
number of XL equations	3300	10920	28560	63840
rank of the matrix	1738	5363	13403	29020

In both tables, line 4 is the number of the monomials of degree $\leq D$ in $r(n-r)+n-2$ variables. For not all these monomials would appear in the equations in the XL computation, line 5 is the number of these not emerging in the equations; line 6 is the difference of line 4 and line 5, and it is the number of the monomials of those equations; line 7 is the number of equations. For the data of line 7 is larger than that of line 6, we try to solve this system by Gaussian reduction as linearization technique. However, it does not work even though the XL equations are much more than the XL monomials. Then we get the rank of matrix recorded in line 8, which is formed by that each equation as a row and each monomial as a column. In both tables, each number of line 8 is smaller than what is needed to solve the equations, and we are unable to recover the variables $t_0, t_1, \cdots, t_{n-1}$.

4.2 Toy Example of How the XL Algorithm Terminates

In Section 4.1, we have showed that when $D = 4$ or 5 and $n \in \{5, 6, \cdots, 12\}$, XL can not terminate because we can not solve the equations system directly by Gaussian reduction. Therefore, here we fix $n = 5$ and keep all other parameters as before, except that $D \in \{4, 5, 6, 7\}$. Well, $n(n-r)=15$ equations and $r(n–r)+n-2 = 9$ variables of the generated quadratic equation system are invariable as n and r fixed. The result of experiment is recorded in Table 3.

Table 3. Experiment of Increasing D for Solving Equations by XL

	$D=4$	$D=5$	$D=6$	$D=7$
monomials of degree $\leq D$	715	2002	5005	11440
monomials not emergence	105	182	294	450
number of monomials	610	1820	4711	10990
number of equations	825	3300	10725	30030
rank of the matrix	556	1738	4595	10834
difference of lines 4 and 6	54	82	116	156

From this table, we find that the difference between the number of monomials and rank of the matrix is increasing by the growth of D. We can not solve the original equation system when increasing the parameter D of XL algorithm only by a few degrees.

4.3 Experiment of MinRank Attack

Similarly as the previous subsection, we choose $q = 2$, $n \in \{5, 6, \cdots, 10\}$; choose two kinds of public key polynomials: $r = 2$ and $P(x) = ax^3$, and $r = 3$ and $P(x) = ax^3 + bx^5 + cx^6$, where a, b, and c are random elements chosen from $\mathbf{K}$, respectively; choose two random invertible linear transformations T and S.

When $P(x) = ax^3$. Here we can fix two variables. There are $\binom{n}{r+1}^2$ equations with $\binom{n-1}{3} + (n-2)(n-1) + (n-1)$ monomials in $n-2$ variables. We try

Table 4. Simulation of MinRank Attack of $r = 2$ and $P(x) = ax^3$

	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
monomials of degree $\leq r + 1$	20	35	56	84	120	165
rank of the matrix	15	29	49	76	111	155
difference of above two lines	5	6	7	8	9	10

Table 5. Experiment of MinRank Attack of $r = 3$ and $P(x) = ax^3 + bx^5 + cx^6$

	$n = 6$	$n = 7$	$n = 8$	$n = 9$
monomials of degree $\leq r + 1$	126	210	330	495
rank of the matrix	90	161	266	414
difference of above two lines	36	49	64	81

to solve the equation system by Gaussian reduction, and we also find that it is impossible to be solved. Then we record the rank of the matrix, which is formed as above, in Table 4.

When $P(x) = ax^3 + bx^5 + cx^6$. Here $r = 3$, so we can fix one variable, and we choose $n \in \{6, 7, 8, 9\}$. As the same as before, we can not solve this equation system of $\binom{n}{r+1}^2$ equations with $\binom{n}{4} + n\binom{n-1}{2} + n(n-1) + \binom{n}{2} + n$ monomials in $n - 1$ variables. Then we record the rank of the matrix generated from the equation system in Table 5.

We can observe from Tables 4 and 5 that the difference between the number of monomials of degree $r + 1$ and the rank of the matrix is equal to or larger than n and very regular. Therefore, we can conclude that MinRank attack is unsuccessful to recover the secret $t_0, t_1, \cdots, t_{n-1}$.

4.4 Experiment of Solving Equations by F_4

From [18], it is true that XL acts as a redundant version of the F_4 algorithm. Currently it is commonly recognized that the new Gröbner basis algorithm F_4 [7] and F_5 [8] are the most powerful tools to solve polynomial equations [17,21]. Because F_4 is the only one which is publicly available, which is implemented in Magma, to further understand the quadratic equation system generated by the Kipnis-Shamir attack, we should use the Magma implementation of the new Gröbner basis F_4 to test if finding the solutions are indeed still polynomial.

Because of our degree argument, we do not expect Magma to run up to degree n and therefore we expect the complexity to grow up very fast. This time, we run the experiments by Magma V2.13 on a 2.6GHz AMD 64 computer in TU Darmstadt.

In the same way as above, we choose $q = 2$ and $r = 2$. We fix two variables to reduce the number of solutions and then we use Magma to try to find the Gröbner basis of this system. The experiments as expected produce the full triangular Gröbner basis is in lex order, and we get precisely n solutions from

Table 6. Experiment of Solving Equations by F_4

	$n = 10$	$n = 11$	$n = 12$	$n = 13$	$n = 14$	$n = 15$	$n = 16$	$n = 17$	$n = 18$
time (seconds)	0.1	0.16	0.3	0.51	0.9	1.5	2.5	4	6.7
memory (megabit)	8.2	9.1	10	11.4	12.6	15	17	22	32.3
	$n = 19$	$n = 20$	$n = 21$	$n = 22$	$n = 23$	$n = 24$	$n = 25$	$n = 26$	$n = 27$
time (seconds)	10.8	16.3	32.7	50.5	63	91.7	121.5	171.4	218
memory (megabit)	36	48	58	85	75.9	122	145	136.4	203

the Gröbner basis. Meanwhile, our program also verifies that they are indeed the solutions. Therefore, it supports our theoretical argument.

Table 6 below gives the running time and required memory of each n specifically. In Figure 1, we use logarithmic coordinate and take n as X-coordinate and running time and required memory as Y-coordinate respectively. It clearly shows the growing tendency when increasing n. Though the timing and memory data is smaller than what we expected, but for computing Gröbner basis when increasing the degree n, the timing, we still conclude, should be exponential and not polynomial. The reason that the timing and the memory is far less than what we expect is that the degree of the final Gröbner basis is indeed n. Also we want to emphasize that our result is just the simplest and the easiest case of the HFE family.

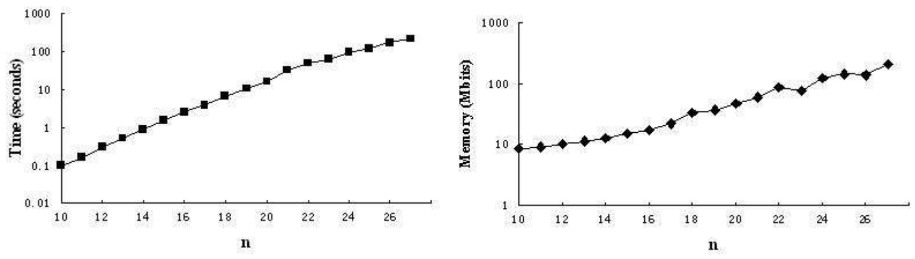


Fig. 1. Running Time and Required Memory

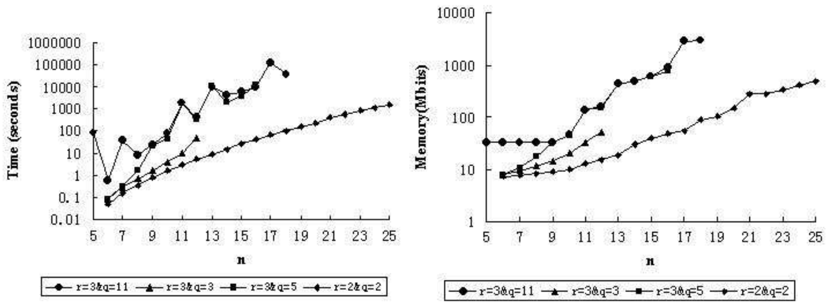


Fig. 2. Running Time and Required Memory Between Different q and r

We did some more experiments by increasing r to 3 or choosing different small field $\mathbf{F}$ as $q = 3, 5, 11$, and the result is compared by Figure 2. It shows that in this situation the equations are much more difficult to solve. This means that this set of systems of highly over-defined equations have much more structures that we still do not understand and much more theoretical and experimental work are still needed to understand fully the complexity behavior.

5 Conclusion

We revisited the original Kipnis-Shamir attack on the HFE cryptosystems. We show in theory and experiments that the original Kipnis-Shamir attack on the HFE cryptosystems and the improved attack by Courtois can not work as efficiently as claimed. Furthermore, we showed that even by the new Gröbner basis algorithm F_4, the complexity of the attack should be exponential and not polynomial, though the performance of F_4 is clearly far better than the XL algorithm and more work is still needed to understand what is really going on. The key point of our theoretical argument is based on the simple idea that when solving a polynomial equation system, the degree parameter of the XL or similar algorithm is lower bounded by the number of solutions.

Acknowledgment. The authors would like to thank the anonymous referees for their helpful suggestions. The work of the second author was partially supported by the Charles Phelps Taft Research Center and the Alexander von Humboldt Foundation. The third author was supported by the by NSFC (60573053) and National 863 Project of China (2006AA01Z416).

References

1. Bardet, M., Faugère, J.-C., Salvy, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proceedings of the International Conference on Polynomial System Solving, pp. 71–74 (2004) Previously appeared as INRIA report RR-5049.
2. Courtois, N.T.: The Security of Hidden Field Equations (HFE). In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, Springer, Heidelberg (2001)
3. Courtois, N.T.: The Minrank Problem. MinRank, a new Zero-knowledge scheme based on the NP-complete problem. Presented at the rump session of Crypto (2000), available at: `http://www.minrank.org`
4. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
5. Ding, J., Gower, J., Schmidt, D.: Multivariate Public-Key Cryptosystems. In: Advances in Information Security, Springer, Heidelberg (2006)
6. Ding, J., Schmidt, D.S.: Cryptanalysis of HFEV and Internal Perturbation of HFE. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 288–301. Springer, Heidelberg (2005)
7. Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner Bases (F_4). Journal of Plure and Applied Algebra 139, 61–88 (1999)

8. Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero(F_5). In: Mora, T. (ed.) Proceeding of ISSAC, pp. 75–83. ACM Press, New York (2002)
9. Faugère, J.-C.: Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 44–60. Springer, Heidelberg (2003)
10. Garey, M.R., Johnson, D.S.: Computers and Intractability – A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
11. Granboulan, L., Joux, A., Stern, J.: Inverting HFE Is Quasipolynomial. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 345–356. Springer, Heidelberg (2006)
12. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, Springer, Heidelberg (1999)
13. Shamir, A., Patarin, J., Courtois, N., Klimov, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, Springer, Heidelberg (2000)
14. Matsumoto, T., Imai, H.: Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988)
15. Patarin, J.: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996)
16. Patarin, J., Courtois, N., Goubin, L.: QUARTZ, 128-Bit Long Digital Signatures. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, Springer, Heidelberg (2001)
17. Diem, C.: The XL-Algorithm and a Conjecture from Commutative Algebra. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 323–337. Springer, Heidelberg (2004)
18. Imai, H., Sugita, M., Faugère, J.-C., Ars, G., Kawazoe, M.: Comparison Between XL and Gröbner Basis Algorithms. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 338–353. Springer, Heidelberg (2004)
19. Wolf, C., Preneel, B.:, Taxonomy of public key schemes based on the problem of multivariate quadratic equations. Cryptology ePrint Archive, Report 2005/077, 64 (12th of May, 2005), `http://eprint.iacr.org/2005/077/`
20. Yang, B.-Y., Chen, J.-M.: All in the XL Family: Theory and Practice. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 67–86. Springer, Heidelberg (2005)
21. Segers, A.J.M.: Algebraic Attacks from a Groebner Basis Perspective Master's Thesis, 10, 110 (2004) `http://www.win.tue.nl/~bdeweger/ReportSegersGB2-11-04.pdf`
22. Björk, G.: Functions of modulus one on $\mathbf{Z}_p$ whose Fourier transforms have constant modulus. In: Proceedings of Alfred Haar Memorial Conference. Colloquia Mathematica Societatis Jnos Bolyai, Budapest, vol. 49 (1985)

Cryptanalysis of General Lu-Lee Type Systems

Haijian Zhou, Ping Luo, Daoshun Wang, and Yiqi Dai

Tsinghua National Laboratory for Information Science and Technology (TNlist)
Department of Computer Science and Technology,
Tsinghua University, Beijing, China, 100084
zhouhj04@mails.tsinghua.edu.cn,
{luop,daoshun,daiyq}@mail.tsinghua.edu.cn

Abstract. The Lu-Lee public key cryptosystem and Adiga-Shankar's modification are considered to be insecure with cryptanalysis by integer linear programing, since only 2 or 3 unknown message blocks are used in the modular linear equation for encryption procedure. Unfortunately integer linear programming algorithms falls in trouble with more unknowns. In this paper we present a probabilistic algorithm for cryptanalysis of general Lu-Lee type systems with n message blocks. The new algorithm is base on lattice reduction and succeeds to break Lu-Lee type systems with up to 68 message blocks.

Keywords: Lu-Lee type systems, cryptanalysis, lattice reduction.

1 Introduction

The Lu-Lee public key cryptosystem [1] was firstly proposed to construct a simple and effective system by solving simultaneous modular linear equations, and achieves much faster encryption and decryption speed than systems such as RSA and ECC. Cryptanalytic algorithms [2-4] show that the basic Lu-Lee cryptosystem is insecure. Later modifications are proposed, including the Adiga-Shankar scheme [5] that extends the Lu-Lee system to 3 unknown message blocks from 2, and the Duan-Nian scheme [6] that employs an exponential factor in the encryption procedure. The Duan-Nian scheme can be broken with a probabilistic algorithm in [7]. For the Adiga-Shankar's modification (and also the basic Lu-Lee system), Brickell and Odlyzko have summarized cryptanalysis based on integer linear programming algorithms [8]. Kannan's integer linear programming algorithm [9] runs in $O(n^{9n} \log r)$ in worst case on problems with n variables and integer coefficients bounded by r. Since $n \leq 4$ in both of the above two systems, Kannan's algorithm is a viable threat to them.

Kannan's integer linear programming algorithm falls in trouble when the number of variables increases. Actually the worst computational complexity grows larger than 2^{100} when $n \geq 5$. On the other hand, we will show in this paper that extending Lu-Lee systems to n message blocks is straight forward from Adiga-Shankar's scheme, which we call general Lu-Lee type systems.

To perform cryptanalysis of such systems, we turn to helps from lattice reduction algorithms. The most famous lattice reduction algorithm is the so-called LLL or L^3 algorithm proposed by Lenstra, Lenstra and Lovász [10]. With the development of fast L^3 algorithms [11-13], lattice reduction has been applied to many fields, especially in cryptology, as summarized in [14].

By applying L^3 to a $d = n + 2$ dimensional lattice constructed from the modular linear equation in Lu-Lee type systems with n message blocks, we show that cryptanalysis is possible to these systems. Actually, in the proposed probabilistic algorithm, the probability for uncovering the unknown message blocks is approximating 1 when $n \le 68$.

The remainder of the paper is organized as following. We first introduce the Lu-Lee type systems in Section 2. Section 3 is the preliminaries for cryptanalysis by lattice reduction. In Section 4 we give the probabilistic algorithm based on lattice reduction for breaking general Lu-Lee type systems, and a detailed description of probability analysis is in Section 5. Section 6 gives the experimental results. We conclude in Section 7.

2 The Lu-Lee Type Systems

We first introduce the basic Lu-Lee cryptosystem and Adiga-Shankar's modification, and then present a general Lu-Lee type system that extends Adiga-Shankar's scheme to n message blocks.

2.1 The Basic Lu-Lee Cryptosystem and Adiga-Shankar's Modification

Let p_1, p_2 be large primes (e.g. 256 bits) and $r = p_1 p_2$ a large composite integer. k_1, k_2 are integers of the same order of magnitude as r and $k_j = a_{ij} \bmod p_i$ for $i, j = 1, 2$. Here a_{ij} are moderate-sized numbers (e.g. 32 bits) satisfying

$$a_{11}a_{22} - a_{12}a_{21} \neq 0 .$$

M_1, M_2 are the upper limits on the message blocks m_1, m_2 , satisfying

$$M_i \le \left\lfloor \frac{1}{2} \min\left\{ \frac{q}{a_{1i}}, \frac{q}{a_{2i}} \right\} \right\rfloor, \ i = 1, 2 ,$$

where $q = \min\{p_1, p_2\}$ and $\lfloor y \rfloor$ denotes the integer part of a real number y .

With the denotations above, the public encryption key is (r, k_1, k_2, M_1, M_2) and the private decryption key $(p_1, p_2, a_{11}, a_{12}, a_{21}, a_{22})$. The encryption procedure is simply

$$c = (k_1 m_1 + k_2 m_2) \bmod r . \qquad (1)$$

For decryption, first compute $c_i = c \bmod p_i$, $i = 1, 2$, and the following simultaneous equations hold:

$$c_i = a_{i1}m_1 + a_{i2}m_2, \ i = 1, 2 .$$

Then by solving the equations we can recover the original message blocks m_1, m_2.

In Adiga-Shankar's modification, three message blocks are used:

$$c = (k_1 m_1 + k_2 m_2 + m) \bmod r . \tag{2}$$

The readers may find more details in [5].

For both of the basic Lu-Lee cryptosystem and Adiga-Shankar's modification, Kannan's integer linear programming algorithm is a viable threat to them since $n \le 4$.

2.2 Extend Lu-Lee System to n Variables

Extending the Lu-Lee system to n variables is straight forward from Adiga-Shankar's scheme, as shown below.

Let $p_1, p_2, \cdots, p_n$ be large primes (e.g. 256 bits) and $r = p_1 p_2 \cdots p_n$ a large composite integer. $k_1, k_2, \cdots, k_n$ are integers of the same order of magnitude as r and

$$k_j = a_{ij} \bmod p_i, \ i, j = 1, 2, \cdots, n .$$

Here a_{ij} are moderate-sized integers (e.g. 128 bits). $M_1, M_2, \cdots, M_n, M_{n+1}$ are the upper limits on the message blocks $m_1, m_2, \cdots, m_n, m_{n+1}$, also moderate-sized (e.g. 120 bits). The public encryption key is $(r, k_1, \cdots, k_n, M_1, \cdots, M_n, M_{n+1})$, and the private decryption key $(p_1, \cdots, p_n, a_{ij})$, $i, j = 1, 2, \cdots, n$. The encryption procedure is simply

$$c = (k_1 m_1 + k_2 m_2 + \cdots + k_n m_n + m_{n+1}) \bmod r . \tag{3}$$

On decryption, first calculate the residues $c_i = c \bmod p_i$, which leads to the simultaneous equations

$$a_{i1}t_1 + a_{i2}t_2 + \cdots + a_{in}t_n = c_i, \ i = 1, 2, \cdots, n .$$

By solving the equations we get $m_i = [t_i]$, $i = 1, 2, \cdots, n$, where $[y]$ denotes the integer closest to the real number y. The last message block m_{n+1} can be easily recovered by solving the equation

$$a_{11}m_1 + a_{12}m_2 + \cdots + a_{1n}m_n + m_{n+1} = c_1 .$$

The proof of the correctness of this modified scheme is similar as that in Adiga-Shankar's paper [5].

For all the previous mentioned Lu-Lee type systems, the encryption functions (1), (2) and (3) can be uniformly denoted by

$$c = \sum_{i=1}^{n} k_i m_i \bmod r\ . \tag{4}$$

We call this a general Lu-Lee type system; construction of such systems is not limited to the method above.

For systems with more than $n \geq 4$ message blocks, Kannan's integer linear programming algorithm falls in trouble that the computational complexity exceeds 2^{100} in worst case. To break such systems, we will turn to helps from lattice reduction.

3 Cryptanalysis Based on Lattice Reduction

3.1 Introduction to L^3 Lattice Reduction

We only present the basic concepts of lattice and lattice reduction here. The readers may find detailed descriptions in [10]. Let $b_1, b_2, \cdots, b_d$ be linear independent vectors in $\mathbb{R}^n$ with $n \geq d$. We denote by

$$L(b_1, b_2, \cdots, b_d) = \left\{ \sum_{i=1}^{d} z_i b_i \mid z_i \in \mathbb{Z} \right\}.$$

the set of all integer linear combinations of the b_i's. This set is called a lattice and $(b_1, b_2, \cdots, b_d)$ a basis of that lattice. Usually $n = d$, and we define the determinant of the lattice L as

$$\det(L) = |\det(b_1, b_2, \cdots, b_n)|,$$

i.e., the absolute value of the matrix formed by $b_1, b_2, \cdots, b_n$.

There are infinitely many basis for a lattice if $d \geq 2$. People are often interested in basis called reduced, which is made of reasonably short vectors which are almost orthogonal. The L^3 lattice reduction algorithm outputs such a good reduced basis, with which we have the following important Theorem [10]:

Theorem 1. Let $b_1, b_2, \cdots, b_d$ be an LLL-reduced basis of the lattice L, then the first vector b_1 satisfies

$$\| b_1 \| \leq 2^{(d-1)/4} \det(L)^{1/d}\ . \tag{5}$$

To enhance the efficiency of L^3 lattice reduction, floating-point (*fp*) algorithms are developed [11-13], and on the theoretic side, the fastest provable *fp*-variants of L^3 is

Nguyen-Stehlé's L^2 given in [12]. For L^3 and L^2 algorithms, the output quality is much better than the worst case in Theorem 1. We have the following heuristic from [13].

Heuristic 1. Let δ be close to 1 and η close to 1/2. Given as input a random basis of almost any lattice L of sufficiently high dimension, L^3 and L^2 with parameters δ and η outputs a basis whose first vector b_1 satisfies

$$\| b_1 \| \approx 1.02^d \det(L)^{1/d} . \tag{6}$$

3.2 Cryptanalysis of General Lu-Lee Type Systems

For the modular linear equation (4) in the general Lu-Lee type system, finding a short solution is an interesting problem. Nguyen and Stern's summarizing paper [14] includes an algorithm based on CVP (closest vector problem [15-16]) to find a short solution to multivariate modular linear equations. For the following multivariate modular linear equations

$$c_j = \sum\nolimits_{i=1}^{n} k_{ji} m_i \bmod r,\ 1 \le j \le l , \tag{7}$$

the algorithm outputs a solution $m = (m_1, m_2, \cdots, m_n)$ approximately satisfying

$$\| m \| < 2^{-n/2} r^{l/n} , \tag{8}$$

where $\| m \|$ denotes the norm of vector m. To break general Lu-Lee type systems, we are more interested in the case that $l = 1$, i.e., a single equation is used, for which we have

$$\| m \| < 2^{-n/2} r^{1/n} \approx 2^{-n/2} p_1 , \tag{9}$$

and the message blocks $m_1, m_2, \cdots, m_n$ also satisfy this bound. Here we assume that all the primes $p_1, p_2, \cdots, p_n$ are of the same order of magnitude and hence $p_1 \approx r^{1/n}$. Look back to the choice of parameters in Section 2, $p_1, p_2, \cdots, p_n$ are 256 bits and $m_1, m_2, \cdots, m_n, m_{n+1}$ are 120 bits integers, so the solution in (9) is approximately $256 - n/2$ bits long. This is a relative short solution, but not satisfying the need that m_i are 120 bits integers when $n < 256$.

We only present an example here that the CVP algorithm above fails to break a general Lu-Lee type system, since the short solution it outputs usually (in most case) is not the one we are interested in. More precisely, given a multivariate modular linear equation as in (4), there is only a unique solution $m^* = (m_1^*, m_2^*, \cdots, m_n^*)$ with the unknown message blocks bounded by selected ranges, i.e., $0 \le m_i \le M_i$,

$i = 1, 2, \cdots, n$, and we are to find out this unique solution. To do so we will introduce the following probabilistic algorithm based on lattice reduction.

4 Probabilistic Algorithm Based on Lattice Reduction

4.1 Terminology for the New Algorithm

Consider the multivariate modular linear equation derived from the general Lu-Lee type systems

$$f(m) = \sum_{i=1}^{n} k_i m_i \bmod r = c \tag{10}$$

with the unknowns m_i bounded by

$$0 \le m_i \le M_i,\ i = 1, 2, ..., n, \tag{11}$$

where M_i are big integers satisfying $M_i < r$. In this bounded range, there is a unique solution $m^* = (m_1^*, m_2^*, \cdots, m_n^*)$. In order to solve the equation (10) and get this unique solution, we write

$$M_i = r^{\gamma_i}, i = 1, 2, \cdots, n. \tag{12}$$

with $0 < \gamma_i < 1$. Define a $d = n + 2$ dimensional upper triangular matrix

$$M = \begin{pmatrix} v_1 & 0 & \cdots & 0 & 0 & k_1 \\ & v_2 & \cdots & 0 & 0 & k_2 \\ & & \cdots & \cdots & \cdots & \cdots \\ & & & v_n & 0 & k_n \\ & & & & u & c \\ & & & & & r \end{pmatrix} \tag{13}$$

where $v_1, v_2, \cdots, v_n$ and u are all *randomized* rationals, which we denote by

$$u = r^{\lambda} / 2^s, v_i = r^{\delta_i} / 2^s,\ i = 1, 2, \cdots, n. \tag{14}$$

Here the parameters δ_i and λ satisfy

$$\lambda \approx 1 - \sigma,\ \delta_i \approx 1 - \sigma - \gamma_i,\ i = 1, 2, \cdots, n, \tag{15}$$

where

$$\sigma = \sum_{i=1}^{n} \gamma_i \le 1. \tag{16}$$

The value s in (14) is relevant to the dimension of the matrix M, i.e.,

$$s \approx (d \log_2 d)/2. \tag{17}$$

We define a lattice $L(M) \in \mathbb{R}^n$ with the rows of the matrix M. Then the determinant of this lattice is

$$\begin{aligned} \det(L) &= u \cdot r \cdot \prod_{i=1}^{n} v_i \\ &= r^{\lambda+1+\sum_{i=1}^{n}\delta_i} / 2^{(n+1)s} \\ &\approx r^{d(1-\sigma)} \cdot 2^{-(d-1)s} \end{aligned} \tag{18}$$

Note that $L(M)$ is a lattice with rational entries. Multiplying by least common denominator will produce an integer lattice on which basis reduction can be applied.

4.2 Description of the Algorithm

Algorithm:

1) Generate random rationals for v_i and u as defined in (14).
2) Construct a lattice $L(M) \in \mathbb{R}^n$ with the rows of the matrix M in (13).
3) Apply LLL basis reduction algorithm to the lattice $L(M)$, and we get a short vector b_1 of the reduced basis. We rewrite it by $b_1 = W = (w_1, w_2, \cdots, w_d)$.
4) Compute $x = [w_{d-1}/u]$, $\overline{c} = w_d$ and $z = (z_1, \cdots, z_n) \in \mathbb{Z}^n$ by $z_i = [w_i / v_i]$, $i = 1, 2, \cdots, n$, where $[a]$ denotes the integer closest to a.
5) If $\overline{c} = 0$, $x = -1$ and $0 \le z_i \le M_i$ for $i = 1, 2, \cdots, n$, then $z = (z_1, \cdots, z_n)$ is the unique solution to the equation (10) in the given bounded range and the algorithm terminates. Else, repeat steps above.

We first prove the correctness of the algorithm, and give probability analysis in the next section.

Proposition 1. With the denotations above, if the algorithm outputs a short vector $W = (w_1, w_2, \cdots, w_d)$ such that $\overline{c} = 0$, $x = -1$ and $0 \le z_i \le M_i$ for $i = 1, 2, \cdots, n$, then $z = (z_1, z_2, \cdots, z_n)$ is the unique solution to the modular linear equation (10) in the bounded range, i.e., $z = m^*$.

Proof. Since $W = (w_1, w_2, \cdots, w_d)$ is a vector that belongs to the lattice $L(M) \in \mathbb{R}^n$, it can be constructed by the integer linear combination of the rows of the matrix M;

In other words, there is a integer vector A such that $W = AM$. Denote $A = (z_1, z_2, \cdots, z_n, x, y)$, then the vector W can be rewritten by

$$W = \left(z_1 v_1, z_2 v_2, \cdots, z_n v_n, ux, \overline{c}\right) \tag{19}$$

where

$$\overline{c} = \sum_{i=1}^{n} k_i z_i + cx + yr\,. \tag{20}$$

Hence we can compute

$$x = w_{d-1}/u = [w_{d-1}/u],$$
$$z_i = w_i / v_i = [w_i / v_i],\ i = 1, 2, \cdots, n,$$

just as done in the proposed algorithm. With equation (20) and the assumptions that $\overline{c} = 0$, $x = -1$, we have $c = \sum_{i=1}^{n} k_i z_i \bmod r$. And when $0 \le z_i \le M_i$ for $i = 1, 2, \cdots, n$ is also satisfied, $z = (z_1, z_2, \cdots, z_n)$ is the unique solution to the equation (10) in the bounded range. □

5 Probability Analysis

With the denotations above, $m^* = (m_1^*, m_2^*, \cdots, m_n^*)$ is the unique solution to the equation (10) in the given bounded range $0 \le m_i \le M_i$ ($i = 1, 2, \cdots, n$). Consider the vector $W^* = \left(m_1^* v_1, m_2^* v_2, \cdots, m_n^* v_n, -u, 0\right)$, we have the following facts: first, W^* is probably outputted by the above algorithm since its norm is within the bound for that of W; second, W^* is a "good" short vector such that the conditions $\overline{c} = 0,\ x = -1$ *and* $0 \le z_i \le M_i$ are satisfied in step 5 of the algorithm, leading to the unique solution $m^* = (m_1^*, m_2^*, \cdots, m_n^*)$; and finally, when the number of unknowns n is not too large ($n \le 68$), our algorithm will output this "good" vector W^* in polynomial time, with a high probability approximating 1.

Proposition 2. The norm of the vector $W^* = \left(m_1^* v_1, m_2^* v_2, \cdots, m_n^* v_n, -u, 0\right)$ is within the upper bound for W (or b_1, namely) outputted by LLL lattice reduction on $L(M)$, so that our algorithm will probably output it.

Proof. According to Theorem 1, the upper bound for W satisfies

$$\| W \| \le 2^{(d-1)/4} \cdot (\det(L))^{1/d} = 2^{(d-1)/4} \cdot r^{(1-\sigma)} \cdot 2^{-(d-1)s/d} \tag{21}$$

On the other hand, by equations (14) and (15), we have

$$\begin{aligned}\|W^*\|^2 &= \sum_{i=1}^{d} w_i^2 = u^2 + \sum_{i=1}^{n} (m_i^* v_i)^2 \\ &\le (r^{1-\sigma}/2^s)^2 + \sum_{i=1}^{n} (r^{\gamma_i} r^{1-\sigma-\gamma_i}/2^s)^2 \\ &= (d-1)(r^{1-\sigma}/2^s)^2\end{aligned}$$

which yields that

$$\begin{aligned}\|W^*\| &\le \sqrt{d-1}\cdot r^{1-\sigma}/2^s \\ &= \sqrt{d-1}\cdot 2^{-s/d}\cdot r^{1-\sigma}\cdot 2^{-(d-1)s/d}\end{aligned}$$

Note that we choose $s \approx (d\log_2 d)/2$ in (17), so

$$\sqrt{(d-1)}\cdot 2^{-s/d} \approx \sqrt{(d-1)}\cdot 2^{-(\log_2 d)/2} = \sqrt{(d-1)/d} < 1$$

Now we can see by (21) that the norm of W^* is within the upper bound for W :

$$\|W^*\| < r^{1-\sigma}/2^{(d-1)s/d} < 2^{(d-1)/4}\cdot r^{(1-\sigma)}\cdot 2^{-(d-1)s/d}\ .$$

Hence the LLL lattice reduction procedure of the algorithm will probably output W^* as b_1. □

Remarks. The factor $2^{(d-1)/4}$ above should be replaced by 1.02^d according to Heuristic 1, but this will not affect the result here.

Proposition 3. With the denotations above, the outputted vector W^* guarantees that the conditions $\overline{c}=0$, $x=-1$ and $0 \le z_i \le M_i$ ($i=1,2,\cdots,n$) are satisfied, leading to the short solution $m^* = (m_1^*, m_2^*, \cdots, m_n^*)$.

Proof. We have

$$W^* = \left(m_1^* v_1, m_2^* v_2, \cdots, m_n^* v_n, -u, 0\right),$$

and by step 4 of the algorithm we can compute $\overline{c}=0$, $x=-1$ and

$$0 \le z_i = m_i^* \le M_i \ \ (i=1,2,\cdots,n),$$

producing the unique solution $z = m^*$. □

Next we are to analyze the probability with which our algorithm will output the short vector W^* . We first introduce the following function $g(z,x)$ of $z=(z_1,z_2,\cdots,z_n)$ and x :

$$g(z,x) = (cx + \sum_i^n k_i z_i) \bmod r . \tag{22}$$

The module r and coefficients c and k_i are the same as in the original equation (10), and so the outputted result $\overline{c}$, x and $z = (z_1, \cdots, z_n)$ by our algorithm satisfies $g(z,x) = \overline{c}$.

For simplicity we only consider the case that the function $f(m)$ in (10) is uniformly distributed in $[0, r)$, and accordingly $g(z,x)$ is also a uniformly distributed function mapping to $[0, r)$. Let $D(g)$ be the domain and $R(g)$ the range of $g(z,x)$ relevant to the algorithm proposed. Note that, by the definition of $g(z,x)$, $D(g)$ denotes the set of all different (z,x) pairs and $R(g)$ the set of all different $\overline{c}$ (probably) outputted by the algorithm. According to Propositions 2 and 3, W^* is a short vector (probably) outputted by the algorithm, so that $g(m^*, -1) = 0$, thus we get $(m^*, -1) \in D(g)$ and $0 \in R(g)$.

Proposition 4. If the function $g(z,x)$ is uniformly distributed in $[0, r)$, our algorithm has a probability of $1/\ell$ to output the short vector W^* in one round, producing the unique solution $m^* = (m_1^*, \cdots, m_n^*)$. Here $\ell = 1 + \left(1.02^d \cdot 2/\sqrt{d}\right)^d$.

Before proving proposition 4, we first give the following some Lemmas.

Lemma 1. For vector $W = (w_1, w_2, \cdots, w_d)$ and $\| W \|^2 = \sum_{i=1}^d w_i^2$, define $\pi_w = \prod_{i=1}^d | w_i |$, then we have $\pi_w \le \left(\| W \| / \sqrt{d}\right)^d$. The equality holds when $| w_1 | = | w_2 | = \cdots = | w_d |$.

Lemma 2. Let z_i ($i = 1, 2, \cdots, n$), x and $\overline{c}$ be the result outputted by one round of the algorithm, then

$$| \overline{c} \cdot x \cdot \prod_{i=1}^n z_i | \le r \cdot \left(1.02^d / \sqrt{d}\right)^d \tag{23}$$

Lemma 3. Let $g(z,x)$ be a uniformly distributed function of (z,x) mapping to $[0, r)$ with $g(m^*, -1) = 0$, and denote the sizes of the domain $D(g)$ and range $R(g)$ of $g(z,x)$ respectively by D and R. Let X be the number of variable-pairs (z,x) falling in the domain $D(g)$ and mapping to the range $R(g)$, then the expectation of X is approximately $EX = 1 + D \cdot R / r$.

The proof of these lemmas is given in the appendices.

Proof of Proposition 4. By Lemma 3, the expectation of the number of all variable-pairs falling in the domain $D(g)$ and mapping to $R(g)$ is $EX = 1 + D \cdot R / r$, which is just the average number of all different values that the outputted results $z = (z_1, z_2, \cdots, z_n)$, x, $\overline{c}$ may vary in after one round of our algorithm. To prove the proposition, we only need to estimate the value $D \cdot R$, which is the product of the sizes of domain $D(g)$ and range $R(g)$ involved in the algorithm. Note that $D(g)$ is the set of all different (z, x) pairs and $R(g)$ the set of all different $\overline{c}$ (probably) outputted by the algorithm, so $D \cdot R$ is the supremum of the number of all $z = (z_1, z_2, \cdots, z_n)$, x, $\overline{c}$ that satisfy the bound

$$|\overline{c} \cdot x \cdot \prod_{i=1}^{n} z_i | \le r \cdot \left(1.02^d / \sqrt{d}\right)^d$$

by Lemma 2. Thus we have

$$D \cdot R \le 2^d \cdot r \cdot \left(1.02^d / \sqrt{d}\right)^d$$

Here we take into account that for the outputted result z, x and $\overline{c}$, the values of z_i ($i = 1, 2, \cdots, n$), x and $\overline{c}$ may be either positive or negative, hence introducing a $2^{n+2} = 2^d$ factor. Then the number of different values that the outputted result may vary in is

$$\ell = EX = 1 + D \cdot R / r \le 1 + \left(1.02^d \cdot 2 / \sqrt{d}\right)^d$$

Note that the outputted vector W is relevant to z, x and $\overline{c}$ (in step 4 of the algorithm), so that W may also vary in ℓ different values in average. Finally, by Propositions 2 and 3, we know that W^* is one of the outputted vectors, finishing the proof. □

We have the following two corollaries from Proposition 4.

Corollary 4.1. When $10 \le d \le 70$ (or $8 \le n \le 68$, accordingly), $0.9 < 1/\ell < 1$, thus we have a high probability approximating 1 to obtain the unique solution $m^* = (m_1^*, m_2^*, \cdots, m_n^*)$ after one round of our algorithm.

Corollary 4.2. For the cases $2 \le n < 8$, repeating our algorithm for rounds will output the desired vector W^*, and yield the unique solution $m^* = (m_1^*, m_2^*, \cdots, m_n^*)$, with a high probability approximating 1.

Proof. when $2 \le n < 8$ (or $4 \le d < 10$ accordingly), we have $\ell < 2.4$, and hence $1/\ell > 0.4$. This is the probability for obtaining a short solution after one round of the algorithm. Note that we construct the lattice $L(M)$ with randomized entries u and v_i in (13), which will output different vectors W with different u and v_i, so simply by repeating the algorithm for 5 rounds will output a short solution with a high probability larger than $1-(1-1/\ell)^5 > 0.92$. □

Remarks. Repeating the algorithm for rounds when $n > 68$ is insignificant since the probability $1/\ell$ decreases fast with the growth of n. For example, $1/\ell$ is smaller than 0.0001 when $n = 80$.

Proposition 5. When the number of unknowns satisfies $2 \le n \le 68$, the proposed algorithm will output the unique solution $m^* = (m_1^*, m_2^*, \cdots, m_n^*)$ in polynomial time, with a high probability approximating 1.

Proof. It is well known that the LLL lattice reduction algorithm runs in polynomial time (first $O(d^6(\log B)^3)$ in [10], and later improved by floating-point LLL algorithms [11-13]). For $8 \le n \le 68$, our algorithm will output the short solution $m^* = (m_1^*, m_2^*, \cdots, m_n^*)$ in one round with a high probability approximating 1; moreover, for $2 \le n < 8$, repeating the algorithm for rounds (actually no more than 5 rounds) will also reach this target. So for all $2 \le n \le 68$, our algorithm will output the unique solution $m^* = (m_1^*, m_2^*, \cdots, m_n^*)$ in polynomial time, with a high probability approximating 1. □

6 Experimental Results

We have performed experiments for the proposed probabilistic algorithm, using the NTL library [17] for lattice reduction. Experimental results are given in the following Table 1, where η_1 denotes the ratio to find out the solution m^* in one round, and η_{10} the ratio in no more than 10 rounds. We also record the average running time of the algorithm for the cases that we can find out the solution.

The experimental results show that for $n \le 64$, we are likely to find the solution in one round of our algorithm, with a high probability larger than 70%. And for $n \le 70$,

Table 1. Experimental results for cryptanalysis of Lu-Lee type systems

n	p_i (bits)	r (bits)	η_1	η_{10}	average running time
8	512	4096	100%	–	3 sec
16	64	1024	100%	–	450 ms
32	64	2048	100%	–	15 sec
64	32	2048	60%	90%	6 min
70	32	2240	10%	70%	15 min

in most cases (90%) the algorithm succeed in no more than 10 rounds. This is not far from the theoretic analysis of probability in Section 5.

7 Conclusion

In this paper we consider the general Lu-Lee type systems, in which n unknown message blocks are used, and each of the message blocks is bounded by a selected range. Previous results such as cryptanalysis by integer linear programming and CVP (closest vector problem) are not satisfying the need for cryptanalysis of a general Lu-Lee type system. To solve this problem we present a probabilistic algorithm based on lattice reduction and break general Lu-Lee type systems up to 68 message blocks in polynomial time, with a high probability approximating 1.

Cryptanalysis of Lu-Lee type systems with more than 70 message blocks is not satisfying with the probabilistic algorithm by now. Enhancement of the algorithm is for advanced research.

Acknowledgements

This work is supported by the National Key Basic Research and Development (973) of China (No:2003CB314805), the National Natural Sciences Foundation of China (No:90304014), and the Project funded by Basic Research Foundation of School of Information Science and Technology of Tsinghua.

References

1. Lu, S.C., Lee, L.N.: A Simple and Effective Public-Key Cryptosystem. In: COMSAT Technical Review, vol. 9(1), pp. 16–23. Springer, Heidelberg (1979)
2. Adleman, L., Rivest, R.: How to break the Lu-Lee Public-Key Cryptosystem, MIT Laboratory for Computer Science Technical, July 24, 1-9 (1979)
3. Goethals, J.M., Couvreur, C.: A Cryptanalytic Attack on the Lu-Lee Public-Key Cryptosystem. Philips J. Res. 35, 301–306 (1980)
4. Kochanski, M.J.: Remarks on Lu and Lee's Proposals. Cryptologia 4(4) (1980)
5. Adiga, B.S., Shankar, P.: Modified Lu-Lee Cryptosystem. Electronics Letters, 21(18), 794–795 (1985)
6. Duan, L.X., Nian, C.C.: Modified Lu-Lee Cryptosystem. Electronics Letters 25(13), 826 (1989)
7. Xing, L.D., Sheng, L.G.: Cryptanalysis of New Modified Lu-Lee Cryptosystems. Electronics Letters 26(3), 1601–1602 (1990)
8. Brickell, E.F., Odlyzko, A.M.: Cryptanalysis: A survey of recent results. Proceedings of the IEEE 76, 578–592 (1988)
9. Kannan, R.: Improved Algorithm for Integer Programming and Related Lattice Problems. In: Proc. 15th ACM Symposium on Theory of Computing, pp. 193–206 (1983)
10. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. 261, 513–534 (1982)
11. Schnorr, C.P.: Fast LLL-type lattice reduction. Unpublished draft available at (October, 2004), `http://www.mi.informatik.uni-frankfurt.de/research/papers.html`

12. Nguyen, P.Q., Stehlé, D.: Floating-Point LLL Revisited. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 215–233. Springer, Heidelberg (2005)
13. Nguyen, P.Q., Stehlé, D.: LLL on the Average. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 238–256. Springer, Heidelberg (2006)
14. Nguyen, P.Q., Stern, J.: The Two Faces of Lattices in Cryptology. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 146–180. Springer, Heidelberg (2001)
15. Babai, L.: On Lovász lattice reduction and the nearest lattice point problem. Combinatorica 6, 1–13 (1986)
16. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proc. 33rd STOC, pp. 601–610. ACM, New York (2001)
17. Shoup, V.: NTL. Number Theory C++ Library, `http://www.shoup.net/ntl/`

Appendices

Proof of Lemma 1. Obviously when $|w_1| = \cdots = |w_d|$, $\pi_w = (\|W\|/\sqrt{d})^d$ is satisfied. If there exists some w_i and w_j such that $|w_i| \neq |w_j|$, we are able to raise the value π_w, simply by replacing w_i and w_j by w_i^* and w_j^*, where

$$|w_i^*| = |w_j^*| = \sqrt{(w_i^2 + w_j^2)/2}\ .$$

In this case $w_i^{*2} + w_j^{*2} = w_i^2 + w_j^2$, and $\|W\|^2$ remains the same for the new w_i^* and w_j^*; while on the other hand, the new value of π_w is raised since

$$|w_i^* w_j^*| = (w_i^{*2} + w_j^{*2})/2 = (w_i^2 + w_j^2)/2 > |w_i w_j|\ .$$

Repeating this replacement so far as there exists $|w_i| \neq |w_j|$, we will reach the maximum value of $\pi_w = \left(\|W\|/\sqrt{d}\right)^d$ when $|w_1| = |w_2| = \cdots = |w_d|$. □

Proof of Lemma 2. According to Heuristic 1 and equation (18), the vector W (alias for the first vector b_1 of the reduced basis) satisfies

$$\|W\| \approx 1.02^d \cdot (\det(L))^{1/d} = 1.02^d \cdot r^{(1-\sigma)} \cdot 2^{-(d-1)s/d} \tag{24}$$

Equations (19) and (14) show that

$$\begin{aligned}\prod\nolimits_{i=1}^{d} |w_i| &= \left(\prod\nolimits_{i=1}^{n} |z_i v_i|\right) \cdot |ux| \cdot |\overline{c}| \\ &= \left(\prod\nolimits_{i=1}^{n} |z_i|\right) \cdot |x| \cdot |\overline{c}| \cdot \left(\prod\nolimits_{i=1}^{n} v_i\right) \cdot u \\ &= \left(\prod\nolimits_{i=1}^{n} |z_i|\right) \cdot |x| \cdot |\overline{c}| \cdot r^{d(1-\sigma)-1} \cdot 2^{-(d-1)s}\end{aligned} \tag{25}$$

By Lemma 1 and equation (24) we have

$$\begin{aligned}\prod_{i=1}^{d} |w_i| &\le \left(\|W\| / \sqrt{d}\right)^d \\ &\approx \left(1.02^d \cdot r^{(1-\sigma)} \cdot 2^{-(d-1)s/d} / \sqrt{d}\right)^d \\ &= \left(1.02^d / \sqrt{d}\right)^d \cdot r^{d(1-\sigma)} \cdot 2^{-(d-1)s}\end{aligned}$$

This leads to the equation (23) together with (25). □

Proof of Lemma 3. Because $(m^*, -1) \in D(g)$ and $0 \in R(g)$, we only consider the number of variable-pairs differing from $(m^*, -1)$, denoted by Y with $X = Y + 1$. Since $g(z, x)$ is uniformly distributed in $[0, r)$, for a random variable-pair $(z, x) \in D(g)$, $g(z, x)$ will map to $R(g)$ with a probability of R / r (and not map to $R(g)$ with a probability of $1 - R / r$), then for the $D - 1$ variable-pairs in $D(g)$ except $(m^*, -1)$, the probability for i variable-pairs (z, x) mapping to $R(g)$ (and the other $D - 1 - i$ variable-pairs not mapping to $R(g)$) is

$$(1 - R / r)^{D-1-i} \cdot (R / r)^i .$$

The i variable-pairs (z, x) are randomly chosen, so totally C_i^{D-1} different choices are to be considered. Then for $i = 0, 1, \cdots, D - 1$,

$$P(Y = i) = C_i^{D-1} \cdot (1 - R / r)^{D-1-i} \cdot (R / r)^i ,$$

where $P(Y = i)$ denotes the probability for $Y = i$. This is a binomial distribution of Y : $B(Y;\, D - 1,\, R / r)$, so we have $EY = (D - 1) \cdot R / r \approx D \cdot R / r$, and $EX = EY + 1 \approx 1 + D \cdot R / r$. □

A Timing-Resistant Elliptic Curve Backdoor in RSA

Adam L. Young[1] and Moti Yung[2]

[1] Cryptovirology Labs
aly@cryptovirology.com
[2] Columbia University
moti@cs.columbia.edu

Abstract. We present a fast algorithm for finding pairs of backdoor RSA primes (p, q) given a security parameter. Such pairs posses an *asymmetric backdoor* that gives the designer the exclusive ability to factor $n = pq$, even when the key generation algorithm is public. Our algorithm uses a pair of twisted curves over $\mathrm{GF}(2^{257})$ and we present the first incremental search method to generate such primes. The search causes the $\frac{1}{2}\log(n)+\mathrm{O}(\log(\log(n)))$ least significant bits of n to be modified during key generation after p is selected and before q is determined. However, we show that this is tolerable by using point compression and ECDH. We also present the first rigorous experimental benchmarks of an RSA asymmetric backdoor and show that our OpenSSL-based implementation outperforms OpenSSL RSA key generation. Our application is highly efficient key recovery. Of independent interest, we motivate the need to find large binary twists. We present the twist we generated and how we found it.

Keywords: Twisted elliptic curves, RSA, subliminal channel, kleptography.

1 Introduction

The problem of devising a backdoor within RSA [27] public keys is a well-studied area. It is an important topic since RSA key generation machinery is often implemented in hardware or binary executables. Therefore, the extent to which such implementations can/should be trusted is not always clear and considerable effort may be needed to access the design of the machinery that has been deployed. Without expending such effort, a backdoor may go unnoticed for years. This threat provides a motivation for the study of backdoors.

The approaches to designing backdoors can be divided into two categories: symmetric backdoors and asymmetric backdoors. A symmetric backdoor is suitable when one may assume that the key generation device is a tamper-proof black-box. However, a reverse-engineer that breaches the black-box learns the secrets and is able to use the backdoor in like implementations. On the other hand, an asymmetric backdoor only assumes that the key generator is implemented in a tamper-resistant black-box. The reverse-engineer that breaches the black-box will learn that a backdoor is present but will be unable to use it. The asymmetric backdoor constructions to date have all utilized asymmetric cryptography to construct the asymmetric backdoor (public key of the designer).

Anderson presented a symmetric backdoor [3] and it was subsequently cryptanalyzed by Kaliski [17]. The notion of an asymmetric backdoor was presented in [32] (see also [33]) where the backdoor was in turn an RSA public key. The construction is called a secretly embedded trapdoor with universal protection (SETUP) and the trapdoor is the public key of the malicious designer. A problem in the design is that the secretly embedded trapdoor has a security parameter that is 1/2 that of the key being generated. So, when a 1024-bit RSA key is being generated the secretly embedded trapdoor must be about 512 bits which is unacceptable today. A solution to this problem was recently presented in [34]. It relies on the random oracle model, utilizes a twisted pair of curves over GF(2^m), and has the property that the security parameter of the backdoor is m. The space-efficiency is achieved using point compression and the backdoor works for 1024-bit RSA. All of these works permitted the RSA public exponent e to be small and fixed for all users.

The possibility of detecting a backdoor based on the running time of a black-box key generator was presented in [19]. This is related to the notion of a timing attack [18]. The issue of detection based on running time was mentioned in [7] that presents several constructions for *symmetric* backdoors. However, many of the proposed constructions do not allow e to be fixed among all users (crypto APIs commonly take *e as an input* to the RSA key generation function). The goal was to achieve a fast running time to evade detection, yet no proof of security nor benchmarks were provided. So, it appears that to date *none* of the works on constructing a backdoor in RSA key generation present a provably timing resistant construction nor experimental benchmarks. Our contributions are:

1. We present the first timing-resistant asymmetric backdoor in RSA. The backdoor is based on [34] but is *substantially* redesigned.
2. We present the first *benchmarks* of the running time of the backdoor.
3. We present a backdoor (that we built into OpenSSL) that has a running time that is even *faster* than normal OpenSSL RSA key generation and show that the particulars of the prime incremental search method/trial-division can have a very significant impact on timing-resistance.

Assessing timing-resistance could presumably involve accounting for a multitude of different reference (i.e., honest) RSA key generators, the processors used, and a comparison with the backdoor(s). However, this rather unwieldy challenge is not what we consider here. Rather, we consider only one RSA key generation algorithm, a key generator that is present in the OpenSSL library. We believe that this case is common enough to merit study. So, our timing results are with respect to this commonly accepted method/implementation.

We conducted a methodological investigation where we treated the OpenSSL RSA key generator as our "engineering challenge". Our challenge was to use an "algorithms engineering approach" in order to arrive at a backdoored RSA key generator that runs faster than that of the engineering challenge (OpenSSL).

The idea is that "algorithms engineering" eventually gains from a better understanding of the theory, a better and more careful look at the implementation and its improvement (e.g., implementation tricks) and also new methods of achieving the goals and associated cryptographic issues that relate to them.

But, we believe that it would be unfair to categorize this paper as simply a collection of "tricks" beyond [34]. Our timing results were difficult to achieve. We succeeded only after applying several algorithmic speed-ups and in the end finding that we had to change the incremental search/fast trial-division algorithm to beat the OpenSSL RSA key generation running time. Some of the differences are that [34] *does not*: use incremental search, use fast trial division,[1] use wNAF splitting, choose ECDH exponents to minimize computational cost.[2]

2 Background and Definition

We now present a working definition of a secure SETUP for RSA. It is based on [34] but is modified to accommodate the possibility of timing analysis.

The model involves a designer, an eavesdropper, and an inquirer. The designer builds a black-box A that contains the SETUP and a black-box B that conducts normal (unescrowed) RSA key generation. The designer is given access to the public keys that are generated and the goal of the designer is to obtain the RSA private key of a user who uses A to generate a key pair. Prior to the start of the games, the eavesdropper and inquirer are given access to the SETUP algorithm and the normal key generation algorithm. However, once the games start they are not given access to the internals of A nor B.

It is assumed that the eavesdropper and inquirer are probabilistic poly-time algorithms and that the RSA key generation algorithms are deployed in tamper-resistant black-boxes. We distinguish between delayed oracle access (e.g., once every hour on the hour) and timely oracle access in order to account for polynomial indistinguishability and timing analysis.

Game 1: Select $T \in_R \{A, B\}$ and let the inquirer have delayed oracle access to T. The inquirer wins if he correctly determines whether or not $T = A$ with probability significantly greater than 1/2.

Property 1: (computational indistinguishability) The inquirer fails Game 1 with overwhelming probability (w.o.p.).

Game 2: The eavesdropper may query A but is only given the public keys that result, not the corresponding private keys. He wins if he can learn one of the corresponding private keys.

Property 2: (confidentiality) The eavesdropper fails Game 2 (w.o.p.).

Property 3: (completeness) Let (y, x) be a public/private key generated using A. The designer computes x on input y w.o.p.

[1] In fact, it does not do trial division at all.

[2] [34] requires *point-halving* during key generation, we show how to eliminate this.

In a SETUP, the designer uses his or her own private key(s) in conjunction with y to recover x. For example, y may be present in an X.509 v3 certificate.

Property 4: (uniformity) The SETUP is the same in every black-box device A that is manufactured.

Game 3: Select $T \in_R \{A, B\}$ and let the inquirer have timely oracle access to T. The inquirer wins if he correctly determines whether or not $T = A$ with probability significantly greater than 1/2.

Property 5: (timing indistinguishability) The inquirer fails Game 3 w.o.p.

Property 5 clearly subsumes Property 1. The reason we allowed this is to permit applications in which timely oracle queries may be unlikely (in which case Property 5 may be ignored). We leave as open the issue of extending the definition to account for power analysis and fault analysis.

Property 4 implies that there are no unique identifier strings in the SETUP device. This permits distribution in a compiled program in which all instances of the program are identical without diminishing security. Also, this makes it simpler to manufacture the SETUP in hardware and software.

Definition 1. *If an RSA key generation algorithm satisfies properties 1, 2, 3, 4, and 5 then it is a* **Timing-Resilient SETUP**.

We state up-front that we have no formal proof that our proposed SETUP algorithm achieves Definition 1. The present work is a solid step in this direction since our algorithm runs *faster* than normal OpenSSL RSA key generation.

The probabilistic bias removal method (PBRM) was put forth in [33]. The PBRM produces asymmetric ciphertexts that are computationally indistinguishable from random bit strings. PBRM was employed in [1] to devise a public key stegosystem (PKS). It was shown how to use twists [14,15,16] to devise a PKS [23].

For typical elliptic curves used in cryptography, only about half of $\mathbb{F}_q$ corresponds to the x-coordinates on the given curve. By using two curves where one is the twist of the other, it is possible to implement a trapdoor one-way permutation from $\mathbb{F}_q$ onto itself. The notion of a twist has been used to implement trapdoor one-way permutations [16].

3 System Setup

The key recovery algorithm utilizes two binary curves. Each curve $E_{a,b}$ is defined by the Weierstrass equation $y^2 + xy = x^3 + ax^2 + b$. The coefficients a and b are in $\mathbb{F}_{2^m}$ and $b \neq 0$. We employ the standard group operations for binary EC cryptosystems. By making m an odd prime we avoid the Gaudry-Hess-Smart attack [12]. It is required that these curves provide a suitable setting for the elliptic curve decision Diffie-Hellman problem (ECDDH). It is well-known that for certain elliptic curves, DDH is tractable [13].

The value for m is the prime 257. The irreducible trinomial is $f(z) = z^{257} + z^{12} + 1$. The implementation uses $p = 2^{257} + 2^{12} + 1$ to express this trinomial. The number of points on $E_{0,b}$ is $4q_0$. The number of points on $E_{1,b}$ is $2q_1$. The values for b, q_0, and q_1 that we used are as follows, expressed in hexadecimal.

$b =$ `197D4C3C909B4C8EAC18BB296C11BFB18C80B37C0C62AFD8E5F00104C46EEAF0B`

$q_0 =$ `80000000000000000000000000000005EB3E3179500E2B5D2F8EA6DCC363C1F`

$q_1 =$ `FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF429839D0D5FE3A945A0E2B24679387C3`

We will now review the basics of twisted binary curves. Hasse's inequality implies that $|\#E_{a,b}(\mathbb{F}_{2^m}) - 2^m - 1| < 2 * 2^{m/2}$. Also, if the trace $\mathrm{Tr}_{\mathbb{F}_{2^m}/\mathbb{F}_2}(a) \neq \mathrm{Tr}_{\mathbb{F}_{2^m}/\mathbb{F}_2}(a')$ then $E_{a,b}$ and $E_{a',b}$ are referred to as *twists* of one another. For two such twists, for every $x \in \mathbb{F}_{2^m}$ there exists a $y \in \mathbb{F}_{2^m}$ such that (x, y) is on $E_{a,b}$ or $E_{a',b}$. Furthermore, there are two possibilities. Either $(x, x+y)$ and (x, y) are points on the same curve or $(x, y) = (0, \sqrt{b})$ is on both of the curves. The total number of points on both curves is $\#E_{a,b}(\mathbb{F}_{2^m}) + \#E_{a',b}(\mathbb{F}_{2^m})$ which is $2^{m+1}+2$. Therefore, from Hasse's inequality, $\#E_{a,b}(\mathbb{F}_{2^m}) \approx \#E_{a',b}(\mathbb{F}_{2^m}) \approx 2^m$.

Since m is an odd integer, $\mathrm{Tr}_{\mathbb{F}_{2^m}/\mathbb{F}_2}(0) = 0$ and $\mathrm{Tr}_{\mathbb{F}_{2^m}/\mathbb{F}_2}(1) = 1$. So, $E_{0,b}$ and $E_{1,b}$ are a pair of twisted curves. It is important to choose curves that resist known cryptanalytic attacks (e.g., that satisfy the MOV condition). Using point-counting techniques it is known how to efficiently generate $E_{0,b}$ and $E_{1,b}$ with orders $4q_0$ and $2q_1$, respectively, where the values q_0 and q_1 are prime. The curve $E_{0,b}$ will have a cofactor of 4 and $E_{1,b}$ will have a cofactor of 2.

4 Fast SETUP Algorithm

For ease of exposition we will describe the algorithm such that it requires that $|n| \bmod 8 = 0$. Here $|n|$ denotes the length in bits of n represented as a binary integer. So, we assume that n fills an integral number of bytes completely. The actual implementation relaxes this to require that n be even. Naturally there are other elements of the implementation that we do not reflect here. For example, in many places sensitive values are zeroized in memory.

Once the twist is found, a base point G_0 having order $4q_0$ that is on $E_{0,b}(\mathbb{F}_{2^m})$ is generated and a base point G_1 having order $2q_1$ that is on $E_{1,b}(\mathbb{F}_{2^m})$ is generated. The designer generates the private key $x_0 \in_R \{1, 2, ..., q_0 - 1\}$ and the public key $Y_0 = x_0 4G_0$. The designer also generates the value $x_1 \in_R \{1, 2, .., q_1 - 1\}$ and the public key $Y_1 = x_1 2G_1$. The precomputed wNAF splitting values [22] for (G_0, Y_0, G_1, Y_1) are stored in the RSA key generation device.

4.1 Building Blocks for RSA SETUP

Following [34] the SETUP algorithm utilizes point compression. The compressed public point in an EC Diffie-Hellman key exchange is embedded in the upper order bits of the RSA modulus using a well-known channel in composites (see [32]). A point (x, y) on the curve over $\mathbb{F}_{2^m}$ can be uniquely expressed by using $m+1$ bits [30]. We denote the compressed point by $(x, ybit)$ where $ybit \in \{0, 1\}$.

NewCurve takes as input the EC parameters p, a, and b and returns `group`, a data structure that is used in basic elliptic curve operations. Such operations include point compression, fast scalar point multiplication, and so on. The data structure can also be loaded with wNAF splitting precomputations.

PointCompress(group,P) compresses the point $P = (U, V)$ on the curve defined by the `group` data structure. The algorithm outputs $(x \mid\mid ybit)$ which is the compressed representation of (U, V). PointDecompress(a, $x \mid\mid ybit$) decompresses $(x \mid\mid ybit)$ that is on twist a $\in \{0, 1\}$ and outputs $((U, V), w)$. If $w = 1$ then (U, V) is the decompressed point on curve a. If $w = 0$ then an error occurred since $(x, ybit)$ is not on curve a. In this case (U, V) is undefined.

The algorithm GenDHExchangeValues generates the public Diffie-Hellman key exchange parameter and the Diffie-Hellman shared secret. It effectively conducts an ECDH key exchange between the key generation device and it's designer. In short, the shared secret is employed to generate one of the RSA primes and the public Diffie-Hellman value is displayed in the bit representation of n. The designer need only obtain n (e.g., from a Certification Authority) to recover the DH exchange value and factor n.

Both groupValsG$_i$ and groupValsY$_i$ specify such things as blocksize, numblocks, and w for wNAF splitting. Both groupPointsG$_i$ and groupPointsY$_i$ point to 2 dimensional byte arrays that store the precomputed points for wNAF splitting. groupValsG$_i$ and groupPointsG$_1$ store the precomputations for G$_i$. groupValsY$_i$ and groupPointsY$_i$ store the precomputations for Y$_i$. LoadwNAFsplit loads a group data structure with the precomputed values for G_i (or Y_i).

The SetGenerator algorithm takes as input a group data structure, a generator G, the order of G, and the cofactor cof. The algorithm configures the group data structure to use G as the base point in calls to FastPointMul.

PointMul takes as input a group structure and K and performs wNAF point multiplication. It returns $P = KG$ where G is configured using SetGenerator. FastPointMul is the same except that it performs point multiplication using wNAF splitting [22]. FastKmodr uses the precomputations $(2q_0, 3q_0, 4q_0, 2q_1)$ to compute $k = K \bmod r$ quickly. It uses a binary search that costs at most 2 multiprecision comparisons and 1 subtraction.

```
GenDHExchangeValues():
Input: none
Output: $s_{pub}, s_{priv} \in \{0,1\}^{m+1}$
1. with probability $\frac{4q_0-1}{2^{m+1}}$ set $a = 0$ and with probability $\frac{2q_1-1}{2^{m+1}}$ set $a = 1$
2. if $a = 0$ set (r, cof) = ($q_0$, 4) else set (r, cof) = ($q_1$, 2)
3. set groupg = NewCurve($p, a, b$) and set groupy = NewCurve($p, a, b$)
4. if $a = 1$ then
5.       LoadwNAFsplit(groupg, groupValsG$_1$, groupPointsG$_1$)
6.       LoadwNAFsplit(groupy, groupValsY$_1$, groupPointsY$_1$)
7. else
8.       LoadwNAFsplit(groupg, groupValsG$_0$, groupPointsG$_0$)
9.       LoadwNAFsplit(groupy, groupValsY$_0$, groupPointsY$_0$)
10. SetGenerator(groupg, $G_a$, 2(2-a)r, cof)
```

```
11. SetGenerator(groupy, $Y_a$, r, cof)
12. choose $K$ uniformly at random such that $0 < K < 2(2-a)r$
13. set $k = K$
14. if $(K \geq r)$ then set $k = \text{FastKmodr}(K, r)$
15. $P$ = FastPointMul(groupg, $K$)   /* compute $P = KG_a$ */
16. $s_{pub}$ = PointCompress(groupg, $P$)
17. $P$ = FastPointMul(groupy, $k$)   /* compute $P = kY_a$ */
18. $s_{priv}$ = PointCompress(groupg, $P$)
19. return $(s_{pub}, s_{priv})$
```

The "public" DH key exchange value is s_{pub}. The shared secret is s_{priv}. As shown in [34,23], this randomized method of selecting a curve and computing s_{pub} causes s_{pub} to be indistinguishable from a random $m + 1$-bit string. This forms the basis for Property 1.

The PointDouble algorithm takes as input a group data structure and P and returns the point $Q = P + P$. RecoverDHSharedSecret recovers s_{priv}.

```
RecoverDHSharedSecret($s_{pub}, x_0, x_1$):
Input: $s_{pub} \in \{0,1\}^{m+1}$ and EC private keys $x_0, x_1$
Output: $s_{priv} \in \{0,1\}^{m+1}$
1. set (v, r, cof) = (0, $q_0$, 4)
2. $((U,V),w)$ = PointDecompress(0, $s_{pub}$)
3. if $(w = 0)$ then
4.     compute $((U,V),w)$ = PointDecompress(1, $s_{pub}$)
5.     set (v, r, cof) = (1, $q_1$, 2)
6. set $a = v$ and set group = NewCurve($p, a, b$)
7. let $P_1$ be the point corresponding to $(U,V)$
8. $P_1$ = PointDouble(group, $P_1$)
9. if $(v = 0)$ then set $P_1$ = PointDouble(group, $P_1$)
10. SetGenerator(group, $P_1$, r, cof)
11. $P_2$ = PointMul(group, $x_a$)   /* compute $P_2 = x_a P_1$ */
12. return $s_{priv}$ = PointCompress(group, $P_2$)
```

IsAcceptablePrimeFast decides acceptable RSA primes. PrimeChecksForSize takes as input the bit length of a candidate prime and returns a positive integer specifying the number of iterations of Miller-Rabin [20,26]. This value is based on the average case error estimates for the strong probable prime test [9]. PrimeChecksForSize is defined by Table 4.4 in [21] and on input 1024, 2048, and 4096 it returns 3, 2, and 2, respectively. IsPrimeFastTest(p,t) returns the Boolean result of t iterations of Miller-Rabin on p.

```
IsAcceptablePrimeFast(e, len, $p_1$):
Input: RSA exponent e, required bit length len of $p_1$, candidate prime $p_1$
Output: true or false, false if $p_1$ is not an acceptable prime
1. if $(|p_1| \neq \text{len})$ then halt with false
2. if the two uppermost bits of $p_1$ are not $11_2$ then halt with false
3. numchecks = PrimeChecksForSize(len)
```

```
4. result = IsPrimeFastTest($p_1$, numchecks)
5. if (result = false) then halt with false
6. if ($\gcd(p_1 - 1, e) \neq 1$) return false else return true
```

A random oracle [5] can be used to derive q based on the DH shared secret. This approach leads to pairs of backdoor primes that are computationally indistinguishable from normal RSA primes under ECDDH [34]. Given that we want a fast running time, we instantiate the oracle using a fast PRNG. ANSI X9.17 [4] was chosen for this purpose, which is clearly not a random oracle (that cannot exist), but that seems to satisfy our needs in practice. However, we modified the ANSI X9.17 PRNG to use AES-128 [8] instead of DES.

The algorithm DHSecretToPRNGParams takes as input the point s_{priv} and returns (seed,key,D). Recall that $|s_{priv}| = m + 1$ bits. In our case $m = 257$. The algorithm right shifts s_{priv} by 2 bits to derive 256 bits for use in the PRNG. 128 of these bits are used for `key` and the remaining 128 bits are used for `seed`. The date value D is a fixed 128-bit value that was chosen uniformly at random.

PseudoRndGenAES128$_1$(seed,key,D) initializes the PRNG (D is the ‘date’ [4]). PseudoRndGenAES128$_0$(t) returns the next t bytes from the PRNG stream. PseudoRndGenAES128$_2$ zeroizes values and frees memory. These calls maintain state information across calls.

The constant `NUMPRIMES` is 2048 and the array `primes` stores the first 2048 primes, from `primes[0] = 2` up to `primes[NUMPRIMES − 1] = 17863`, inclusive. The array `mods` has the same number of elements as `primes`. We define `MAXDELTA` to be $2^{32} - 1 -$ `primes[NUMPRIMES − 1]`. This constant is used as an upper limit on `delta` to ensure that the 32-bit word (`mods[i] + delta`) does not overflow.

```
GenPrimeWithPRNGFast(seed, key, D, len, e):
Input: seed, key, D $\in \{0,1\}^{128}$, required bit length len, RSA exponent e.
Output: An acceptable prime $p_1$
1.  PseudoRndGenAES128$_1$(seed, key, D)
2.  $p_1$ = PseudoRndGenAES128$_0$(len/8)
3.  set the 2 most significant bits and the least significant bit of $p_1$ to 1
4.  for $i = 1$ to NUMPRIMES − 1 step 1 do:
5.      mods[i] = $p_1$ mod primes[i]
6.  delta = 0
7.  for $i = 1$ to NUMPRIMES − 1 step 1 do:
8.      if ((mods[i] + delta) mod primes[i] $\leq$ 1)
9.          delta = delta + 2
10.         if (delta > MAXDELTA) then goto step 2
11.         goto step 7
12. if (IsAcceptablePrimeFast(e, len, prime + delta) = false)
13.     delta = delta + 2
14.     if (delta > MAXDELTA) then goto step 2
15.     goto step 7
16. PseudoRndGenAES128$_2$()
17. set $p_1$ = $p_1$+ delta and return $p_1$
```

The prime generator GenPrimeWithPRNGFast uses incremental search. It also uses a very fast trial-division algorithm that is based on [21,24]. In [21] an array-update method is given. This method is effectively as follows. The mods array is iteratively updated by computing mods[i] = mods[i] + 2 mod primes[i] for $i = 1$ to NUMPRIMES − 1. The entire mods array is continually updated until an iteration occurs in which none of the resulting mods[i] values is 0.

The algorithm above differs from the array-update method since it does not iteratively increment entries in the mods array. It increments delta instead. In the event that (mods[i]+delta) mod primes[i] ≤ 1 it terminates the iteration altogether, so it is faster than [21]. The mods array is changed only when a new candidate prime p_1 is chosen.

However, by checking that (mods[i]+delta) mod primes[i] ≤ 1 an additional restriction is imposed on p_1. It is guaranteed that the resulting integer p_1 will be such that $\mathrm{p}_1 - 1$ is not divisible by any of the primes in the primes array (except 2).[3] Observe that IsAcceptablePrimeFast does not need to do trial division since trial division was already applied to the value passed into p_1.

The final building block is the algorithm GenPrimeWithOracleIncr that takes as input (s_{priv},len,e) and that returns an acceptable prime p_1. This algorithm calls DHSecretToPRNGParams(s_{priv}) that returns the triple (seed, key, D) that is needed for the PRNG. The algorithm then returns the acceptable RSA prime number p_1 where p_1 = GenPrimeWithPRNGFast(seed, key, D, len, e).

4.2 Fast RSA SETUP Key Generation

The following is the main algorithm that produces the backdoor RSA primes.

```
GetPrimesFast(bits, e, $s_{pub}$, $s_{priv}$):
Input: bits is the desired modulus length, exponent e, $s_{pub}, s_{priv} \in \{0,1\}^{m+1}$
Output: A pair of acceptable RSA primes $(p_1, q_1)$
1. len = bits/2
2. $\mathrm{p}_1$ = GenPrimeWithOracleIncr($s_{priv}$, len, e)
3. $\mu$ = bits − (8 + m + 1)
4. choose $r_1 \in_R \{0,1\}^7$ and $r_2 \in_R \{0,1\}^{\mu}$
5. set $n_c = 1 \mid\mid r_1 \mid\mid s_{pub} \mid\mid r_2$
6. solve for $(q_1, r)$ in $n_c = q_1 p_1 + r$
7. if ($|q_1| \neq$ len or the 2nd most significant bit of $q_1$ is not 1) then goto step 4
8. set the least significant bit of $q_1$ to 1
9. for $i = 1$ to NUMPRIMES − 1 step 1 do:
10.     mods[i] = $\mathrm{q}_1$ mod primes[i]
11. delta = 0
12. for $i = 1$ to NUMPRIMES − 1 step 1 do:
13.     if ((mods[i] + delta) mod primes[i] $\leq 1$)
14.         delta = delta + 2
```

[3] This algorithm is related to the way OpenSSL 0.9.8 generates RSA primes. However, we added a crucial speed-up that we cover in Section 5 that causes the SETUP to outperform the running time of OpenSSL RSA key generation.

```
15.         if (delta > MAXDELTA) then goto step 4
16.         goto step 12
17. if (IsAcceptablePrimeFast(e, len, q_1 + delta) = false)
18.     delta = delta + 2
19.     if (delta > MAXDELTA) then goto step 4
20.     goto step 12
21. set q_1 = q_1 + delta and return (p_1, q_1)
```

When overflow occurs p$_1$ is not fully rerandomized due to s_{pub}. We found that by making delta a multiprecision variable (i.e., much larger) and eliminating overflow, the running time became far too slow. So, in practice this decision seems to be the best one.

The algorithm KleptoKeyGenFast takes as input the integer `bits` and the RSA exponent e and returns the final values of p$_1$ and q$_1$. Let these final values be denoted by $p =$ p$_1$ and $q =$ q$_1$. The algorithm first calls GenDHParamAndDHSecret to generate the s_{pub} and s_{priv} Diffie-Hellman key exchange values. It then passes these to GetPrimesFast along with `bits` and e. It returns the pair of primes that GetPrimesFast halts with.

The algorithm RSAGenerateKeyEx generates the backdoor RSA primes. It takes as input $(\texttt{bits}, e)$ and uses KleptoKeyGenFast to generate the primes only when $\texttt{bits} \geq 1024$ and $2 \mid \texttt{bits}$. Otherwise it performs normal (unescrowed) RSA key generation. The OpenSSL 0.9.8 version of RSAGenerateKeyEx generates at most 3 primes q and if all of them equal p then it reports an error. This is designed to gracefully handle small (p, q). Since $p = q$ occurs with negligible probability for our backdoor keys we omit this restriction.

RSAGenerateKeyEx returns $(e, d, p, q, dP, dQ, qInv)$. These values can be found in the PKCS #1 standard [25] and they enable a fast version of the Chinese Remainder Theorem to be used for RSA inversion.

$$ed \equiv 1 \bmod \lambda(n),\; q * qInv \equiv 1 \bmod p,\; e * dP \equiv 1 \bmod p-1,\; e * dQ \equiv 1 \bmod q-1$$

4.3 RSA SETUP Key Recovery

The designer obtains the RSA modulus n and RSA exponent e of the user. The following algorithm permits n to be factored.

```
KleptoRecoveryKeyFast($n$, e, $x_0$, $x_1$):
Input:  RSA modulus $n$, RSA exponent e, and EC private keys $(x_0, x_1)$
Output: (p_1,v) where v ∈ {true, false}. v is true ⇔ p_1 | $n$
1. bits = |n|
2. $\mu$ = bits − $(8 + m + 1)$
3. let $t_0$ be $n$ right shifted $\mu$ bits  /* throws away the $\mu$ rightmost bits */
4. set $s_{pub} = t_0 \bmod 2^{m+1}$
5. $s_{priv}$ = RecoverDHSecret($s_{pub}$, $x_0$, $x_1$)
6. p_1 = GenPrimeWithOracleIncr($s_{priv}$, bits/2, e)
7. if (n mod p_1 = 0) then halt with (p_1, true) else halt with (p_1, false)
```

A result of v = false implies that some form of unexpected error occurred. It is possible yet overwhelmingly unlikely that key recovery will fail due to a borrow bit being taken. To see this, note the following. The candidate composite is $n_c = q_1 p_1 + r$ for some remainder r. When q_1 is incremented by 2, $n_c + 2p_1 = (q_1 + 2)p_1 + r$. Due to the Prime Number Theorem, about $O(\log(p_1))$ increments will be performed until an acceptable prime (q_1 + delta) is found. The subtraction of $2p_1$ from both sides may result in a borrow bit that threatens the representation of s_{pub} in the upper order bits of n_c. The chances that this representation is altered is negligible for the backdoor keys that are generated. Security is addressed in Appendix A.

The following is a variant that uses Coppersmith's factoring algorithm. Coppersmith showed that if the $|n|/4$ most significant bits of p_1 are known then $n = \mathrm{p}_1\mathrm{q}_1$ can be efficiently factored [6]. Following [7,34], this cryptanalytic method can be exploited to lower the number of pseudorandom bits that are used to construct p_1. The oracle (that we instantiate using a PRNG) can be used to generate the $|\mathrm{p}_1|/2$ most significant bits of p_1 (except the 2 most significant bits that are 1). The remaining lower order bits are selected randomly. The value p_1 is subjected to trial-division and is tested for being an acceptable RSA prime. The next $|\mathrm{p}_1|/2$ bits are taken from the PRNG stream as needed, and so on. This repeats until an acceptable prime p_1 is found. To factor a backdoored n, Coppersmith's algorithm $\mathcal{C}$ is used. The PRNG bit stream is regenerated $|\mathrm{p}_1|/2$ bits at a time. Each of these strings is given to $\mathcal{C}$ along with n and one will lead to the factorization of n.

5 Performance

We first searched for a twisted pair and then benchmarked key generation and recovery. The test machine was a Dell Intel Pentium 4 running at 2.79 GHz with 512 MB RAM. The underlying OS was Ubuntu Linux 5.04.[4] The value $e = 65,537$ was used in all of our experiments. Also, all implementations use [9] to determine the number of iterations of Miller-Rabin. The implementations are based entirely on OpenSSL 0.9.8 that contains an EC library.

Table 1 gives the performance of the *fast SETUP algorithm* (ALG3) from Section 4. Key generation involves the computation of the public DH key exchange parameter and the shared secret. There are 2 wNAF splitting scalar multiplications per RSA key pair that is generated. So, the average number of inversions, additions, and doublings needs to be halved to evaluate wNAF splitting.

For the purposes of comparison, we also considered a modified version (called ALG 1) of [34] that has the following improvements (see row 1 of Table 2).

1. wNAF splitting is used to speed up the EC operations.
2. The scalar K is chosen and used and k is chosen based on K (same as in the fast SETUP algorithm). This means that point halving is avoided.
3. Once p is found, the speed-up in [2] is used to avoid divisions while searching for q ($1/p$ is precomputed, then we multiply and shift to compute q).

[4] For optimal results, the key generation/recovery program was run with root privileges using the `nice` − 10 command.

Table 1. Performance averages for Fast SETUP Algorithm

Performance Measure	$\lvert n\rvert = 1024$ 10,000 trials	$\lvert n\rvert = 2048$ 10,000 trials	$\lvert n\rvert = 4096$ 10,000 trials
block size for wNAF	8	8	8
value for w for wNAF	4	4	4
ave. # inversions/RSA key	43.00	43.10	43.13
ave. # additions/RSA key	84.33	84.28	84.29
ave. # doublings/RSA key	14.42	14.41	14.41
ave. RSA key gen. time (sec)	0.111	0.768	8.723
ave. RSA key rec. time (sec)	0.048	0.374	4.308

Table 2. Average RSA key generation/recovery time in seconds

	RSA Key Generation Method	$\lvert n\rvert = 1024$ 10,000 trials	$\lvert n\rvert = 2048$ 10,000 trials	$\lvert n\rvert = 4096$ 10,000 trials
1	Modified SAC '05 Gen (see [34])	0.233	1.164	10.171
2	Modified SAC '05 Rec	0.116	0.740	7.390
3	Fast SETUP Algorithm Gen	0.111	0.768	8.723
4	Fast SETUP Algorithm Rec	0.048	0.374	4.308
5	Normal OpenSSL Gen	0.131	0.937	9.294
6	OpenSSL with ext. incr. srch. Gen	0.082	0.733	8.632

The random permutation π that is used to transform the ECDH key exchange value is as follows. F is SHA-224 [10] modified to append a fixed, randomly selected prefix to the value that is hashed. H is the same except that it has a different randomly selected prefix. The input to π is $x = x_u \mid\mid x_\ell$ where $|x_u| = |x_\ell|$. Finally, $\pi(x) = (x_u \bigoplus F(x_\ell \bigoplus H(x_u))) \mid\mid (x_\ell \bigoplus H(x_u))$.

From Table 2, it is clear that despite these improvements to [34], the fast SETUP is still better. The hindrance to the running time is due to: (1) incremental search is not used to find p nor q, and (2) in each iteration in which q is tested, π is called and a Karatsuba multiplication is computed.

Consider the normal OpenSSL RSA key generation algorithm (ALG5) [24]. The performance of ALG5 is given in row 5 of Table 2. Like GenPrimeWithPRNGFast, it uses incremental search and trial-division to find primes. However, a critical difference is as follows. Suppose that overflow due to delta does not occur, a number that withstands trial-division is found, and Miller-Rabin is performed. If Miller-Rabin reports that p_1 is composite, then p_1 is selected over again randomly. In other words, OpenSSL gives up on p_1 if p_1 + delta is found before overflow and p_1 + delta is composite. ALG3 doesn't give up like this (see row 2 of Table 2). It will continue to increment delta by 2. It reselects p_1 only if delta exceeds `MAXDELTA`. See Appendix A for the security implications of this extended incremental search.

The gain from performing this extended incremental search more than compensates for the 2 wNAF splitting multiplications and the PRNG that generates p in $n = pq$ (the entries in row 3 are lower than those of row 5). We have therefore shown an asymmetric backdoor in key generation that is faster than OpenSSL RSA key generation. This implies that it may be worth researching an approach which provides provable resistance to timing analysis.

We benchmarked ALG5 when modified to perform this extended incremental search (ALG6). The results are given in row 6 of Table 2. This shows a way to speed up OpenSSL RSA key generation (the entries in row 6 are lower than those of row 5).

We also adapted ALG3 to use the OpenSSL incremental search method. The running time was slower than ALG5 so we omit the data. This, combined with the benchmark of ALG1 suggests that the approach of [34] will not achieve Property 5. No SETUP attacks have been designed against timing attacks, so there's no reason to believe that Property 5 will hold for previous solutions.

6 Conclusion

We presented a fast EC backdoor for RSA that uses incremental search. We showed that it runs *faster* than OpenSSL key generation. This shows that asymmetric backdoors in RSA can resist timing analysis.

References

1. von Ahn, L., Hopper, N.J.: Public-Key Steganography. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 323–341. Springer, Heidelberg (2004)
2. Aho, A., Hopcroft, J., Ullman, J.: Data Structures and Algorithms, p. 281. Addison-Wesley, Reading (1983)
3. Anderson, R.J.: A Practical RSA Trapdoor. Electronics Letters 29(11) (1993)
4. American National Standards Institute. ANSI X9.17: Financial institution key management (wholesale). ASC X9 Secretariat—ABA (1985)
5. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: 1st Annual ACM CCCS, pp. 62–73 (1993)
6. Coppersmith, D.: Finding a small root of a bivariate integer equation; factoring with high bits known. In: Eurocrypt 1996, pp. 178–189 (1996)
7. Crépeau, C., Slakmon, A.: Simple Backdoors for RSA Key Generation. In: The Cryptographers Track at the RSA Conference, pp. 403–416 (2003)
8. Daemen, J., Rijmen, V.: The Block Cipher Rijndael. In: Smart Card Research and Applications—CARDIS 1998, pp. 277–284 (2000)
9. Damgård, I., Landrock, P., Pomerance, C.: Average Case Error Estimates for the Strong Probable Prime Test. Math. Comp. 61(203), 177–194 (1993)
10. NIST (2002) FIPS 180-2, Secure Hash Standard (SHS). Change notice (Feb, 2004) introduces SHA-224.

11. Gaudry, P.: A comparison and a combination of SST and AGM algorithms for counting points on elliptic curves in characteristic 2. In: Advances in Cryptology—Asiacrypt 2002, pp. 311–327 (2002)
12. Gaudry, P., Hess, F., Smart, N.: Constructive and Destructive Facets of Weil Descent on Elliptic Curves. J. of Cryptology 15, 19–46 (2002)
13. Joux, A., Nguyen, K.: Separating DDH from CDH in Cryptographic Groups. Journal of Cryptology 16(4), 239–247 (2003)
14. Kaliski, B.S.: A Pseudo-Random Bit Generator Based on Elliptic Logarithms. In: Advances in Cryptology—Crypto 1986, pp. 84–103 (1986)
15. Kaliski, B.S.: Elliptic Curves and Cryptography: A Pseudorandom Bit Generator and Other Tools, Feb. 1988. PhD Thesis. MIT, Cambridge (1988)
16. Kaliski, B.S.: One-Way Permutations on Elliptic Curves. Journal of Cryptology 3(3), 187–199 (1991)
17. Kaliski, B.: Anderson's RSA trapdoor can be broken. Electronics Letters 29(15), 1387–1388 (1993)
18. Kocher, P.: Timing Attacks on Implementations of DH, RSA, DSS, and Other Systems. In: Advances in Cryptology—Crypto 1996, pp. 104–113 (1996)
19. Kucner, D., Kutylowski, M.: Stochastic Kleptography Detection. In: Public-Key Cryptography and Computational Number Theory, pp. 137–149 (2001)
20. Miller, G.L.: Riemann's Hypothesis and Tests for Primality. J. Comp. Syst. Sci. 13(3), 300–317 (1976)
21. Menezes, A., van Oorschot, P., Vanstone, S.: Vanstone. Handbook of Applied Cryptography. In: Table 4.4, Note 4.51 (ii), p. 148. CRC Press, Boca Raton (1997)
22. Möller, B.: Improved Techniques for Fast Exponentiation. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 298–312. Springer, Heidelberg (2003)
23. Möller, B.: A Public-Key Encryption Scheme with Pseudo-Random Ciphertexts. In: ESORICS 2004, pp. 335–351 (2004)
24. OpenSSL version 0.9.8. On the web at `http://www.openssl.org`
25. PKCS #1 v2.1: RSA Cryptography Standard. RSA Labs (Jun 14, 2002)
26. Rabin, M.O.: Probabilistic Algorithms for Testing Primality. J. Number Th. 12, 128–138 (1980)
27. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. CACM 21(2), 120–126 (1978)
28. Satoh, T., Skjernaa, B., Taguchi, Y.: Fast computation of canonical lifts of elliptic curves and its application to point counting. Finite Fields and Their Applications 9, 89–101 (2003)
29. Schoof, R.: Elliptic Curves Over Finite Fields and the Computation of Square Roots mod p. In: Mathematics of Computation, vol. 44, pp. 483–494 (1985)
30. Vanstone, S.A., Mullin, R.C., Agnew, G.B.: Elliptic curve encryption systems. US Patent 6,141,420, Filed: (Jan 29, 1997)
31. Vercauteren, F., Preneel, B., Vanderwalle, J.: A memory efficient version of Satoh's algorithm. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 1–13. Springer, Heidelberg (2001)
32. Young, A., Yung, M.: The Dark Side of Black-Box Cryptography, or: Should we trust Capstone? In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 89–103. Springer, Heidelberg (1996)
33. Young, A., Yung, M.: Kleptography: Using Cryptography Against Cryptography. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 62–74. Springer, Heidelberg (1997)
34. Young, A., Yung, M.: A Space Efficient Backdoor in RSA and its Applications. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 128–143. Springer, Heidelberg (2006)

A Security

The fast SETUP does not produce prime pairs where each prime is uniformly distributed among the set of acceptable RSA primes. Consider the twin primes $p_a, p_b = p_a + 2$. p_a may have numerous composites immediately before it, whereas p_b has only 1 chance at being selected during incremental search (lsb of $p_a + 1$ is set to 1). The experiments lead to the following apparent dichotomy:

1. By using the OpenSSL prime search method the SETUP will likely be slower to due the overhead of the EC and Feistel operations. This leads to polynomially indistinguishable distributions but distinguishability via timing analysis.
2. By using the extended incremental prime search method, the SETUP will be faster than OpenSSL RSA key generation and this leaves some leeway to refine that algorithm so that the running times match. This *may* lead timing indistinguishability but runs the risk of poly-time distinguishability.

We do not have a concrete resolution for this dichotomy. In some sense the problem is to achieve the impossible: have matching running times while performing extra work (EC/Feistel operations). We state up-front Assumption 1 below that is needed for provable indistinguishability of the backdoor key generator vs. normal OpenSSL key generation. As assumptions go it could be a single point of failure in the backdoor algorithm. But, as explained below such a failure may have important implications in practice.

Assumption 1: The distribution over pairs of primes that result from the extended incremental search (used in the fast SETUP) is computationally indistinguishable from the distribution over pairs of primes that result from the incremental search algorithm of OpenSSL.

We leave the validation/invalidation of Assumption 1 open.

The designer may deploy a key generator with a secret design. ALG3 might be fast enough to avoid suspicion, so "indistinguishability" may not be a critical issue. Smart card manufacturers and software companies often keep the key generation algorithm secret on the basis that it is proprietary. From this perspective our results serve a different purpose: if Assumption 1 is wrong then this may provide a method for *detecting* backdoor attacks in black-box products based on, e.g., the efficiently detectable suspicious distribution over prime pairs that results from the particulars of the prime incremental search algorithm.

Put another way, our results indicate that *particular attention should be paid* to the incremental search algorithm used in selecting primes in a secret key generation algorithm. One may be able to create an independent implementation using the same hardware and exact same incremental search algorithm. By comparing the running time with the black-box version, a backdoor could possibly be detected based on speed. Our experiments lead us to the following observation.

Claim 1: *When one can choose a faster incremental search method for primes and by using a ECDH backdoor with precomputations it is possible to construct an asymmetric backdoor in RSA key generation that runs faster than native RSA key generation.*

A Watermarking Scheme in the Encrypted Domain for Watermarking Protocol

Bin Zhao[1], Lanjun Dang[1], Weidong Kou[1], Jun Zhang[2], Zan Li[1], and Kai Fan[1]

[1] The State Key Laboratory of ISN, Xidian University, Xi'an, 710071, China
{binzhao,ljdang,wdkou,zanli,kfan}@mail.xidian.edu.cn
[2] School of CSSE, University of Wollongong, Wollongong, NSW 2522, Australia
jz484@uow.edu.au

Abstract. In most watermarking schemes for copyright protection, a seller always knows the embedded watermark identifying a buyer. However, it incurs both repudiation issue and framing issue. To solve these problems, many watermarking protocols have been proposed based on watermarking schemes in the encrypted domain. In this paper, we enhance an existing watermarking scheme and discuss public key cryptosystems used in watermarking protocols. Then, a new watermarking scheme in the encrypted domain with flexible watermarking capacity is proposed for watermarking protocol. It improves the robustness of watermarked image against JPEG compression after decryption and enables image tampering detection. The blind watermark extracting employs the same threshold criterion and secret keys as watermark embedding. Experimental results demonstrate that the enhanced scheme reduces computing overload and increases effective watermarking capacity, and that the proposed watermarking scheme in the encrypted domain outperforms a previous scheme.

1 Introduction

In order to deal with many important issues of multimedia security, such as distribution, authentication, and copyright protection, watermarking has increasingly become a surge of research activity. It enables a seller to hide additional bits into multimedia content while preserving content's quality. In the past decade, a large number of excellent watermarking schemes have been presented. In most watermarking schemes, embedding watermark into host content is executed by the seller in behalf of intellectual property. However, since the seller always knows the embedded watermark identifying a buyer, it causes both repudiation issue (a guilty buyer producing unauthorized copies could be able to repudiate the fact and claim that the copies were possibly made by the seller) and framing issue (an honest buyer could be falsely accused for reparation by a malicious seller who can reuse the embedded watermark to frame). Hence it is significant in the sense that the watermarking framework needs protocols to solve both the resolution of the rightful ownership problem and the protection of the buyer's right problem, which is first introduced by L. Qiao and K. Nahrstedt [1].

For this reason, many watermarking protocols have been proposed based on watermarking scheme in the encrypted domain. In such a condition, the seller can not produce copies containing the watermark identifying the buyer, because he does not know the exact watermark from the ciphertext in the embedding procedure. When an illegal copy is found, the seller can prove to a third party that the buyer is certainly guilty without repudiation and framing. N. Memon and P. W. Wong [2] presented a buyer-seller watermarking protocol to resolve both the pirate tracing problem and the buyer's right problem. Successively, Chin-Laung Lei *et al.* [3] pointed out the unbinding problem and proposed an efficient and anonymous buyer-seller watermarking protocol. Recently, J. Zhang *et al.* [4] proposed a secure buyer-seller watermarking protocol based on the idea of sharing a secret. In these protocols, the protocol's security depends essentially on the watermarking scheme in the encrypted domain. However, none of these protocols further considered how to embed encrypted watermark bits into host image content in detail.

M. Kuribayashi and H. Tanaka [5] presented an anonymous fingerprinting protocol and a quantization-based scheme for embedding encrypted watermark bits by additive homomorphic property. According to the simulation results, 32-bit information can be embedded into the 256×256 grayscale image "Lena". However, the information payload of this watermarking scheme is too deficient to employ anti-collusion codes [6] which are designed to immunize the analysis of colluded buyers in practical applications. Furthermore, it seems inefficient that 1-bit information is repeatedly embedded dozens of times to resist against general JPEG compression.

In this paper, an existing watermarking scheme is enhanced, and then a watermarking scheme in the encrypted domain is proposed for watermarking protocol. It aims to embed encrypted watermark bits with flexible volume, adaptively satisfying different practical requirements. The new watermarking scheme in the encrypted domain differs from [5] in several aspects. (1) Threshold criterion is employed to decide where to embed watermark bits among candidate transformation coefficients. (2) Flexible watermarking capacity is achieved by adjusting many parameters as secret keys. (3) Preprocessing method is adopted to reduce computing overload and increase effective watermarking capacity. (4) Watermark extracting does not need the original image, but employs the same threshold criterion and secret keys as watermark embedding to extract the entire watermark bits. (5) The resistance against JPEG compression after decryption is improved, and the tampering on watermarked images can be detected.

2 Enhanced Watermarking Scheme

2.1 SEC Scheme

K. Solanki *et al.* [7] proposed an image-adaptive watermarking technique called selectively embedding in coefficients (SEC) scheme. It applies local criteria to choose where to hide data by means of one-by-one DCT coefficients selection with the goal of minimal visual distortion. Here, SEC scheme is reviewed briefly.

An image is partitioned into 8×8 nonoverlapping blocks, to which the DCT is applied, and then DCT coefficients c_{ij} of a block are divided by the quantization table at designated quality factor (QF). After zig-zag scanning, the quantized coefficients $\widehat{c_k}$ in a fixed low frequency band (1≤*k*≤N) are rounded to the nearest integer, and then taking their magnitude to obtain $\overline{r_k}$.

$$\overline{r_k} = \left|\mathrm{int}_{near}(\widehat{c_k})\right|, \quad 1 \le k \le \mathrm{N}. \tag{1}$$

A quantized coefficient $\widehat{c_k}$ will be selected to embed a bit, if and only if it satisfies the threshold criterion that $\overline{r_k}$ is greater than a positive integer threshold *t*. Thus, the quantized coefficients $\widehat{c_k}$ are processed as following.

$$\widehat{d_k} = \begin{cases} \mathrm{int}_{bl}(\widehat{c_k}), & \text{if } \overline{r_k} > t, \text{ and } 1 \le k \le \mathrm{N}, \\ \pm t, & \text{if } \overline{r_k} = t, \text{ and } 1 \le k \le \mathrm{N}, \\ \widehat{c_k}, & \text{otherwise.} \end{cases} \tag{2}$$

where $\mathrm{int}_{bl}(\bullet)$ denotes either $\mathrm{int}_{odd}(\bullet)$ (round to the nearest odd integer) or $\mathrm{int}_{even}(\bullet)$ (round to the nearest even integer), which is depended on the incoming bit with prior knowledge, either 1 or 0 correspondingly. After inverse zig-zag scanning, multiplication by the quantization table at designated QF and IDCT, the watermarked image of that block can be reconstructed.

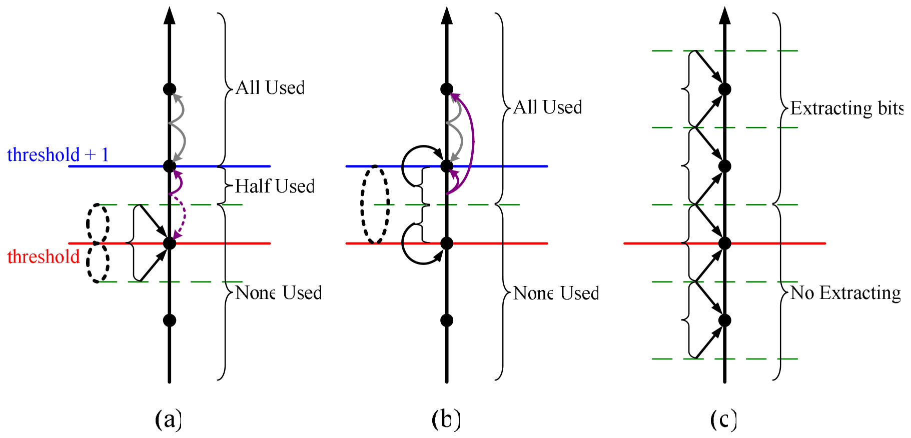


Fig. 1. (a) SEC scheme. (b) Enhanced scheme. (c) Extracting method.

However, if the magnitude of a rounded coefficient $\mathrm{int}_{bl}(\widehat{c_k})$ equals threshold *t* for a bit *bl*, then the extractor will disregard this coefficient and lose the embedded bit *bl*, because its magnitude is not greater than threshold *t* with respect to the threshold criterion. It actually happens to the selected coefficients $\widehat{c_k}$ whose magnitude lies between (t+1/2) and (t+1) as Fig. 1(a) illustrated. In order to maintain the synchronization between embedder and extractor, SEC scheme uses checking and re-embedding to deal with these lost bits [7]. If $\mathrm{int}_{bl}(\widehat{c_k})$ equals threshold t, the same bit bl will be embedded into the next qualified coefficient. Thus the computing overload obviously rises, because it requires always checking whether this condition happens. Moreover, since generally half of these coefficients whose magnitude lies between (t+1/2) and (t+1) are used in vain, the effective embedding capacity is decreased due to re-embedding.

2.2 Enhanced Scheme

To solve these problems, an enhanced scheme is proposed in this section. After 8×8 block partition, DCT, division by the quantization table at designated QF and zig-zag scanning, in a fixed low frequency band (1≤k≤N), the quantized coefficient $\widehat{c_k}$ whose magnitude lies between threshold t and (t+1) are rounded to the nearest integer as a preprocessing, leaving a protection interval there as Fig. 1(b) shown.

$$\widehat{c_k} = \begin{cases} \pm t, & \text{if } t < \left|\widehat{c_k}\right| < (t+\frac{1}{2}), \text{ and } 1 \le k \le \mathrm{N}, \\ \pm(t+1), & \text{if } (t+\frac{1}{2}) \le \left|\widehat{c_k}\right| < (t+1), \text{ and } 1 \le k \le \mathrm{N}, \\ \widehat{c_k}, & \text{otherwise.} \end{cases} \quad (3)$$

Then, the quantized coefficients $\widehat{c_k}$ in a fixed low frequency band (1≤k≤N) are rounded to the nearest integer, and then taking their magnitude to obtain $\overline{r_k}$. In the enhanced scheme, a quantized coefficient $\widehat{c_k}$ will be selected to embed a bit by the same threshold criterion that $\overline{r_k}$ is greater than threshold t.

$$\widehat{d_k} = \begin{cases} \mathrm{int}_{bl}(\widehat{c_k}), & \text{if } \overline{r_k} > t, \text{ and } 1 \le k \le \mathrm{N}, \\ \widehat{c_k}, & \text{otherwise.} \end{cases} \quad (4)$$

where $\mathrm{int}_{bl}(\bullet)$ also depends on the incoming bit with prior knowledge. To avoid ambiguity in some special cases, if $\widehat{c_k}$ is actually an even integer, $\mathrm{int}_{odd}(\widehat{c_k})$ function should round $\widehat{c_k}$ to the nearest odd integer either $(\left|\widehat{c_k}\right|+1)$ or $-(\left|\widehat{c_k}\right|+1)$; if $\widehat{c_k}$ is actually an odd integer, $\mathrm{int}_{even}(\widehat{c_k})$ function should round $\widehat{c_k}$ to the nearest even integer either $(\left|\widehat{c_k}\right|+1)$ or $-(\left|\widehat{c_k}\right|+1)$, the sign of which depends on positive or negative value correspondingly. These uniform rounding methods are used to the quantized coefficients of integer value, especially $\left|\widehat{c_k}\right| = (t+1)$.

In the enhanced scheme, the preprocessing method plays the same role as $\widehat{d_k} = \pm t$, if $\overline{r_k} = t$, and $1 \le k \le \mathrm{N}$ in formula (2) in SEC scheme, but it avoids using checking and re-embedding to the selected coefficient $\widehat{c_k}$ whose magnitude lies between (t+1/2) and (t+1). The enhanced scheme never makes the rounded coefficient $\mathrm{int}_{bl}(\widehat{c_k})$ down to threshold t after embedding arbitrary bit, so the entire embedded watermark bits can be extracted without losing any bit. Therefore, compared with SEC scheme, the enhanced scheme reduces the computing overload of checking and re-embedding and increases the effective watermarking capacity.

Fig. 1(c) shows that two watermarking schemes employ the same extracting method to obtain $\overline{r_k}$ and use the same threshold criterion to determine embedding positions. Then, the entire embedded watermark bits can be extracted one-by-one using the odd-even judgment: If $\overline{r_k}$ is an odd integer, the embedded bit is 1; else $\overline{r_k}$ is an even integer, the embedded bit is 0.

3 New Watermarking Scheme in the Encrypted Domain

In the watermarking protocols [2]-[5], for the sake of no repudiation and no framing, watermark should be encrypted by the buyer's public key unexposed to the seller. As watermark embedder, the seller usually has the original image and the encrypted watermark. Following the successive processes as Fig. 2 shown, the seller can embed the watermark into the host image in the encrypted domain and send the encrypted watermarked image to the buyer.

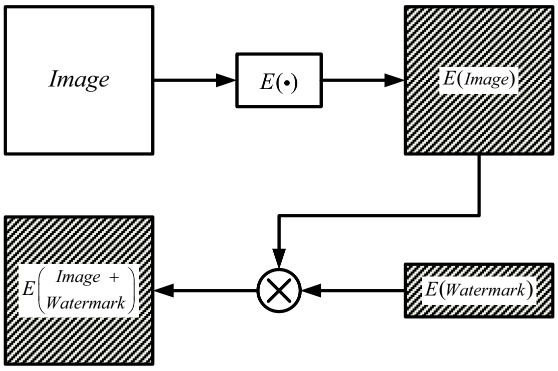


Fig. 2. Embedding watermark in the encrypted domain

A public key cryptosystem used in watermarking protocols should include three important properties as following, but are not limited to. (E(•) and D(•) denote public key encryption and decryption respectively, random number r_1 and r_2 are uniformly and independently selected)

1) Additive Homomorphic Property: Multiplying two ciphertexts $E(x,r_1)$ and $E(y,r_2)$ leads to addition of two plaintexts x and y.

$$D(E(x,r_1)\bullet E(y,r_2)) = D(E(x+y,r')) = x+y \mod \mathrm{n} \tag{5}$$

$$D(E(x,r)^k) = D(E(kx,r')) = kx \mod \mathrm{n} \tag{6}$$

2) Semantic Security: The encryption of a message x is computationally indistinguishable to a polynomial time from the encryption of a different message y.

$$E(x,r_1) \xleftrightarrow{?} E(y,r_2) \tag{7}$$

3) Self Blinding: A given ciphertext can be publicly changed into another one without altering the plaintext, and the relationship between two ciphertexts can be concealed.

$$D(E(x,r_1)\bullet f(r_2)) = D(E(x,r')) = x \mod \mathrm{n} \tag{8}$$

It is known that Paillier cryptosystem [8] and Okamoto–Uchiyama cryptosystem [9] conform to these important properties and share a similar structure. Hence, both of them can be used in the proposed scheme, and the employment of which cryptosystem depends on practical applications.

3.1 Overview

The main goal of this work is embedding encrypted watermark bits with flexible volume, adaptively satisfying different practical requirements. The enhanced scheme is adopted to select qualified coefficients to embed watermark bits, and the embedding method proposed in [5] is employed to insert the encrypted watermark bits into the encrypted coefficients by additive homomorphic property.

Since the quantization table used in JPEG compression algorithm [10] is based on human perceptual characteristics, it is used in SEC scheme and the enhanced scheme to reduce visual distortion. Note that the quantization table at designated QF is composed of real numbers, and they can not apply public key cryptosystem based on the algebraic property of integer. Therefore, the quantization table at designated QF is slightly modified in such a way that all the real numbers are cut off to integers M_{ij}^{QF}. Furthermore, the seller should predefine many parameters as a set of secret keys, such as a positive integer threshold t for the threshold criterion, the value of designated QF, the number N of candidate coefficients per block in a fixed frequency band ($1 \le k \le \mathrm{N}$), and a random permutation $P(\bullet)$.

In watermark embedding procedure, the threshold criterion that the quantized coefficient's magnitude is greater than the threshold t is adopted to decide where to embed watermark bits. In order to employ a public key cryptosystem, special round-off strategies are applied to both the selected coefficients and the other coefficients. In the embedding positions, watermark insertion is performed in the encrypted domain by additive homomorphic property. The value of each encrypted selected coefficient can be correspondingly altered by each encrypted watermark bit whose value, whether 1 or 0, is completely unknown in advance.

In watermark extracting procedure, the watermarked image is a plaintext after decryption and IDCT. Watermark extractor does not need the original image, but employs the same threshold criterion and secret keys as the watermark embedder to determine the embedding positions, and then extracts the watermark.

3.2 Watermark Embedding Procedure

The watermark embedding steps are shown in the flow chart Fig. 3.

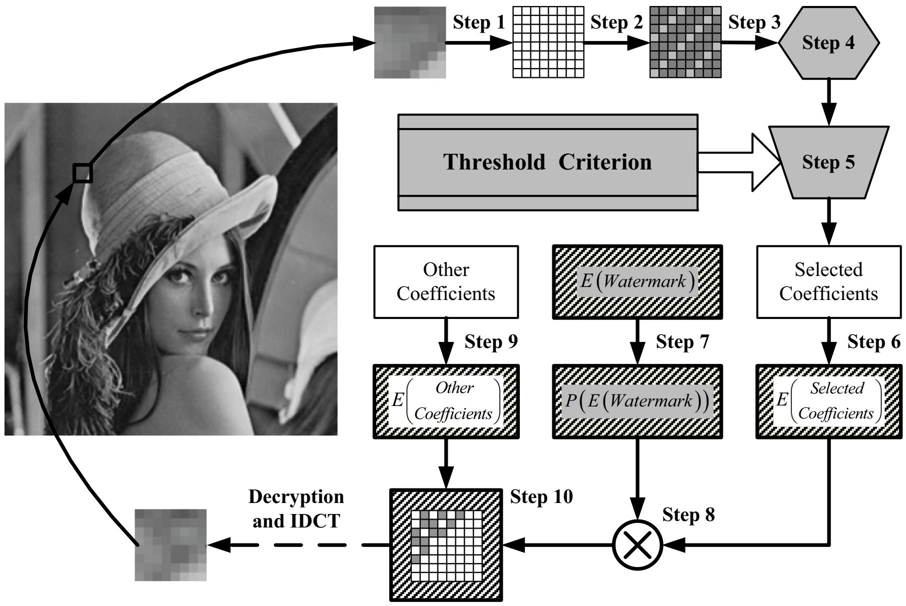


Fig. 3. Watermark embedding steps

Step 1. An image is partitioned into 8×8 nonoverlapping blocks and each block is transformed by DCT.

Step 2. All the DCT coefficients c_{ij} of a block are divided by the customized quantizing step size M_{ij}^{QF} to obtain the quantized coefficients $\widehat{c_{ij}}$.

$$\widehat{c_{ij}} = \frac{c_{ij}}{M_{ij}^{QF}}, \quad \forall i, j \in \{0,1,...,7\}. \tag{9}$$

Step 3. The quantized coefficients $\widehat{c_{ij}}$ are zig-zag scanned to obtain $\widehat{c_k}$ ($0 \le k \le 63$). Note that only the quantized coefficients in a fixed low frequency band ($1 \le k \le N$) are considered as candidate coefficients, and the DC frequency component coefficient denoted by $\widehat{c_0}$ should avoid using.

Step 4. In the low frequency band (1≤*k*≤N), the quantized coefficients $\widehat{c_k}$ whose magnitude $\left|\widehat{c_k}\right|$ lies between threshold *t* and (*t*+1) are rounded to the nearest integer as a preprocessing.

$$\widehat{c_k} = \begin{cases} \pm t, & \text{if } t < \left|\widehat{c_k}\right| < (t+\frac{1}{2}), \text{ and } 1 \le k \le \mathrm{N}, \\ \pm(t+1), & \text{if } (t+\frac{1}{2}) \le \left|\widehat{c_k}\right| < (t+1), \text{ and } 1 \le k \le \mathrm{N}, \\ \widehat{c_k}, & \text{otherwise.} \end{cases} \tag{10}$$

In the other frequency band (N<*k*≤63), the quantized coefficients $\widehat{c_k}$ are never used for embedding and will be processed in the later steps.

Step 5. In the low frequency band (1≤*k*≤N), the quantized coefficients $\widehat{c_k}$ whose magnitude $\left|\widehat{c_k}\right|$ is greater than threshold *t* as the threshold criterion are selected and rounded to the nearest even integer.

$$\overline{c_k} = \mathrm{int}_{even}(\widehat{c_k}), \quad \text{for } \left|\widehat{c_k}\right| > t, \text{ and } 1 \le k \le \mathrm{N}. \tag{11}$$

Note that if $\widehat{c_k}$ is actually an odd integer, $\mathrm{int}_{even}(\widehat{c_k})$ function should round $\widehat{c_k}$ to the nearest even integer either $(\left|\widehat{c_k}\right|+1)$ or $-(\left|\widehat{c_k}\right|+1)$, the sign of which depends on positive or negative value correspondingly.

The other quantized coefficients $\widehat{c_k}$ in the low frequency band (1≤*k*≤N) are never used for embedding and will be processed in the later steps.

Step 6. All the selected coefficients $\overline{c_k}$ are inverse zig-zag scanned to obtain $\overline{c_{ij}}$, and then every $\overline{c_{ij}}$ is encrypted with the buyer's public key and a random number b_n to calculate the encrypted coefficient $E(\overline{c_{ij}}, b_n)$. Note that each embedding position is represented by subindex *ij* in that block.

Step 7. The order of the encrypted watermark bitstream $E(\overrightarrow{w_s}, \overrightarrow{a_l})$ is changed by the random permutation $P(\bullet)$ as one of secret keys to obtain the encrypted permuted watermark bitstream $E(\overrightarrow{w_p}, \overrightarrow{a_m})$.

$$P(E(\overrightarrow{w_s}, \overrightarrow{a_l})) = P(E(w_s^1, a_l^1); \ldots; E(w_s^N, a_l^N)) = E(w_p^1, a_m^1); \ldots; E(w_p^N, a_m^N) = E(\overrightarrow{w_p}, \overrightarrow{a_m}) \tag{12}$$

Step 8. In the embedding positions, the watermarked coefficients could be represented as following.

$$\overline{d'_{ij}} = (\overline{c_{ij}} \pm w_p) \bullet M_{ij}^{QF}, \quad \text{for } \left|\widehat{c_k}\right| > t, \text{ and } 1 \le k \le \mathrm{N}. \tag{13}$$

Without the knowledge of the permuted watermark bit w_p 's value from the ciphertext $E(w_p, a_m)$, when to adopt $\overline{c_{ij}} + w_p$ or $\overline{c_{ij}} - w_p$ relies on the following embedding method. If $\widehat{c_{ij}} \geq \overline{c_{ij}}$ (Case 1), then calculate $\overline{c_{ij}} + w_p$; else $\widehat{c_{ij}} < \overline{c_{ij}}$ (Case 2), then calculate $\overline{c_{ij}} - w_p$. Since no selected coefficient lies between threshold t and (t+1) after the preprocessing in step 4, it never makes the value $(\overline{c_{ij}} \pm w_p)$ down to threshold t.

Hence the encrypted watermarked coefficients $E(\overline{d'_{ij}}, r')$ can be calculated by multiplying two ciphertexts $E(\overline{c_{ij}}, b_n)$ and $E(w_p, a_m)$ in either Case 1 or Case 2:

Case 1: If $\widehat{c_{ij}} \geq \overline{c_{ij}}$, then

$$\begin{aligned} E(\overline{d'_{ij}}, r') &= (E(\overline{c_{ij}}, b_n)\bullet E(w_p, a_m))^{M_{ij}^{QF}} \\ &= (E(\overline{c_{ij}} + w_p, b_n + a_m))^{M_{ij}^{QF}} \\ &= E((\overline{c_{ij}} + w_p)\bullet M_{ij}^{QF}, (b_n + a_m)\bullet M_{ij}^{QF}) \end{aligned} \tag{14}$$

Case 2: Else $\widehat{c_{ij}} < \overline{c_{ij}}$, then

$$\begin{aligned} E(\overline{d'_{ij}}, r') &= (E(\overline{c_{ij}}, b_n)\bullet E(w_p, a_m)^{-1})^{M_{ij}^{QF}} \\ &= (E(\overline{c_{ij}} - w_p, b_n - a_m))^{M_{ij}^{QF}} \\ &= E((\overline{c_{ij}} - w_p)\bullet M_{ij}^{QF}, (b_n - a_m)\bullet M_{ij}^{QF}) \end{aligned} \tag{15}$$

Step 9. In the other positions, the unwatermarked coefficients are rounded to the nearest integer by the following operations.

$$\overline{d_{ij}} = \begin{cases} \text{int}_{near}(c_{ij}), & \text{for } 0 \leq \left|\widehat{c_k}\right| < t, \text{ and } 1 \leq k \leq \text{N}, \\ \pm t\bullet M_{ij}^{QF}, & \text{for } \left|\widehat{c_k}\right| = t, \text{ and } 1 \leq k \leq \text{N}, \\ \text{int}_{near}(c_{ij}), & \text{for } \forall\left|\widehat{c_k}\right|, \text{ and } k = 0 \bigcup \text{N} < k \leq 63. \end{cases} \tag{16}$$

Then, every $\overline{d_{ij}}$ is encrypted with the same public key and a random number r to calculate the encrypted unwatermarked coefficient $E(\overline{d_{ij}}, r)$.

Step 10. All the DCT coefficients d_{ij} of that block could be represented as following.

$$d_{ij} = \begin{cases} \overline{d'_{ij}}, & \text{for } \left|\widehat{c_k}\right| > t, \text{ and } 1 \leq k \leq \text{N}, \\ \overline{d_{ij}}, & \text{otherwise.} \end{cases} \tag{17}$$

Hence all the encrypted DCT coefficients $E(d_{ij}, r)$ of that block are composed of watermarked coefficients and unwatermarked coefficient in the encrypted domain.

$$E(d_{ij},r)=\begin{cases}E(\overline{d_{ij}},r), & \text{for } \left|\widehat{c_k}\right|>t, \text{ and } 1\le k\le \text{N},\\ E(\overline{d'_{ij}},r'), & \text{otherwise.}\end{cases} \tag{18}$$

With block-by-block processing, the seller obtains the entire encrypted DCT coefficients of the watermarked image, and then sends them to the buyer. Finally, the buyer obtains the watermarked image by decrypting the entire DCT coefficients and employing IDCT.

3.3 Watermarking Extracting Procedure

Some initial steps of the extracting procedure are similar to the one of embedding procedure, e.g. from **Step 1** to **Step 3**.

Step 4. In the low frequency band (1≤*k*≤N), the quantized coefficients $\widehat{d_k}$ are rounded to the nearest integer.

$$\overline{d_k}=\text{int}_{near}(\widehat{d_k}),\quad 1\le k\le \text{N}. \tag{19}$$

Step 5. The quantized coefficient integers $\overline{d_k}$ whose magnitude $\left|\overline{d_k}\right|$ is greater than the threshold *t* as the same threshold criterion are considered as embedding a permuted watermark bit. Hence, every permuted watermark bit w_p can be readily extracted using the odd-even judgment as following.

$$w_p=\begin{cases}1, & \text{if } \overline{d_k} \text{ is odd}, \quad \left|\overline{d_k}\right|>t, \text{ and } 1\le k\le \text{N},\\ 0, & \text{if } \overline{d_k} \text{ is even}, \quad \left|\overline{d_k}\right|>t, \text{ and } 1\le k\le \text{N}.\end{cases} \tag{20}$$

Step 6. The permuted watermark bitstream $\overrightarrow{w_p}$ is changed to the original order by the same random permutation $P(\bullet)$, and finally the original watermark bitstream $\overrightarrow{w_s}$ is retrieved.

$$P(\overrightarrow{w_p})=P(w_p^1;...;w_p^N)=w_s^1;...;w_s^N=\overrightarrow{w_s} \tag{21}$$

4 Experimental Results

4.1 Results of the Enhanced Scheme and SEC Scheme

In this section, we report some simulation results to compare the enhanced scheme with SEC scheme. Using the same 512×512 grayscale images, the enhanced scheme embeds watermark bits into the DCT coefficients selected by the same threshold criterion in the same low frequency band (1≤*k*≤14) as SEC scheme [7]. This parameter 14 can be further changed, and it is independent of the host image. Note that the real numbers of the quantization table at designated QF are cut off to integers, but we do not apply round-off to the entire DCT coefficients. All of the embedded watermark bits are equiprobably and independently generated with $p(1) = p(0) = 0.5$, and each result is the average over a large number of repeated tests. Table 1 gives the number of embedded and lost watermark bits and the corresponding PSNR of two watermarking schemes in different images. The embedded bits varies from one to another depended on image inherent characteristics and threshold t. Surprisingly, we observe that both the effective watermarking capacity and the number of lost bits in SEC scheme undesirably shift at every time depended on each embedded watermark bitstream, while the enhanced scheme is not subject to this flaw.

Using the same parameters, the enhanced scheme embeds the same number of watermark bits as SEC scheme, and the PSNR is quite near to that of SEC scheme. In addition, the enhanced scheme reduces the computing overload of checking and re-embedding. Furthermore, in the enhanced scheme, the entire embedded watermark bits can be extracted without any loss, and the effective watermarking capacity is higher than that of SEC scheme because of no re-embedding.

Table 1. The number of watermark bits and PSNR at designated QF 25 in different images

512×512 QF=25	Threshold t	Embed bits	SEC scheme		Enhanced scheme	
			Lose bits	PSNR (dB)	Lose bits	PSNR (dB)
Baboon	0	31056	5734.6	32.25	0	30.25
	1	13291	1995.9	35.66	0	35.10
	2	6637	950.7	39.33	0	38.44
	3	3436	481.2	42.09	0	41.24
Boat	0	19291	4112.5	34.17	0	32.05
	1	7269	1068.6	38.03	0	37.77
	2	3782	417.2	41.68	0	41.01
	3	2372	224.3	43.68	0	43.10
Peppers	0	13261	2764.1	35.84	0	33.91
	1	5338	670.8	39.77	0	39.65
	2	3122	305.3	42.71	0	42.10
	3	2048	161.6	44.25	0	43.74

4.2 Results of New Watermarking Scheme in the Encrypted Domain

In this section, we present many experimental results to demonstrate the performance of the new watermarking scheme in the encrypted domain. For less ciphertext length and lower transmission load, $|n|$=512-bit Paillier cryptosystem [8] is employed in our experiments. The entire tests were performed on the same 256×256 grayscale image "Lena", the single test image reported in [5]. Watermark bits are embedded into the DCT coefficients selected by the threshold criterion in a fixed low frequency band ($1 \le k \le 14$). All of the embedded watermark bits are equiprobably and independently generated with $p(1) = p(0) = 0.5$, and each result is the average over a large number of repeated tests.

4.2.1 Watermarking Capacity

The proposed watermarking scheme in the encrypted domain has a flexible watermarking capacity in a given host image by adjusting many parameters. Table 2 reports the number of watermark bits and the PSNR of watermarked images after decryption. It indicates that using higher threshold *t* results in better visual quality of watermarked image but lower capacity to embed watermark bits. The visual quality measured by PSNR can be further improved by using higher threshold *t*. Therefore, there is a tradeoff between the embedding capacity and visual quality, and it adaptively satisfies different practical requirements. Note that the number of watermark bits reported here is actually uncoded bits.

Table 2. The number of watermark bits and PSNR at different designated QF

256×256 Lena	QF=25		QF=50	
Threshold *t*	Embed bits	PSNR (dB)	Embed bits	PSNR (dB)
0	4916	32.29	7011	36.86
1	2260	37.40	3828	40.78
2	1341	40.27	2628	42.89
3	890	42.50	1948	44.28
4	626	43.89	1502	45.45
5	456	45.21	1208	46.29

4.2.2 JPEG Compression Resistance

For JPEG compression, since the proposed watermarking scheme in the encrypted domain is tuned to JPEG quantization table at designated QF, the embedded watermark bits is resilient for the JPEG compression less severe than the designated QF and can be retrieved perfectly at the extractor. To provide an objective measure of JPEG compression resistance, we adopted the fair evaluation strategies proposed by M. Kutter and F. A. P. Petitcolas in [11]. The performance of the watermarked image resisting against JPEG compression can be measured by the bit-error rate (BER), which is defined as the ratio of the number of incorrectly extracted watermark bits to the total number of embedded watermark bits.

First, we display the ability of JPEG compression resistance. Fig. 4 shows the resistance against JPEG compression at various cases with different parameters. In a given JPEG compression case, the watermarked image at low designated QF has less BER than the one at high designated QF, and the BER can be further decreased by using higher threshold *t* at the cost of reducing watermarking capacity.

In addition, we illustrate the performance comparison between the proposed scheme and M. Kuribayashi *et al.* scheme, because the JPEG compression is the dominating attack reported in [5]. Fig. 5 shows the correct extraction rate (CER) of both schemes under JPEG compression. It is seen that the CER of the proposed scheme at designated QF 25 is superior to M. Kuribayashi *et al.*'s, which is represented by the curve with square taken from [5], and that the proposed scheme at designated QF 50 is of comparable performance. Note that M. Kuribayashi *et al.* scheme employs 75 times

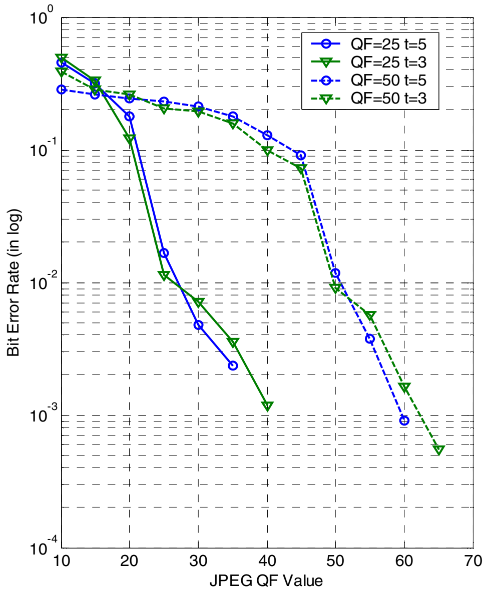


Fig. 4. The resistance against JPEG compression

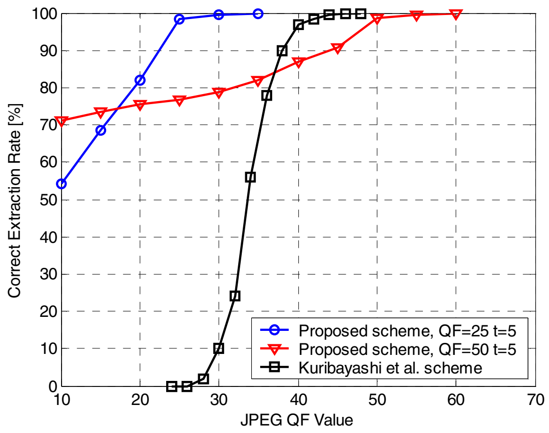


Fig. 5. The comparison of correct extraction rate

repeated coding for 1-bit information, while the entire embedded watermark bits of the proposed scheme are uncoded and efficient enough for free bit-error recovery against JPEG compression.

4.2.3 Image Tampering Detection

For image tampering, the proposed watermarking scheme in the encrypted domain can detect various tampering on the watermarked image after decryption and locate the tampered area in block level. If a watermarked image has undergone tampering, the tampered area in the watermarked image can be easily detected by comparing the retrieved watermark bitstream with the original watermark bitstream, and most errors are concentrated in the area where image tampering was done. It is a useful ability in commercial and forensic applications to detect whether the image has been tampered and disclose the tampered area as evidence. For example, two globally tampered watermarked image with the same parameters QF=25, t=1 are displayed in Fig. 6(a) (PSNR=19.62dB) and Fig. 6(c) (PSNR=17.42dB), and Fig. 6(b) and Fig. 6(d) show the correspondingly tampered area in block level.

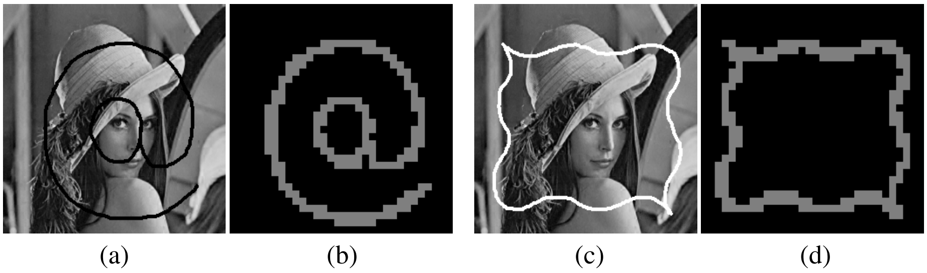


Fig. 6. Detect watermarked image with global tampering

5 Conclusion and Future Work

In this paper, we enhance an existing watermarking scheme in terms of reducing computing overload and increasing effective watermarking capacity. Based on the enhanced scheme, a new watermarking scheme in the encrypted domain with flexible watermarking capacity is proposed. It improves the robustness of watermarked image against JPEG compression after decryption and enables image tampering detection with blind watermark extracting. Experimental results demonstrate that the new watermarking scheme in the encrypted domain outperforms the previous scheme [5]. Therefore, the proposed watermarking scheme in the encrypted domain is a suitable solution to implementing existing watermarking protocols. Since bit errors are inevitable in certain scenarios, it is necessary to employ effective error correcting codes to solve this problem and improve robustness against other severe attacks in the future. We will design new secure watermarking protocol for copyright protection using the proposed watermarking scheme in the encrypted domain.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 90304008, the College and University Subject of Doctors Specific Scientific Research Foundation of China under Grant 2004071001, and the Graduate Innovation Fund of Xidian University 05017 and 05019. This work was partially supported by the

National Natural Science Foundation of China under Grant 60672129, the National 863 Project of China under Grant 2007AA01Z182.

References

1. Qiao, L., Nahrstedt, K.: Watermarking schemes and protocols for protecting rightful ownerships and customer's rights. Journal of Visual Communication and Image Representation 3, 194–210 (1998)
2. Memon, N., Wong, P.W.: A buyer-seller watermarking protocol. IEEE Trans. Image Processing 4, 643–649 (2001)
3. Lei, C.-L., Yu, P.-L., Tsai, P.-L., Chan, M.-H.: An efficient and anonymous buyer-seller watermarking protocol. IEEE Trans. Image Processing 12, 1618–1626 (2004)
4. Zhang, J., Kou, W., Fan, K.: Secure buyer-seller watermarking protocol. IEE Proceeding of Information Security 1, 15–18 (2006)
5. Kuribayashi, M., Tanaka, H.: Fingerprinting protocol for images based on additive homomorphic property. IEEE Trans. Image Processing 12, 2129–2139 (2005)
6. Trappe, W., Wu, M., Wang, Z.J., Liu, K.J.R.: Anti-collusion fingerprinting for multimedia. IEEE Trans. Signal Processing 4, 1069–1087 (2003)
7. Solanki, K., Jacobsen, N., Madhow, U., Manjunath, B.S., Chandrasekaran, S.: Robust Image-Adaptive Data Hiding Using Erasure and Error Correction. IEEE Trans. Image Processing 12, 1627–1639 (2004)
8. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
9. Okamoto, T., Uchiyama, S.: A New Public-Key Cryptosystem as Secure as Factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 308–318. Springer, Heidelberg (1998)
10. Wallace, G.K.: The JPEG still picture compression standard. Communications of the ACM 4, 30–44 (1991)
11. Kutter, M., Petitcolas, F.A.P.: A fair benchmark for image watermarking systems. In: FMCO 2004, vol. 3657(1), pp. 25–27 (1999)

Security Enhancement of a Flexible Payment Scheme and Its Role-Based Access Control

Chin-Chen Chang[1,2], Yi-Fang Cheng[2], and Iuon-Chang Lin[3,⋆]

[1] Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan
ccc@cs.ccu.edu.tw

[2] Department of Computer Science and Information Engineering, National Chung Cheng University, Chaiyi, Taiwan

[3] Department of Management Information Systems, National Chung Hsing University, Taichung, Taiwan
iclin@nchu.edu.tw

Abstract. Recently, Wang, Cao, and Zhang proposed a practical and anonymous payment scheme. In the scheme, the authors claimed that their scheme can identify those who spend a coin more than once. That means the scheme can verify the payments in an offline batch process and prevent a consumer from double spending. In this paper, we show that Wang, Cao, and Zhang's scheme can not identify consumers those who spend the same coin repeatedly in two or more different shops at the same time. So, all consumers can apply the security flaw to perform double spending successfully. In order to overcome this security flaw, we provide an improved version of the scheme in this paper.

1 Introduction

Digital cash can be divided into various categories, one of which is online system [2,5]. It requires that a shop communicates with certain bank in every payment. The shop can terminate transaction if there appears any problem. However, this requirement increases the computation and communication loads of the bank. Unlike online system, offline system [3,6] needs no communication between the bank and the shop during every payment. Nevertheless, the bank can only find out the consumer's identity, but has no way to stop their cheating. Anonymity is another subject of digital cash. The private information of customers is preserved from disclosing when they use digital cash to purchase.

Recently, Wang, Cao, and Zhang [4] proposed a digital cash scheme with flexible anonymity. For simplicity, we call their scheme the WCZ scheme hereafter. WCZ scheme allows the consumer to have an option to either pay with or without anonymity. In order to provide flexible anonymity, the authors of WCZ scheme proposed a new payment model, in which they take the traditional offline

⋆ The corresponding author.

payment model as basic foundation and meanwhile adopt the AP(Anonymity Provider) agent. With the help of the AP agent, the consumer is able to increase his anonymity at his/her will.

However, it can not be denied that the essential problem of an offline payment system is double spending. The authors claimed that although they can not prevent double spending, they can detect whether there occurs double spending. Moreover, identities of those who have performed double spending will be revealed. Unfortunately, in some circumstances, not all identities of dishonest consumers are successfully exposed, and as a result, the bank and the shop will not know the ones who have spent a coin more than once.

In this paper, we point out the security weakness of WCZ scheme and provide an improved version to overcome the weakness. The rest of this paper is organized as follows. In Section 2, we review WCZ scheme. In Section 3, we point out the weakness of their scheme. Then, the improvement of their scheme is proposed in Section 4. Finally, we make our conclusions in Section 5.

2 A Review of the WCZ Scheme

WCZ scheme [4], consists of four phases, withdrawal phase, anonymity scalability phase, payment phase, and deposit phase. Here, some system initializations are necessary to be explained before we perform the payment process. Firstly, the bank has to choose two primes, p and q, for a specified constant δ and a specified small integer γ, which satisfies $|p-1| = \delta+\kappa$ and $p = \gamma q+1$, where κ is an integer. Then a generator g of subgroup G_q with order q of the multiplicative group Z_p is defined. The bank also processes a private key $x_B \in Z_q$ and its corresponding public key Y, such that $Y = g^{x_B} \ mod\ p$. Moreover, a collision intractable hash function H is defined. The bank then publishes p, q, g, H, and the public key Y. On the other hand, the consumer generates a secret key $x_u \in Z_q$ and then the bank associates the consumer with his/her identity I, where $I = g^{x_u} \ mod\ p$.

2.1 Withdrawal Phase

In this phase, the consumer withdraws coins from the bank.

Step 1: Suppose that the consumer's identity is $I = g^{x_u} \ mod\ p$. The consumer can generate a coin c,

$$c : \{a = g^r, b = Y^r I^s\},$$

where r is randomly chosen from Z_q and s is the secret key of the coin.

Step 2: The consumer generates a signature of the message on c, together with date, etc. by using Schnorr's signature scheme with the consumer's private key x_u. Then the consumer sends the signature, the date, the parameters c, r, and I to the bank.

Step 3: After receiving these messages from the consumer, the bank verifies the signature by using the consumer's public key. Since only the actual consumer is able to generate a valid signature, the bank is convinced that the ownership of the coin goes to the consumer. Then, the bank modified the consumer's account and sends back a certificate of c, called $Cert_c$, to the consumer.

Step 4: The consumer has to carefully record r, s, and $Cert_c$.

2.2 Anonymity Scalability Phase

In this phase, the consumer can achieve a higher level of anonymity with the aid of AP agent. Certainly, if the consumer is not willing to be anonymous, he/she could skip this phase.

Step 1: The consumer randomly chooses a number ρ and reencrypts his/her coin into

$$c' : \{a' = g^{\rho} a, b' = Y^{\rho} b\}.$$

Step 2: The consumer also generates a signature on m (where $m = h^{\rho}$), together with date, etc. by using Schnorr's signature scheme and the consumer's private key x_u. Then the consumer sends the signature, the date, the parameters c, c', and m to the AP agent.

Step 3: The consumer also provides a designated verifier proof of the equality of discrete logarithms, $log_h m = log_g(a'/a) = log_Y(b'/b)$.

Step 4: The AP agent checks the certificate $Cert_c$, and also verifies the received signature. If the verification is positive, the AP agent generates a new certificate $Cert_{c'}$ on c' and transmits it to the consumer.

Step 5: To prove the relationship between c and c', the AP agent keeps c, c', m, and S in its database. As for the consumer, he/she then replaces $Cert_c$ with $Cert_{c'}$.

2.3 Payment Phase

In this phase, the consumer proves his knowledge of the value x_u and s. To show his knowledge of x_u and s, the consumer only has to generate a signature. Besides, if the consumer wants to pay with anonymity, he/she must use anonymous coin c'; otherwise he/she uses coin c instead. Here we suppose that the consumer has intention to pay with anonymity. The detailed process are described as below.

Step 1: The consumer selects a random number $k \in Z_p$, and computes

$$t = Y^k g^s \; mod \; p, and \tag{1}$$

$$e = H(m, t), \tag{2}$$

in which m is a mixed message of c', current time.

Step 2: The consumer computes

$$u = k - re \; mod \; p, \tag{3}$$

$$v = s - x_u e \; mod \; p, and \tag{4}$$

$$t_1 = g^{(s-1)x_u e} \; mod \; p. \tag{5}$$

Step3: The consumer sends the signature $S : \{e, u, v, t_1\}$, together with (m, t), to the shop.

Step 4: After receiving signature S and (m, t) from the consumer, the shop computes $t' = Y^u g^v b'^e \ mod\ p$ and checks the validity of the payment, computing whether $t' = tt_1 \ mod\ p$ and $e = H(m, t'/t_1)$ or not. If the verification is positive, the shop then accepts the payment.

2.4 Deposit Phase

In this phase, the shop deposits the money by sending the payment transcript to the bank. The payment transcript consists of a coin c or c', the signature S, and the date/time of the transaction. After verifying the correctness of the payment transcript, the bank credits the shop's account.

2.5 Detecting Double Spending

If a coin is spent twice, there will be two signatures, s_1 and s_2, where $s_1 = (e_1, u_1, v_1, t_{11})$ and $s_2 = (e_2, u_2, v_2, t_{12})$. The bank computes

$$x_u = (v_2 - v_1)/(e_1 - e_2). \tag{6}$$

Then the secret key x_u and its corresponding identity $I = g^{x_u} \ mod\ p$ of the consumer is revealed. Thus, the bank can identify who actually spends a coin more than once.

3 Comments on the WCZ Scheme

In [4], the authors proposed a practical and flexible digital cash scheme with anonymity and claimed that their scheme is offline verification and can identify those who perform double spending. In their scheme, they assume that if a dishonest consumer uses a coin c twice, then the bank will have two different signatures. In fact, the assumption is not true. In some circumstances, the consumer can still use the same coin more than once and the associated signatures are identical. The consumer can perform double spending successfully, because the bank can not identify the identity of the dishonest consumer. In this section, we point out the security weakness of the WCZ scheme.

Suppose that a consumer simultaneously performs payments in two or more different shops. In this circumstance, the consumer can use the same coin in all payments and the identity will not be disclosed in double spending checking.

For example, we suppose that a consumer makes a purchase at shop A and shop B at the same time. The consumer follows the steps of payment phase to complete the transactions. First, the consumer chooses a random k and generates a signature $S : \{e, u, v, t_1\}$ and (m, t) by using Equations 1, 2, 3, 4, and 5, where m is a mixed message of c' and the date/time of the transaction. Then the consumer sends the signature S, together with (m, t), to shop A. And meanwhile, the consumer also sends the signature S together with (m, t) to shop B. Both shops A and B will accept the payments because the payment system is offline and the verifications of payments are positives.

After that, both shop A and shop B transmit their payment transcripts to bank for batch processes. Having searching its database, the bank will notice that the coin c' is used twice. However, the bank has no way to trace the identity of the dishonest consumer according to the used coin, because the coin is associated with two identical signatures, $S : \{e, u, v, t_1\}$. Therefore, $(v - v)/(e - e) = 0$, and no information will be revealed performing this equation.

4 Improvement

In order to avoid the weakness of the WCZ scheme, we modify the WCZ scheme to enhance the security.

In the improved version, the withdrawal phase, anonymity scalability phase, deposit phase, and the process of detecting double spending are the same as the WCZ scheme. Only the payment phase is modified. The improved payment phase is described as follows.

Step 1: The consumer selects a random number $k \in Z_p$, and computes

$$t = Y^k g^s \ mod\ p, and$$
$$e = H(m, t, ID_s),$$

where ID_s is the individual identity of the shop.

Step 2: The consumer computes

$$u = k - re \ mod\ p,$$
$$v = s - x_u e \ mod\ p, and$$
$$t_1 = g^{(s-1)x_u e} \ mod\ p.$$

Step3: The consumer sends the signature $S : \{e, u, v, t_1\}$ together with (m, t) to the shop.

Step 4: After receiving signature S and (m, t), the shop computes $t' = Y^u g^v b'^e$ $mod\ p$, and then uses t' and the shop's identity ID_s to check the validity of the payment through computation of $t' = tt_1 \ mod\ p$ and $e = H(m, t'/t_1, ID_s)$. If the above equations holds, the shop then accepts the payment.

The improvement can prevent the dishonest consumer from using the same coin in different shops at the same time. In addition, due to the the payment signature S includes the shop's identity, it proves that the consumer has traded with the shop using this coin, which, on the other hand, can also prevent the dishonest shop from transmitting the duplicated payment transcripts to another shop for depositing.

5 Conclusions

In this paper, we have indicated the weakness of the WCZ scheme, and accordingly, proposed an improved version to enhance the security of the WCZ scheme. This improvement not only can avoid double spending from the dishonest consumers, but also can prevent dishonest shops from depositing repeatedly. And meanwhile, all the advantages mentioned in the WCZ scheme are kept.

References

1. Chaum, D.: Blind Signature for Untraceable Payments. In: Advances in Cryptology-Cryoto 1982, pp. 199–203. Springer, Heidelberg (1983)
2. Chaum, D.: Security Without Identification: Transaction Systems to Make Big Brother Obsolete. Communications of the ACM 28(10), 1030–1044 (1985)
3. Chaum, D., Fiat, A., Naor, M.: Untraceable Electronic Cash. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 319–327. Springer, Heidelberg (1990)
4. Wang, H., Cao, J., Zhang, Y.: A Flexible Payment Scheme and Its Role-Based Access Control. IEEE Transactions on Knowledge and Data Engineering 17(3), 425–436 (2005)
5. Damgard, I.B.: Payment Systems and Credential Mechanisms with Provable Security Against Abuse by Individuals. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 203–216. Springer, Heidelberg (1990)
6. Okamoto, T., Ohta, K.: Disposable Zero-Knowledge Authentications and Their Applications to Untraceable Electronic Cash. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 134–149. Springer, Heidelberg (1990)

Building Trusted Sub-domain for the Grid with Trusted Computing

Jing Zhan[1], Huanguo Zhang[2], and Fei Yan[1]

[1] School of Computer Science, Wuhan University, Wuhan, 430079, China
{jingzhan_whu,yfpost}@yahoo.com.cn
[2] State Key Lab of Software Engineering (Wuhan University), Wuhan, 430072, China
liss@whu.edu.cn

Abstract. The Grid is all about collaboration, which is supported by dynamic, multi-institutional virtual organizations (VO). The fact that Grid users and resource providers often suffer from attacks outside or inside the VO make it necessary to build a trusted sub-domain. The TCG (Trusted Computing Group) proposes Trusted Computing (TC) to enhance users' trust on today's open architecture platforms by adding a tamper-resistant hardware module called Trusted Platform Module (TPM) to the end system. In this paper, we propose and design an open-source security system based on Linux and TPM hardware to extend the trust on the platform to the Grid environment, and hereby provide sharing of trusted environment. Especially, we demonstrate how to build a trusted sub-domain for the Grid with our system by using trusted attestation and migration based on the TC.

Keywords: Grid Security, Trusted Sub-Domain, Trusted Computing (TC), Trusted Attestation, Trusted Migration.

1 Introduction

The Grid is a large scale distributed computing system for collaboration among dynamic, multi-institutional virtual organizations (VO), which may be across numerous administrative domains and subject to their own various security, management and usage policies [1].

Although the current Grid security solution, Grid Security Infrastructure (GSI) [11] for Globus Toolkit (GT) [10], offers comprehensive security services by applying public-key cryptography, cryptographic protocols methodologies and necessary infrastructure supporting services, it still need place more strict constraints on VO members to protect them from threats outside or inside the VO.

Trusted computing (TC) is a new trend of information security field [4]. The Trusted Computing Platform Alliance (TCPA, now is known as Trusted Computing Group, TCG) adopts a practically available way to provide trust on general computing platform. The TC uses a secure coprocessor called Trusted Platform Module (TPM) [14] to measure and report the way the platform is operating. With the tamper-resistance property of the TPM, the VO leader can deploy it for the Grid to attest VO members' platform with expected policy without worrying about interference from software.

This paper presents an open-source system, designed and developed by the authors and teammates to provide a trusted sub-domain for the Grid using trusted attestation and migration, which enables VO members to use resources protected by GSI in a secure, consistent manner with Trusted Computing feature. Section 2 describes the limitations of current Grid security practice and explains why the development of a trusted sub-domain for the Grid is necessary. Section 3 presents key technologies from TC used in design of our work. Our system is introduced in detail in Section 4, along with a description of how it works in Grid environment. Section 5 gives a discussion of security considerations with the system and related works, and section 6 comes to our conclusion and future work.

2 Current Grid Security Practice

The Grid security practice should should facilitate cooperative problem-solving based on VO which consists of dynamic components across multiple domains.

In the general Grid environment, the cooperation among VO members are protected by Grid Security Infrastructure. The GSI implements many standards and specifications to provide the fundamental security features such as integrity, confidentiality, authorization and authentication through the Public Key Infrastructure (PKI).

However, GSI provides no effective way to detect the misuse and compromise of users' data and computations by a system administrator at a his/her platform, which brought about the need for a system with tamper-proof attestation feature and provide a trusted sub-domain for the Grid. Trusted Computing can help enable conformable constraints on VO members. For instance, we can define an attestation attributes-based VO policy as the cooperation can only be available with the conformity to our requirements.

3 Key TC Technologies for the Grid Environment

This section introduces general concept and security services based on TCG capabilities that we implemented for requirements described in section 2, namely trusted attestation and migration, which base on transitive trust on Trusted Platform (TP). For a full description, see for examples in TCG specifications [13], [14], [15], [16].

The core concept of the TC is to use the TPM to achieve the transitive trust on TP. This is currently done through integrity measurement of the boot process of the platform.

Each and every component during system startup has to check the next booting component's metrics with TPM and pass on the execution. The TPM in turn calculates each component's hash value and stores it inside its platform configuration registers (PCRs) as integrity metrics. All the values once calculated and stored in every PCR are stored ordinarily as PCR entries outside TPM for future attestation of the current platform state.

3.1 Trusted Attestation

For verification purposes, a remote party can query the measurement of the TP by means of attestation. Trusted attestation is an operation that provides proof of a set of the platform's integrity measurements [14], [16], which provides remote users with the evidence of platform properties. This is done by signing one or several PCRs using an Attestation Identity Key (AIK) in the TPM as showed in Fig. 1.

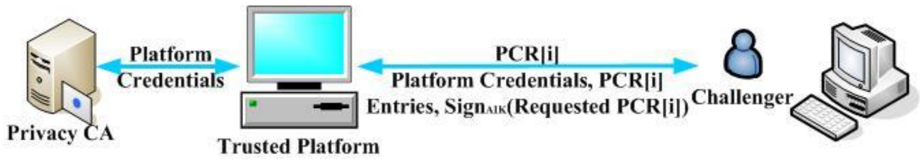

Platform Credentials
Privacy CA
Trusted Platform
PCR[i]
Platform Credentials, PCR[i] Entries, Sign_AIK(Requested PCR[i])
Challenger

Fig. 1. Trusted Attestation

First, Privacy CA and TPM work together to get an AIK and corresponding credentials. Then the user asks TP for needed PCRs. TP sends PCR values signed by private AIK, PCR entries of corresponding PCR and Platform Credentials back to user. After verifies Platform Credentials, user can decide whether this platform can be trusted. If nothing goes wrong, user can use public key of AIK decrypt the current hash values stored in the PCRs. The user can now check what software the remote system is running because each state on the PCR entries list has a unique hash value. By comparing the decrypted PCR and results of calculation of PCR entries list, user can find out whether requested state of platform can be trusted: if the list is incomplete or being tampered, the reported PCR value will not match the hash value as recalculated by user. In this way, the TP attests that its platform environment can be trusted and shared by other users for collaboration.

3.2 Trusted Migration

The Migration capability provided by the TCG is intended for backup, upgrade or to clone of keys. In this paper, we carried out trusted migration as a way to provide the Grid with a trusted sub-domain with conformed VO policy.

Fig. 2 described the migration process. In order to migrate Alice's key to Bob's TPM, Alice's the TPM needs to create a data blob, namely MigrationBlob, containing the encrypted MigratedKey that Bob's TPM can decrypt. This is done like this: Alice loads in a public key of StorageKey from Bob's platform by her TPM to create MigrationBlob for her MigratedKey and send it to Bob; Bob use the private key of his StorageKey to decrypt and get the MigratedKey generated by Alice's TPM.

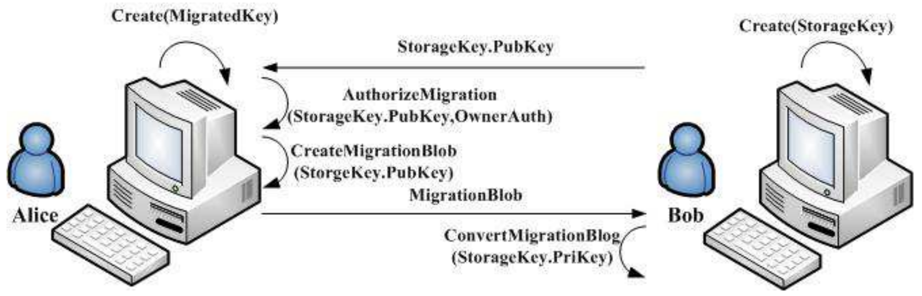

Create(MigratedKey)
Create(StorageKey)
StorageKey.PubKey
AuthorizeMigration (StorageKey.PubKey,OwnerAuth)
CreateMigrationBlob (StorgeKey.PubKey)
MigrationBlob
ConvertMigrationBlog (StorageKey.PriKey)
Alice
Bob

Fig. 2. Trusted Migration

4 Our Work

In this section we describe the system that we designed and developed in order to meet the requirements as laid out in section 2. We first describe the architecture of our work. Then we show how the system provides trusted sub-domain for the Grid. Finally we describe our implementation status.

4.1 Architecture

Fig. 3 shows an architectural description of Trusted Computing enabled grid security solution.

We adopt TC technology to enhance Grid security infrastructure, namely the gray components in the diagram. And all the work is below the GSS API provided by the Globus toolkit. It relies upon the credential and certificate utility APIs for general certificate acquisition and inspection functionality, which means for grid applications that needs to be transplanted from legacy applications to TCG-enabled ones; it need not to modify the applications.

Our work can be vertically divided into two parts: Special Security Module for Grid and TSS (Trusted Software Stack) related work. The Special Security Module for Grid provides Trusted Computing features with extended GSS-API. TSS related part is to interact with TPM hardware and utilize the cryptographic function of TPM.

Original GSI get cryptographic service by calling standard Crypto API. Cryptographic service providers (CSP) are implementations of crypto algorithms which can be in either software (running in the general CPU) or hardware (running in e.g., a smartcard or a USB token). Also CSPs can be implemented by the TPM hardware (Crypto API call the TCG Crypto Security service), which is one work of the GSI related part.

TPM. This is the central hardware module in TCG. It provides several important low-level functions including input/output interface, non-volatile storage, integrity attestation, random number generator, HASH engine, key generation, and so on.

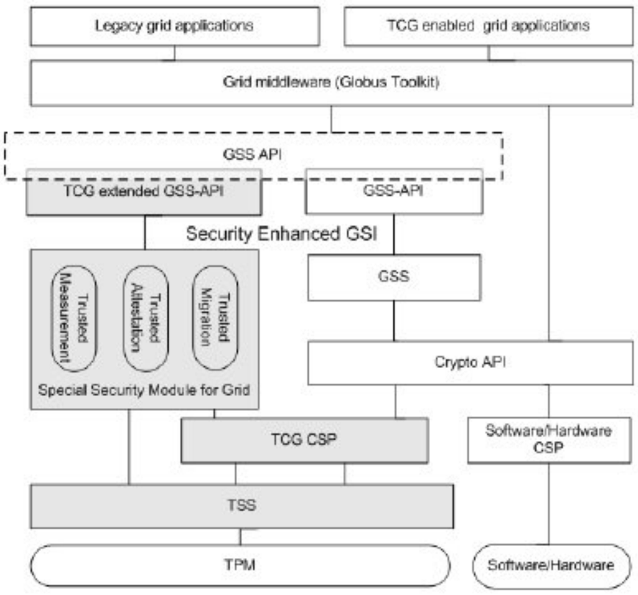


Fig. 3. Architecture of our system

TSS. Users require trusted hardware to offer enhanced secure solution as well as software middleware to conceal heterogeneous applications' requirement. We have developed a common standard API for applications to utilize the TCG services according to TCG specification. The three special security modules for Grid are based on the TSS.

Trusted Measurement Module. This module gets the accurate integrity metrics of the platform's environment state which are kept inside the TPM's PCR and tests whether they match the computation results of measurement values (called PCR events) based on the TSS.

Trusted Attestation Module. This module can provide the proof that the VO members works basing on required attestation attributes with attestation protocol (as elaborated in section 3.1).

The attestation attributes represent trusted environment of the platform are collected from trusted measurement module. In our system, they include integrity metrics on local platform and Grid Job related requirements, which is CPU and RAM performance.

Trusted Migration Module. GSI has proposed proxy to achieve an automatic delegation of resource request. However, the proxy credential is stored in the file space after the VO building has been done. To mitigate the potential danger of key compromise, GSI stipulates that a proxy credential should have a short lifetime.

In our system, we suppose the proxies are TPM equipped platforms. The VO leader starts the VO building and acts as a migration authority, to have the private key of his/her cryptographic credential migrated by trusted migration module. With TPM protection of the proxy credential, short lifetime stipulation for the proxy credentials becomes unnecessary. As the key in the proxy credential is often related with authorization status of Grid resources, the trusted migration module can hereby offer strong protection on resources sharing policy.

4.2 Building Trusted Sub-domain with a Grid Security Enhanced System

Here is our Grid Security Enhanced System providing Trusted Sub-Domain with expected VO policy. The workflow is described in Fig. 4. It only demonstrates the situation containing two VO members, but can be extended to multiple members easily.

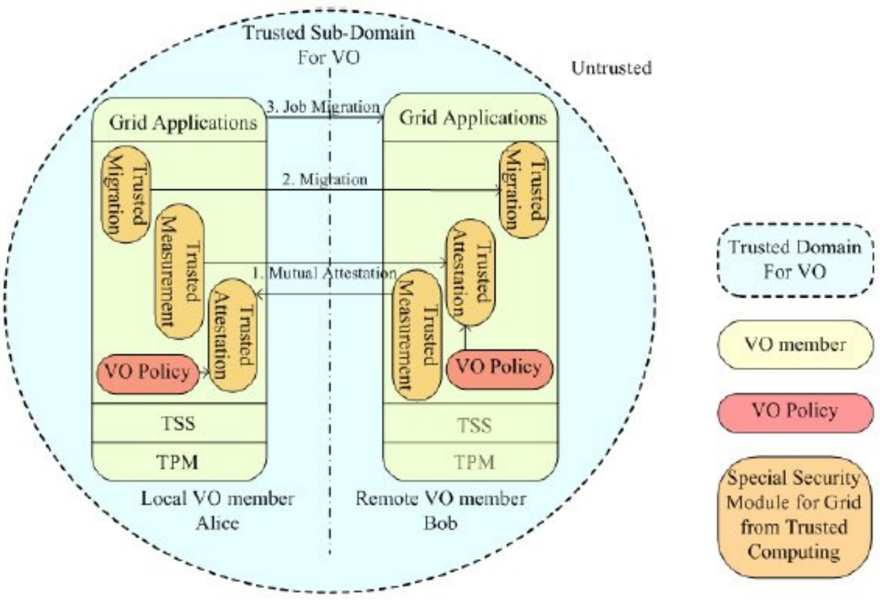


Fig. 4. Trusted Sub-Domain for Grid environment.When VO building is done, the VO becomes a trusted sub-domain. Alice can migrate Grid job to Bob.

Alice and Bob are two VO members. Suppose Alice starts the VO and invites Bob. Our demo system comes to two steps:

Mutual Attestation. Alice use trusted attestation module to get attributes form trusted measurement module on Bob's platform. The attestation attributes are decided by the VO policy requirements. If the attributes don't meet the requirements, Alice wouldn't invite Bob.

Then Bob get attestation attributes from Alice. If they conform to the policy requirements, Bob would accept Alice's invitation.

Trusted Migration. After mutual attestation, Alice can migrate her proxy credential to Bob and get offline, for Bob can migrate the proxy credential to others now.

4.3 Implementation Status

We have planned to make our system an open-source system. The implementation has greatly benefited from the open-source Trusted Software Stack (TSS) package TrouSerS [17], Open- SSL library [12] and the open-source grid middleware package GT4.

Our system has been completed for both the TPM chip version 1.1b and 1.2 manufactured by Infineon on a number of HP laptop platforms, Ubuntu 6.06 "Dapper Drake" with Linux kernel 2.6.15. We have already accomplished elementary demo system [9]. At the time of writing this paper, we have been under full implementation version.

5 Discussion and Related Works

As we discuss above, the goal of our system is to provide a trusted sub-domain in Grid environment with the TC. Our system contains two phases to achieve these goals.

The first phase is mutual attestation. VO members can decide whether to invite potential candidates basing on the attestation attributes, namely desired PCR-Setting values(e.g., measurements of BIOS, OS, applications) protected by the TPM.

The second phase is proxy migration. As the private key of proxy credentials from the original TPM is encrypted by keys from the other TPM during the migration, the migration is actually under the protection of the crypto chip TPM and can't be tampered. In a TP, no super-privilege users of Operating System but TPM users who hold secret shared with the TPM can access its special resource. Our system will decrease the risk of private key disclosure.

There are already many researches based on trusted hardware such as [5], [6], [7], [18], and secure computing environment [19]. Xen Community, Intel and IBM have been working together to provide TPM virtualization for Xen [8], and the work can be used for TPM-sharing for a group of users. [23] has adopted the TC on virtual machine-based platform. All these efforts needs completely change in current open platform architecture.

[5] has developed a Linux-based integrity measurement scheme for TCG, while [20] implements it with Linux Security Modules (LSM) patch for newer kernel versions 2.6.20, which can be used in our architecture to help form the intact transitive trust with Linux kernel.

A delegation mechanism named MyProxy [2] proposed in Grid computing environment. In [3], M. Lorch et al. have proposed an enhanced a credential repository for Grid PKI secured by IBM4758, which can be built based on our proposal too, with much cheaper commercially available TPM.

Smith et al. [21] have closely examined the scenarios for which TC techniques are necessary and useful and the scenarios for which traditional protection mechanisms suffice.

Most of researches related with Grid and trusted hardware focus on PKI and TPM virtualization problem. Our work has promoted a novel solution for providing a trusted sub-domain for the Grid.

6 Conclusion and Future Works

The TCG-compliant PCs are already available for purchase. To date, the TCG specifications have only specified in full how to build a TCG-compliant PC and enhance security for host platform, the TCG features of transitive trust, attestation and migration can apply to other platforms and environment.

In this paper, we apply core concepts of trusted computing technology to the Grid. Based on TC, we define attestation attributes as VO policy through PCR values which is rather rough now. The next step would be enforcing a fine granularity policy for the Grid.

For now, our work mainly works on the TSS and the grid middleware layer without touching the operating system. We'd like to work on future development in that direction too.

Acknowledgments. We are extremely grateful to Dr. Wenbo Mao of HP Labs China for his help in Daonity system design. Liqiang Zhang, Lihua Song, Qi Sun, Miao Zhang, Lei Zhang, Lihua Song (of Wuhan Univ.), Chuanjiang Yi, Weizhong Qiang, Chunren Chen, Wenbo Yan and Shuming Liu (of HUST) participated in the implementation work. This research has been sponsored by HP Labs China and supported by the National Natural Science Foundation of China (60373087, 60473023, 60503040, 90104005 and 60673071) and National Hi-Tech Research and Development Program of China (2006AA01Z442). We would also like to thank anonymous reviewers of The 3rd International SKLOIS Conference on Information Security and Cryptology for important review feedbacks.

References

1. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the Grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications 15(3), 200–222 (2001)
2. Novotny, J., Tueke, S., Welch, V.: An Online Credential Repository for the Grid: MyProxy. In: Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, Los Alamitos (2001)
3. Lorch, M., Basney, J., Kafura, D.: A Hardware-secured Credential Repository for Grid PKIs. In: 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (April 2004)

4. Smith, S.W., Dengguo, F., Zhen, X., Liwu, Z.: Trans. Trusted Computing Platforms: Design and Applications, pp. 14–15. Tsing Hua University Press, Beijing (in Chinese) (2006)
5. Sailer, R., Zhang, X., Jaeger, T., Van Doorn, L.: Design and Implementation of a TCG-based integrity measurement architecture. In: Proceedings of the 11th USENIX Security Symposium (August 2004)
6. Dyer, J., Lindemann, M., Perez, R., Sailer, R., Smith, S.W., van Doorn, L., Weingart, S.: Building the IBM 4758 Secure Coprocessor. IEEE Computer 34, 57–66 (2001)
7. Smith, S.: Outbound Authentication for Programmable Secure Coprocessors. In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ESORICS 2002. LNCS, vol. 2502, pp. 72–89. Springer, Heidelberg (2002)
8. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Proceedings of the 19th Symposium on Operating Systems Principles (SOSP 2003), pp. 164–177 (2003)
9. Trusted Computing Research Group. Open Grid Forum, `http://forge.gridforum.org/projects/tc-rg/`
10. Globus Toolkit 4, `http://www-unix.globus.org/toolkit/`
11. Open Grid Forum. Overview of the GSI, `http://www.globus.org/security/overview.html/`
12. OpenSSL, `http://www.openssl.org/`
13. Trusted Computing Group, TCG Specification Architecture Overview (April 2004), `http://www.trustedcomputinggroup.org/specs/IWG/TCG_1_0_Architecture_Overview.pdf`
14. Trusted Computing Group, TPM Main: Part 1 Design Principles (March 2006), `http://www.trustedcomputinggroup.org/specs/TPM/Main_Part1_Rev94.zip`
15. Trusted Computing Group, TCG Infrastructure Working Group Reference Architecture for Interoperability (June 2005), `http://www.trustedcomputinggroup.org/groups/infrastructure/IWG_Architecture_v1_0_r1.pdf`
16. Trusted Computing Group, TSS_Version_1.2_Level_1_FINAL (January 2006), `http://www.trustedcomputinggroup.org/specs/TSS/TSS_Version_1.2_Level_1_FINAL.pdf`
17. Watson Research I.B.M - Global Security Analysis Lab: TCPA Resources (April 2006), `http://sourceforge.net/projects/trousers`
18. LaGrande Technology Architectural Overview (September 2003), `http://www.intel.com/technology/security/`
19. Microsoft, Next-Generation Secure Computing Base home page, `http://www.microsoft.com/resources/ngscb`
20. IBM Integrity Measurement Architecture (March 2007), `http://sourceforge.net/projects/linux-ima`
21. Smith, M., Friese, T., Engel, M., Freisleben, B.: Countering security threats in service-oriented on-demand grid computing using sandboxing and trusted computing techniques. Journal of Parallel and Distributed Computing 66(9), 1189C1204 (2006)
22. Sailer, R., Jaeger, T., Zhang, X., van Doorn, L.: Attestationbased policy enforcement for remote access. In: Proceedings of the 11th ACM conference on Computer and communications security (CCS 2004), pp. 308–317. ACM Press, New York (2004)
23. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: A virtual machine-based platform for trusted computing. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (2003)

Enhanced Security by OS-Oriented Encapsulation in TPM-Enabled DRM

Yongdong Wu[1], Feng Bao[1], Robert H. Deng[2], Marc Mouffron[3], and Frederic Rousseau[3]

[1] Institute for Infocomm Research, Singapore
{wydong,baofeng}@i2r.a-star.edu.sg
[2] Singapore Management University, Singapore
robertdeng@smu.edu.sg
[3] EADS Secure Networks, France
{marc.mouffron, frederic.rousseau}@eads.com

Abstract. The Trusted Computing Group (TCG) defines the specifications for the Trusted Platform Module (TPM) and corresponding trust mechanisms that allow a TPM-enabled platform to run only authenticated software. For example, the operating system (OS) can use the facilities provided by the TPM to authenticate a Digital Rights Management (DRM) application before allowing it to run. However TCG does not provide any clear specification on what kind of software can be regarded as trusted and hence be authenticated. In fact it is unlikely that there will be a clear line between the software that should be authenticated and those should not, *e.g.*, debugger for developing binary codes and Internet browser for running applets. This leaves a grey area where even authenticated software may be exploited for malicious usage. This paper investigates the security of DRM applications in a relaxed scenario where users have larger purview. We present two attacks: abuse attack and injection attack where some reasonably authenticated software can be exploited for stealing protected contents. In the abuse attack, an attacker uses an authenticated debugger to monitor the internal state of a DRM application for the purpose of violating the access privilege in the application. In the injection attack, an adversary is able to make malicious modifications on an original DRM application at will. These two attacks demonstrate that it is not straightforward to impose DRM in a TPM-enabled system. To counter the attacks, we provide the OS-encapsulation scheme which ensures that only the genuine OS can start the DRM application. Our scheme is an enhancement of security for TPM-enabled DRM in a loose but more practical environment, where people are allowed to use the debugger, web browser, etc.

1 Introduction

General-purpose computer platforms provide the convenience for developing and running software applications. However, it also allows a malicious user to tamper with and possibly control any software component at will. Software tampering not only affects the software industry in general, it can also impede content delivery services and compromise enterprise systems. Hence, many software vendors employ software protection technologies to counter such threats. Unfortunately, it has been generally accepted in the information security research community that software alone cannot provide an adequate foundation for building a high-assurance trusted application [1]. Hardware-assisted protection [2] is a promising solution for strong software security.

The Trusted Computing Group (TCG) [3] defines the specification for a security chip called the Trusted Platform Module (TPM) and its related software interfaces. In the TCG specification, the TPM is designed to provide end-user machines with a minimal but essential hardware base for end-user side security. For instance, there are TPM-enabled schemes in enhancing the security of smartcard [4], attestation [5,6], and privacy protection [7,8]. Since it is impossible to modify the TPM-enabled OS without being detected, the TPM can be employed to defeat sophisticated attacks such as the substitution attack [9,10] which invalidates software integrity protections (e.g., [11,12]). In addition, Felten [13] highlights the possibility that a DRM application can utilize the TPM to protect data such that only cooperating applications are able to consume the protected content.

Although the TPM can potentially be used in many applications, its protection mechanism is still not satisfactory [13,14,15,16]. For instance, Anderson [14] criticized TPM-based DRM applications as treating users in an unfriendly way. Additionally, in a DRM application, a traitor[1] may misuse a genuine software to compromise the built-in access control in the application.

This paper investigates the security of the TPM-based DRM, and proposes the abuse attack and the injection attack. The abuse attack is initiated by a malicious user within a trusted environment. The user makes use of a trusted application such as a debugger to mediate the interactions between the victim program and the OS. Given that a user is allowed to sign and certify his own program on the TPM-enabled platform, the injection attack enables him to run any program locally at his discretion. Both attacks allow the malicious user to violate the content protection objective of DRM. To counter these attacks, it is necessary to verify the owner/signer of the application, and to establish the application's authenticity in order to detect tampering and misuse. To achieve this, we propose two schemes. The first scheme is to use a time-sensitive protocol to detect the presence of the abuse attack. The second technique requires to encrypt the DRM application and start the decrypted application by the authenticated operating system only, thus prevents both abuse attacks and injection attacks. By the two attacks and their countermeasures, we demonstrate that TMP/TCP does not guarantee DRM security automatically and easily. Many issues have to be solved before we can claim that the trusted computing based DRM has better security. Especially, we need to consider the loose scenarios where users are given reasonably more authority. In practice, a very restrictive confine on software usage is hardly acceptable.

[1] We define a traitor to be a user who modifies his authorizations illegally. For example, the traitor may change his read permission into a write permission.

This paper is organized as follows. Section 2 introduces the model of TPM-enabled DRM. Section 3 describes a basic TPM-enabled DRM scheme, followed by two attacks on the scheme. Section 4 proposes our techniques to the TPM-enabled DRM scheme to counter the attacks. Section 5 concludes the paper.

2 Model of TPM-Enabled DRM

2.1 DRM Architecture

The underlying motivation for having DRM applications is that digital objects should remain under the control of their creators, rather than the owner of the platform containing the objects. One major threat to DRM is the tampering of the DRM application and its platform since a traitor can modify them so as to circumvent the DRM protection. Currently, a lot of commercial DRM systems are available in the market, such as Microsoft Windows DRM, and Adobe Web Buy. The architecture of these products and some standard such as [18] can generally be described as follows.

– An author generates digital content such as a video file.
– A DRM-enabled application will create a rights-protected version of the file and assign usage rights at the content server. A portal server provides a Web site so that the consumer can preview, purchase and download content that interests him.
– A license server will be responsible for the enrollment and certification of trusted entities, and for licensing rights-protected content.
– Finally, the consumer's DRM client software interacts with the DRM server, and renders the content according to the consumer's assigned privileges.

Although the general DRM architecture and workflow are applicable, the security depends on the assumption of a trusted environment. In lieu of a fully trusted environment, a TPM-enabled DRM application can be used to provide trust to the content owner. The assumptions for such an application is described in the next subsection.

2.2 Assumptions

Although TCG specifications describe the technology how to authenticate a software, it has never been explicitly described, or even discussed, what kind of software should be allowed to be authenticated. For the practical application of TCP, there is a big gap on how and who will be responsible for the software verification and validation. It is not going to be an easy issue. From the application viewpoint, it is not practical to exclude all the common used software tools, such as debugger and Internet browser. In this paper, we consider a not very restrictive situation of TCP and we argue that the assumption of such a situation is very reasonable and practical.

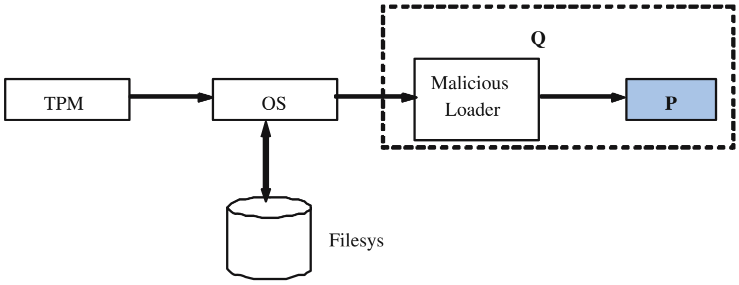


Fig. 1. System diagram. **P** is the target DRM software, and **Q** wraps **P**. **Q** will be loaded by a malicious loader such as a compromised debugger.

In the system diagram Fig. 1, TPM and OS constitute the trusted platform which is compliant with the TCG specification, and software **P** is the DRM application which must be loaded by a trusted OS or a trusted debugger. According to the DRM requirements, we make the following assumptions:

A_1: The trusted platform includes a TPM chip and a trusted OS, and supports a trust transitive process, as mentioned in Section **??**.

A_2: A genuine software application **P** is signed by its producer (or a trusted third party) who has a certificate issued by a CA. The CA's public key is trusted by the OS. To authenticate **P**, the OS will verify its signature before running it. Note that the OS does not use TPM integrity measurements on **P**, but the OS itself will have its integrity verified by way of TPM integrity measurements as mentioned in Section **??**.

A_3: A traitor is able to load and monitor the targeted program **P** in the trusted platform. For example, if an authenticated Internet browser is able to run applet, it is possible to run some DRM application **P** in the browser. Similarly, if a debugger such as `Microsoft Visual Studio` can be used in TPM environment, the traitor gains the privilege on tampering **P**.

A_4: A traitor is able to intercept/modify the messages between **P** and the OS by using a malicious loader. For example, a TPM application developer should be able to debug the his own software related to TPM functions. In other words, a malicious user can always "sit" between OS and DRM application.

A_5: A traitor can not reverse-engineer the target software **P** to gain knowledge of critical algorithms or secrets such as a cryptographic key, otherwise, the attacker can write his own software with the reverse-engineered code and then render the protected content.

A_6: Any user can insert his own root certificate into the OS. By doing this, a traitor is able to develop and sign any software and run it on the platform.

Of the above assumptions, A_1-A_2 are consistent with the TCG specifications, while A_3-A_4 are valid because the traitor owns the platform and all software running on the platform including **P**. Assumption A_5 is mandatory although it is believed to be impossible in theory. Fortunately, the software designer can increase the reverse-engineering cost with the technologies (*e.g.*, diversity, anti-tracing, anti-disassembling, self-modifying code, and code-encryption) such that the hacker gives up the attempt. To enhance the security of software, the application code is encrypted all the time except the execution stage in TPM platform. In addition, A_6 is also reasonable since

- Although it is controversial about the function of TPM, a TPM-enabled computer must have an open structure in terms of software and hardware, otherwise, why not buy a cheap TV setbox for a user?
- A TPM-enabled OS prevents a user from running an "unapproved" application, but it is unacceptable to deprive the user the right of developing his own software. Hence, the platform should allow the user to load applications that are approved by themselves.
- As argued in [15] in terms of protecting fair use rights, users should become the root of their own certification trees and authorize various devices to view purchased content.
- It is not uncommon in commodity platforms that a user is able to install his own root certificates. For example, in browser applications such as the `Microsoft Internet Explorer`, a user can import his own root certificates into the existing set of trusted root certificates.

3 Basic TPM-Enabled DRM: Scheme and Attack

Following the model in Section 2, this section introduces a basic TPM-enabled DRM idea derived from [13]. Then it presents two attacks on the basic scheme such that a traitor can obtain unauthorized access.

3.1 Basic TPM-Enabled DRM Scheme

Although a TPM-enabled platform provides a trusted environment with PCR reports to detect "unapproved" software, its application to DRM is not straightforward if security and inter-operability are important. For example, Felten [13] described the TPM-enabled software interoperation principle which can be applied to design a basic TPM-enabled DRM as follows.

- The digital file is encrypted with an authorized DRM program **P** such that the digital content is under the full control of **P**.
- If another program **Q** wants to consume the file, the only way is to request **P** to decrypt it in a TPM-enabled environment and transfer the decrypted content, but **P** maybe reject the request.
- The protected media will be decrypted on a TPM-enabled platform and used in a controllable way.

To achieve the above, **P** must be tamper-proof. As stated in assumption A_2, the OS will verify **P**′s signature before executing **P**. Hence the traitor can not violate the DRM access by tampering with the software. However, basic TPM-based DRM is vulnerable to the following abuse attack and the injection attack.

3.2 Abuse Attack

With the assumption of trustworthy hardware and software, the TPM-enabled platform ensures that each application is valid when it runs. That is, no untrusted software can execute on the platform. However, the traitor can combine two or more approved programs in a certain way to realize his malicious intent. In particular, if an authorized software can load and execute some other program, as in the case of a typical debugging tool, the traitor can use it to access protected content. Common debugging tools include `Microsoft Visual Studio` (with a built-in debugger), `SoftICE`^{TM} and `IDApro`^{TM}. If one such software is an approved software in the platform, the traitor can do the following:

– Starts the debugger. Since it is genuine and is signed by the producer, it can be started successfully. This follows from assumptions A_1 and A_2.
– Loads application **P** in the debugger. Assumption A_3 allows for this.
– Executes **P** within the debugging environment. Since the debugger has higher execution priority than **P**, the traitor is able to read all the data in the address space of **P**, including the decrypted content.

Therefore, the traitor has easily compromised the TPM-enabled DRM protection by executing the targeted application within a debugger. Presently, Internet browser such as `Internet Explorer` or `Firefox` is able to run applet, it can be misused to launch abuse attack too.

3.3 Injection Attack

According to assumption A_6, any user can insert his certificate as a root certificate into the OS. This means that any software signed by the traitor can pass the verification process described in Subsection **??**. In this case, the traitor can initiate the injection attack to compromise the content protection mechanism of the platform. Concretely, the traitor will

(1) Install his certificate as a root certificate in the OS. Assumption A_6 allows for this.
(2) Modify program **P** by injecting functions into **P** or injecting entries in **P**'s function import tables, resulting in a malicious wrapper **Q**. For example, the traitor can replace or modify the action module of menu-item "`save`", such that it dumps all the decrypted content to memory.
(3) Sign **Q** with the private key corresponding to the newly installed certificate.
(4) Run **Q** as an authorized program. Since **P** is wrapped by **Q**, **P** is started implicitly. Now, since **P** is able to decrypt the protected content, the wrapper **Q** will be able to read the decrypted content.
(5) Invoke the injected function to dump the protected content to memory. The traitor can do this by activating the tainted menu-item "`save`" for example.

In the above injection attack, the traitor wraps an authorized software **P** and produces a new program **Q**. This attack does not require the traitor to know the internal structure of **P**. He only needs to tamper with the code partially to have the decrypted content dumped to memory.

Indeed, the above two attacks can be integrated so as to violate the DRM efficiently. If a user can not install his certificate, but can initiate abuse attack, he can create the modified software **Q** and then run **Q** in the debugger or browser environment. Hence, the security strength of both the OS and DRM application should be enhanced.

A straightforward enhancement is based on PKI. Specifically, for all the TPMs produced from the same manufacturer, they have different private keys (*e.g.*, EK) but the same public key. All the DRM software should be encrypted with the public key such that the protected code can be decrypted by the trusted platform (OS+TPM) only. Its weakness is that it will incur inconvenience in case of TPM upgrading. A second simple enhancement is to verify the authenticity of the DRM application by its immediate loader[2]. Only after the loader confirms the authenticity, can the DRM software be loaded and executed. However, its side effect is that it prohibits the honest user from developing his own software, *i.e.*, violating assumption A_6.

4 OS-Oriented Encapsulation Scheme

This section elaborates an OS-oriented encapsulation scheme defines a new format of applications, and encapsulates the DRM application with the new format so as to match the assumptions $A_1 - A_6$ in Section 2 for sufficient flexibility and security.

4.1 Proposed Scheme

In the proposed scheme, we intend to meeting three objectives: (1) DRM application can be loaded by the trusted OS only; (2) allow the honest user to develop his software; (3) able to defeat injection attack and abuse attack from malicious users. Therefore, the scheme includes three modules:configuration, installation and DRM enforcement. Concretely,

Configuration. A Certificate Authorities (CAs) issues digital certificates to OS vendors and DRM vendors. Each vendor, for example Microsoft, has a pair of (public key e, private key d) for each of its OS products. The OS vendor signs the OS, and similarly, each DRM vendor also signs its own DRM application. In addition, the DRM vendor encrypts its DRM application with the public key of the OS in which the DRM application will run. The encrypted application is distributed to the users.

Installation. The installation process targets for setting up a trusted OS and DRM application in a TPM-enabled environment. Fig.2 illustrates the steps of installation. Specifically,

[2] immediate loader of **P** is the immediate parent process of **P**. For example, in Microsoft windows, `explorer` is the immediate loader of many processes such as `Microsoft winword.exe`.

– when the OS is being installed, the installation software (such as BIOS hard-code) will initiate remote attestation with the vendor server **V**. This means that the installation software will use the TPM remote attestation facilities to remotely authenticate to **V**. (This is similar to OS activation over the Internet, which is currently required by Microsoft Windows XP).
– **V** securely sends the private key d of the OS to the TPM. This private key d shall be protected by the TPM and stored in sealed storage.
– After obtaining the OS package, the TPM-enabled platform should verify its authenticity with the digital certificate of the OS vendor. Only if the OS package is authentic, OS and other OS-related modules can be installed as usual.
– To install a DRM application **P**, TPM system will obtain the ciphertext $\mathbf{C} = Enc_e(\mathbf{P})$ which is produced by the application provider with the OS public key e, and then install **C** directly without decryption.

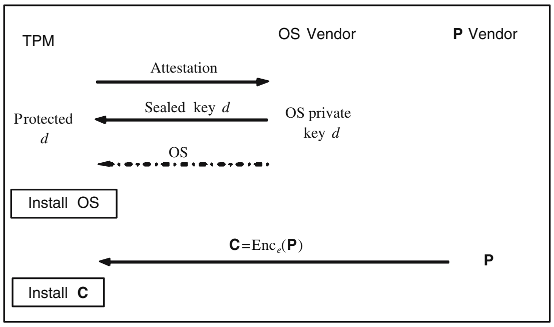


Fig. 2. Installation process. TPM is equipped with some BIOS-like hard codes for attestation when it is manufactured. The dash line for OS downloading means that OS can be obtained via other channels such as CD-ROM.

DRM enforcement. When the DRM application is being executed, a specific loader such as `explorer.exe` in Windows OS package will send a key request to the TPM. TPM verifies the integrity of the OS and the loader, and then decrypts the secret key d to `explorer.exe`. With d, the loader decrypts the encrypted application **C** into memory so as to obtain the memory mapping of **P**. Then, **P** will be executed normally, e.g., read the content and enforce DRM protection.

4.2 Analysis

Correctness. When all the modules are intact, the DRM application can be loaded into process memory, hence the application can enforce the DRM on the protected content.

Security. In the installation stage, the OS provider will testify the authenticity of the TPM based on the manufacture information. As long as the TPM hardware is secure, the OS private key will be kept secret. Meanwhile, since the application provider encrypts the DRM application, and only the encrypted application is stored in the non-protected storage such as hard disk. In the enforcement stage, only the designed process can send the key request to decrypt the protected application. Hence, neither can the traitor tamper/reverse-engineer the DRM code, nor can he use a malicious debugger to load the code.

Additionally, to reduce the risk associated with d being disclosed, we can employ a secret-sharing scheme to split d into $d = d_1 \oplus d_2$. The share d_1 is embedded into the OS directly with some software tamper-resistance technology [19,20,21,22,23], and the other share d_2 is downloaded from the vendor during the *installation* stage. When the DRM application is launched, the OS will obtain d_2 from the TPM to reconstruct d. With this improvement, in the unlikely event of a hacker breaking the TPM and retrieving d_2, he is unable to recover d directly, thus limiting the damage.

5 Conclusion

Since a TPM-enabled platform can prevent a user from running an "unapproved" application, it is possible to develop a tamper-resistant TPM-enabled DRM application, as suggested in [4,15]. But it is not trivial to achieve that. We have presented two attacks on the basic TPM-enabled DRM: the abuse attack and the injection attack. The abuse attack misuses an "approved" loader (e.g., debugger or browser) to run the DRM application, while the injection attack tampers with the genuine DRM application and re-signs the tampered version so that it can run as an "approved" application on the platform. Both attacks enables a traitor to gain unauthorized access to protected DRM content.

In order to counter these attacks, the present OS-encapsulation scheme makes sure that an encrypted DRM application is only loaded and decrypted by an authenticated OS, thus defeating the abuse attack, as well as the injection attack.

References

1. Sandhu, R., Zhang, X.: Peer to Peer Access Control Architecture Using Trusted Computing Technology. In: SACMAT, pp. 147–158 (2005)
2. Schell, R., Thompson, M.: Platform Security: What is Lacking, Elsevier Science, Information Security Technical Report (January 2000), `http://www.dx.doi.org`10.1016/S1363-4127(00)87628-1
3. Trusted Computing Platform Allaince, Tcpa main specification v. 1.2, `https://www.trustedcomputinggroup.org/specs/TSS/`.
4. George, P.: Smart Cards: A Bridge Between Users And Trusted Platforms, e-Smart Conference (2004), see also `http://citeseer.ist.psu.edu/729848.html`
5. Haldar, V., Chandra, D., Franz, M.: Semantic Remote Attestation -A Virtual Machine directed approach to Trusted Computing. In: Virtual Machine Research and Technology Symposium, pp. 29–41 (2004)
6. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: 11th ACM Conference on Computer and Communications Security, pp. 132–145 (2004)

7. Smith, S.W., Safford, D.: Practical Server Privacy Using Secure Coprocessors. IBM Systems J. 40(3), 683–695 (2001)
8. Camenisch, J.: Better Privacy for Trusted Computing Platforms. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 73–88. Springer, Heidelberg (2004)
9. Wurster, G., van Oorschot, P.C., Somayaji, A.: A Generic Attack on Checksumming-based Software Tamper Resistance. In: IEEE Symposium on Security and Privacy, pp. 127–138 (2005)
10. van Oorschot, P.C., Somayaji, A., Wurster, G.: Hardware assisted circumvention of self-hashing software tamper resistance. IEEE Transactions on Dependable and Secure Computing 2(2), 82–92 (2005)
11. Chang, H., Atallah, M.: Protecting Software Code by Guards. In: Sander, T. (ed.) DRM 2001. LNCS, vol. 2320, Springer, Heidelberg (2002)
12. Horne, B., Matheson, L.R., Sheehan, C., Tarjan, R.E.: Dynamic Self-Checking Techniques for Improved Tamper Resistance. In: Sander, T. (ed.) DRM 2001. LNCS, vol. 2320, Springer, Heidelberg (2002)
13. Felten, E.W.: Understanding trusted computing: will its benefits outweigh its drawbacks? IEEE Security & Privacy 1(3), 60–62 (2003)
14. Anderson, R.: Trusted Computing Frequently Asked Questions, `http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html`
15. Arbaugh, B.: mproving the TCPA specification. IEEE Computer 35(8), 77–79 (2002)
16. Safford, D.: The Need for TCPA (October 2002), `http://www.research.ibm.com/gsal/tcpa/why_tcpa.pdf`
17. TCG Specification Architecture Overview, Specification Revision 1.2 28 (April 2004)
18. Architecture, O.D.: Approved Version 2.0: OMAAD-DRM-V2_0-20060303-A (2006), `http://www.openmobilealliance.org/release_program/drm_v2_0.html`
19. Cerven, P.: Crackproof Your Software. William Pollick publisher (2002)
20. Wee, H.: On Obfuscating Point Functions. In: STOC, pp. 523–532 (2005)
21. Wu, Y.: Guarding Software Checkpoint. International Journal Computer Scence and Network Security 5(12) (2005)
22. Chow, S., Eisen, P., Johnson, H., van Oorschot, P.C.: A White-Box DES Implementation for DRM Applications. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 1–15. Springer, Heidelberg (2003)
23. Jacob, M., Boneh, D., Felten, E.: Attacking an Obfuscated Cipher by Injecting Faults. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 16–31. Springer, Heidelberg (2003)

Online Tracing Scanning Worm with Sliding Window[⋆]

Yang Xiang and Qiang Li

College of Computer Science and Technology, JiLin University
ChangChun, JiLin 130012, China
sharang@yahoo.cn, li_qiang@jlu.edu.cn

Abstract. Breaking out of network worms brings a tremendous damage to the Internet. Launch the worm defense and response can improve anti-attack capability of networks. Tracing worm propagation process after its outbreak can reconstruct not only the earliest infected nodes but also the timing order of victims been infected. Based on the improvement of existing offline worm tracing algorithm, we can realize the near real-time tracing for the propagation process of scanning worm: Network traffic data are real-time collected by the detection points from different LANs, then separated into continuous-time detection sliding windows; in every time window, we repeatedly and randomly collect paths that contain worm scanning and infected flow rate, reconstruct path of worm propagation in the current detection window. Results accumulated in sequential detection sliding windows continues doing feedback amendment, real-time reflect the process of worm propagation. we establish a virtual experimental environment of worm propagation and tracing to evaluate the algorithm. Tracing network worm propagation from the initial attack can inhibit continuous spread of the worm, ensure that no more host is infected by the worm, and provide basis for the determination of worm attack origin.

Keywords: Worm, Propagation path, Online tracing, Detection window.

1 Introduction

Network worms' threat to the computer system security and network security is increasing. New intelligent worms and polymorphic worms continue to appear. Diversification of the propagation approach and complexity of the application environment enable network worms' outbreak to increase, latent ability to rise, coverage to extend, and the losses to enlarge[1,2,3]. Since 2001, CodeRed[4] and other network worms' successive outbreak bring a serious threat and destruction to the Internet. Network worms' outbreak allow attackers to control

[⋆] Supported by NSFC(60703023) and Seed Fund of JiLin University. Corresponding author: Qiang Li, li_qiang@jlu.edu.cn thousands of hosts in a short time, launch DDoS attacks, steal security information and destroy key data. Therefore worm outbreak has terrible influence every time[2,3,4,5,6].

Tracing worm's evolution (that is, tracebacking worm's attack path)[7,12,13] can reconstruct not only the patient zero (i.e., the initial victim), but also the infection node list in evolution process. Even if partial trails can be captured, it has significance in restraining evolution of worm, in investigating and collecting evidence. So far, however, even worm adopts a simple scanning strategy, acquiring its evolution process is still a very difficult task.

Existing tracing algorithms for worm propagation evolution are all method of analysis, that is, algorithm trace back and predict the results through a certain period of flow collection after the worm's outbreak. It is quite difficult to achieve real-time tracing (i.e., acquire evolution path nearly when the worm is propagating). Although these offline algorithms can finally obtain worm origin and propagation path, it can not capture propagation path when the worm is spreading, and can not indicate the worm evolution process along with the time changing.

Accordingly, it is necessary to research online worm tracing algorithm in complex network environment, for the near real-time tracing of network worm origin, inhibit continues spread of the worm, ensure that no more host is infected by the network worm.

The most basic and simple propagation mode is scanning worm. Worm propagates between hosts and in the network, when scanning and infected objectives produce communication flows. Online tracing large scale scanning worm is to obtain the timing sequence of hosts and network equipments that have been infected. The timing sequence of worm propagation can not be found and reported by host itself, only can be reconstructed by the analysis of captured network flows.

Based on the improvement of existing offline tracing algorithm, we propose a near real-time online algorithm to trace the evolution process of scanning worm. Network traffic data are real-time collected by the detection points from different LANs, then separated into continuous-time detection sliding window. In every time window, we repeatedly and randomly collect paths that contain worm scanning and infected flow rate, reconstruct path of worm propagation in the current detection window. Results accumulated in sequential detection sliding windows continue doing feedback amendment, and real-time reflect the evolution of worm propagation process. We establish a virtual experimental environment of worm propagation and tracing, and analyze performance of the algorithm. By using online random walk algorithm with sliding window and *causal tree* merge algorithm, worm evolution process can be dynamically reflected in specified time interval. Properly choosing parameters, according to the network traffic and worm outbreak characteristics, can effectively improve tracing accuracy.

This paper follows as: Part 2 introduce the related work of worm tracing, Part 3 shows improved random walk algorithm and online tracing algorithm, Part 4 is a analysis of tracing algorithm using sliding window; Part 5 is conclusion.

2 Related Work

Existing IP tracebacking technology is not applicable to tracing network worm propagation path. If worm attackers use forged addresses, IP tracebacking strategy[8,9] can be used to identify the true host addresses. But in order to improve the success rate of its attacks, worm usually uses real IP addresses to send attack packets during the propagation. In distributing attack, vast majority of packets do not have worm origin address, but only one of the many victims. Therefore, it needs to find infected victims in the earlier stage.

Stepping stones tracing technique is also unsuitable for tracing the worm. Now in the two ways of detecting stepping stones, content-based method needs to spend huge workload on packet content analysis, and can not detect polymorphic worm or worm which uses encryption technology. Method based on the session's correlation between sequential packets[10,11] is also inapplicable to network worm attack, because worm does not use interactive session.

Unfortunately, to date there have been few proposals for offline retracing the steps of a worm infection. Xie et al. [12] offered a randomized approach that traces the origin of a worm attack by performing a random walk over the hosts contact graph. The graph is generated by collecting flow rates between potential victims during the worm's propagation. Although this approach can provide a wealth of information about the worm's evolution, most notably, the who-infected-whom tree and patient zero (i.e., the initial victim), it requires traffic traces on a global scale to reconstruct the evolution of a large scale event. Rajab et al. [7] presented a simple technique that uses the history of scans acquired by a network telescope to infer the actual sequence of host infections. Intuitively, if probes from one infected host arrive at the monitor before scans from another infected host, we can infer that the former host was infected before the latter one. However, inherent randomness in the scanning process, size of the telescope relative to the number of vulnerable hosts, hybrid of worm propagation under different conditions and other factors will have a direct negative impact on the reliability of inferred results. When the interval of arriving packets reduces, it will be more difficult to accurately detect the actual infection sequence. A different approach was suggested more recently by Kumar et al. [13] where the Witty worm was reversely engineered to recover the random scanning algorithm and its corresponding initial seeds. Given knowledge of the target selection algorithm, the sequence of scans could be re-enacted to provide a detailed view of the worm's evolution, and also provide insights into characteristics of the infected hosts. However, although the information required for this approach (i.e., the payload) can be recovered locally, the mechanism can not be easily applied to other worms, since each instance will have to undergo the same, possibly arduous, task of reverse-engineering.

3 Online Tracing Algorithm

Online network worms tracing need to improve the real-time performance of the algorithm. There are two ways that can implement real-time task. First, reconstruction can be made when data are collecting, there is no need to collect all the data before calculations. Tracing algorithm is executed in continuous observation timing windows, gradually amends results and enhances real-time. Second, controlled parameters can dynamically adjust, learn and improve the performance of tracing algorithm based on different networks and worm outbreaks. The initial time of worm outbreak is the best time to trace propagation process. Through continuous data collection and the execution of tracing algorithm, the *causal tree* of worm propagation can be obtained, and then worm origin can be traced.

3.1 Assumptions and Definitions

The host communications in network is defined as a directed graph $G=\langle V, E\rangle$, called *host contact graph*. The nodes of G is $V = H \times T$, where H is all hosts in the network and T is time. The edges E is a subset of $V \times V$. So an edge in G can be expressed as $e = \langle u, v, t^s, t^e\rangle$, $u \in H$, $v \in H$, $t^s \in T$, $t^e \in T$, $\langle u, t^s\rangle \in V$, $\langle v, t^e\rangle \in V$. Figure 1 shows a partial example of the *host contact graph*.

Each direct edge in the *host contact graph* represents a network flow. An edge is defined as an *attack edge* if it carries attack traffic, whether or not it is successful in infecting the destination host. An attack edge is defined as a *causal edge* if it corresponds to a flow that successfully infected a normal host. Other edges are *normal edge*.

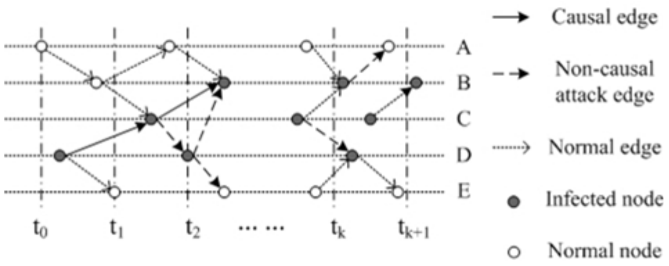


Fig. 1. Example of *host contact graph*

3.2 Offline Tracing Algorithm

Based on the offline random walk algorithm in reference[12], a *causal tree* (i.e., infection tree) can be reconstructed according to the identified *causal edges*. Algorithm's ultimate aim is to identify the *origin edges* of worm propagation.

Edges in the higher level of *causal tree* are more likely to be the *origin edges* of worm propagation.

Algorithm works as follows: Repeatedly walk W times, each walk marks a path in the *host contact graph.* For each path, we first randomly select an initial edge, then reverse-walk along the time. After that a possible *precursor* of the current edge is randomly selected in each step. Every walk begins with a random edge. Suppose current edge $e_1 = \langle u_1, v_1, t_1^s, t_1^e \rangle$, if edge $e_2 = \langle u_2, v_2, t_2^s, t_2^e \rangle$ satisfy $v_2 = u_1$ and $t_2^e < t_1^s < t_2^e + \Delta t$, then e_2 is a possible *precursor* of e_1. We randomly select an edge from e_1's precursors to continue the current walk. A walk is stopped when e_1 have no possible *precursor* or the step of current walk arrives at limitation d. In the process of execution, algorithm records the walk frequency for each selected edge. After W walks are completed, we select the top Z edges (*Top-Z*) which have the biggest walk frequency, and reconstruct a *causal tree.*

3.3 Advanced Tracing Algorithm

In many cases offline random walk algorithm can not obtain a tree, only a jungle or sometimes may be just an ordinary graph. In order to merge the *causal trees* from different phases in online tracing algorithm, we improve the basic moonwalk algorithm so that the *Top-Z* can reconstruct a *causal tree.* First we define the following control parameters:

Table 1. Parameter defined in online tracing algorithm

W	Number of walks in one tracing.
Z	Maximal edges returned in one tracing.
d	Maximal length of one walk path.
Δt	Maximal time difference between two adjacent edges in a walk.
λ	Decide which edge to choose when there is a conflict.
ω	Walks that ignore the path length $\leq \omega$.
ϕ	An edge is selected in *Top-Z* only if walk frequency $\geq \phi \times W$.

Suppose two edges $e_1 = \langle u_1, v_1, t_1^s, t_1^e \rangle$ and $e_2 = \langle u_2, v_2, t_2^s, t_2^e \rangle$ are selected into *Top-Z*. If u_1=v_2 and v_1=u_2 there will be a circle, if v_1=v_2 the *Top-Z* can not be reconstructed to a *causal tree.* Both cases e_1 and e_2 forms a pair of *conflict edges.* Suppose walk frequency of e_1 and e_2 is respectively c_1 and c_2. Define $\lambda_c = ABS(c_1 - c_2) \div MAX(c_1, c_2)$, ABS for absolute value and MAX for the maximum. Because e_1 and e_2 has been selected into *Top-Z*, they have a very large possibility to be *attack edges.* However there is only one *causal edge* in the two. λ_c describes degree of proximity between c_1 and c_2. We select early edge as *causal edge* when c_1 and c_2 is proximity enough (that is $\lambda_c \leq$ threshold λ), otherwise select the edge which has a larger walk frequency.

The advanced algorithm is running online, no matter whether there is a worm outbreak. So an execution of the tracing algorithm may return a *causal tree* even if there is no worm origin in the network. In order to reduce the occurrence of such situation, two control parameters ω and ϕ are defined.

Define L ($1\leq L \leq d$) as the path length of a walk, we do not update edges' walk frequency in the path if $L \leq \omega$, in other words do not consider it as a potential propagation chain if the path is very short. We consider that path of L=1 is not a propagation chain, so there is $\omega \geq 1$ obviously. ω influences algorithm accuracy, in the following sections we will further discuss the effect.

In a tracing, walk frequency will be obtained after W walks. An edge is rejected when its walk frequency $\leq \phi \times W$, even if it was selected into *Top-Z*.

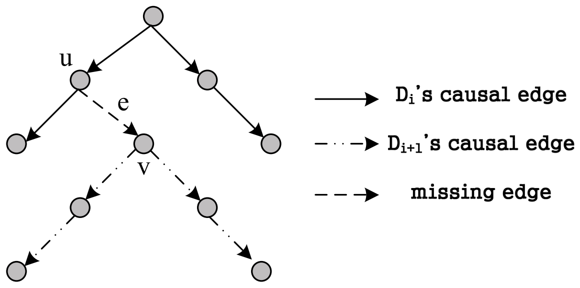


Fig. 2. Missing edge example

Some real *causal edges* may be missed when merging the two *causal trees* from adjacent slide windows. Figure 2 describes this situation. D_i and D_{i+1} denotes two windows in two sequential time intervals. Complete *causal tree* includes all edges in Figure 2. There is one *causal tree* respectively in D_i and D_{i+1}, but edge e is lose due to the randomness of algorithm, so a complete *causal tree* can not be obtained when merging the two trees.

Edge e's missing can not be rectified during D_i, because the algorithm can not predict that host u will infects host v in the future. So this kind of mistake can only be revised in D_{i+1}.

In the *causal tree* of D_i, H_i is defined as the hosts set, Z_i is defined as the edges set. In the *causal tree* of D_{i+1}, v is defined as the root of tree. For all $u \in H_i$, if edge e=$\langle u, v, k\rangle$ does not appear in Z_{i+1}, it will be added to set X_{i+1}. So every tracing will return two sets of Z_i, X_i. When Z_i and Z_{i+1} can not merge to a complete *causal tree*, we merge the two *causal trees* by selecting an edge which has a maximal walk frequency from X_{i+1}.

Advanced algorithm pseudo-code is as follows:

Algorithm 1. Advanced Tracing Algorithm

```
//  function return two sets : topZ, restX
//  W : number of walks
//  d : maximum length of a path
//  delta_t : maximal time difference between two adjacent edges
//  Z : maximal number of edges to be returned
//  omega : effective path length
//  phi : minimal walk frequency rate in Top-Z
Set<Flow>[2] new_random_walk ( Set<Flow> flowSet, Set<Host> H,
    int W, int d, int delta_t, int Z, int omega, int phi )
{
    // Record walk frequences for every chosen Flow
    Map<Flow, int> count;
    for ( int i = 1 to W ) {
        // Select a flow randomly from flowSet
        Flow F(1) = get_random_flow ( flowSet );
        int step = 1;
        for ( int j = 2 to d ) {
            // Get F(j-1)'s candidateSet from flowSet
            Set<Flow> candidateSet =
                get_candidate ( flowSet, F(j-1), delta_t );
            if ( candidateSet is empty ) break;
            ++step;
            // Select a flow randomly from candidateSet
            Flow F(j) = get_random_flow ( candidateSet );
        }
        if ( step <= omega ) continue;
        for ( int k = 1 to step ) {
            ++ count[ F(k) ];
        }
    }
    Set<Flow> topZ = get_topZ ( count, Z, phi );
    Set<Flow> restX = get_restX ( count, H );
    return { topZ, restX };
}
```

3.4 Online Tracing Algorithm with Sliding Window

In order to trace online, we partition time into sliding windows, and then use the current result to revise the previous one, thus real-time updating *causal tree*.

In Figure 3, we use a similar approach to sliding window, the size of window is $k \times \Delta t$ (i.e. $D_i = [t_{i-k}, t_i]$). We collect network traffic data every Δt (Δt is the same as maximal time difference between two adjacent edges in a walk), execute improved tracing algorithm in section 3.3 using the recent k traffic data.

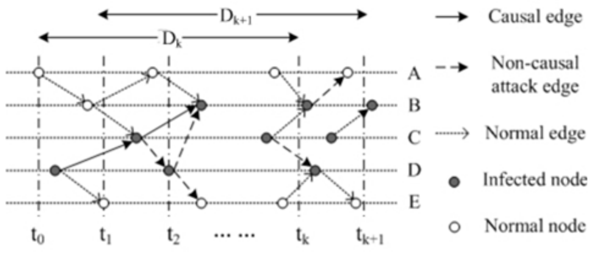


Fig. 3. Example of sliding window

Algorithm pseudo-code is as follows:

Algorithm 2. Online Tracing with Sliding Window

```
//  Tree is a data-structure implementing a tree
void online_walk ( int W, int d, int delta_t, int Z,
                   int X, int omega, int phi )
{
    Set<Flow> topZ = {};
    Set<Host> H = {};
    for (int i = 1; true; ++i)
    {
        // Get network flow of D(i)
        Set<Flow> flowSet(i) = get_flow ( i );
        // Run new_random_walk algorithm
        Set<Flow>[2] setArray = new_random_walk (
            flowSet(i), H, W, d, delta_t, Z, omega, phi );
        Set<Flow> topZ(i) = setArray[0];
        Set<Flow> restX = setArray[1];
        // Construct causal tree
        topZ = forest_to_tree( topZ, topZ(i), restX );
        display_tree( topZ );
        // Update Host set H
        H = get_host_set ( topZ );
    }
}
```

4 Experiments and Analysises

In the *host contact graph*, T_n is defined as the number of total *non-causal edge*, while T_m as the number of total *causal edge*, and T_f as the number of *causal edge* in the *top level* of *causal tree*.

In the edge set *Top-Z*, Z_n is defined as the number of *non-causal edge*, while Z_m as the number of *causal edge*, and Z_f as the number of *causal edge* in the *top level* of *causal tree*.

To evaluate the algorithm, we define three performance metrics:

Accuracy rate:

$$AC = \frac{Z_m}{Z}\% \tag{1}$$

False positive rate:

$$FP = \frac{Z_n + Z_m - Z_f}{T_n}\% \tag{2}$$

False negative rate:

$$FN = \frac{T_f - Z_f}{T_m}\% \tag{3}$$

When considering given parameter's affect to the algorithm performance, we only allow the corresponding parameter to change. Without special note, the initial values of the parameters in the experiment are as follows:

Table 2. Parameters' initial value in the tracing algorithm

D: Size of slide window (second)	4000
H: Number of virtual host	200
Z: Maximal number of edge in *Top-Z*	25
W: Number of walks in a tracing.	5000
Δt: Maximal time difference of two adjacent edges	1200
d: Maximal length of a path.	8
ω: Effective path length	1
λ: Threshold when edges conflict.	0.1
ϕ: Minimal walk frequency rate in *Top-Z*.	0.02

4.1 Experimental Environment

In the algorithm experiment, using UML virtual machine technology[14,15], we establish an experimental environment include 200 virtual nodes base on 7 PCs. Virtual clients running Redhat Linux 6.1 operation system with BIND security holes. Physical hosts running Redhat Linux 9.0 operating system. Several virtual clients in a physical host form a *virtual local network* (VN), virtual clients in different host communicate with each other using gateway in every physical host. This environment can be independently reused. Nodes and network topology can be flexibility configured. Experimenters enable to collect network data and infections for analysis after the outbreak of worm.

Manually launch a worm propagation break source in one of the four LANs, startup Lion worm attack[16], then running tracing algorithm to analyze the final result and true infections. The continuous real time collection network flows include not only worm flows, but also pre-installed normal background flows.

4.2 Parameter Selection

Performance vs. Δt Figure 4 shows the impact of Δt on the AC rate. When Δt increase, AC rate climb up but finally descend. AC rate is very low when Δt

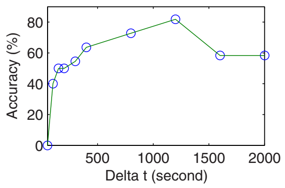


Fig. 4. AC rate vs. Δt

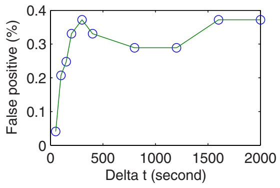


Fig. 5. FP vs. Δt

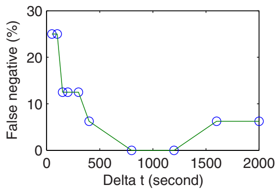


Fig. 6. FN vs. Δt

very small, because walk is more likely to arrive at a host that has no precursor in the time scope Δt, making walk can not go on, so walk is more likely to terminate when the path is very short. At the same time short walk has a lower possibility to arrive the top level of *causal tree*, most of the short walk stopped in the bottom of *causal tree*.

Larger Δt make the walk has more chance to reach the top level of *causal tree*, so AC rate will increase along with Δt. But AC is lower down when continue increase Δt. A wider Δt mean a greater opportunity to select a normal edge, too many *normal edges*' selected make AC rate decrease. We can also discover this from Fig. 5 (FP rate vs. Δt). The FP rate is increasing along with Δt, because of more *normal edges* are selected into *Top-Z*.

Furthermore when Δt is very big, walk have a larger chance to go to a normal host after arrived the worm origin, this is one of the reasons why AC rate decrease. Decrease of AC rate result in the increasing of the FP and FN, the following Fig. 5 and 6 are also proved this.

We also can be seen in Fig. 5 that FP rate is extremely low while Δt is very small. It is because walk path more likely to be a short path when Δt is small, and the short path in the algorithm is ignored, too short path contains more *normal edges*. Only few edges are selected into *Top-Z*, of course FP rate will be very low. Figure 6 also prove this, only a few edges selected into *Top-Z*, failing to report many *causal edges*, FN rate reach the highest level.

Combining Fig. 4, 5 and 6 we can see that there is a optimal Δt, makes AC rate reach the highest level while FP and FN are the lowest. However, the optimal value of Δt is related to the scanning rate of worm, in the experiment use Δt=1200 seconds.

Performance vs. *d*. The reference[12] has indicated that the difference of probability between selecting *attack edge* and *normal edge* is pro rata with maximal hop count d, and AC rate is climbing while d is increasing. Figure 7 proved this issue. However, Fig. 7 shows not entirely the same, when the AC rate reached highest level, continues increasing d will cause AC rate slowly declining (although not decline too much). At the same time FP and FN rate in Fig. 8 and 9 are raising. The reason is already mentioned before, too big d make the walk have a larger chance to go to a normal host after arrived the worm origin, so AC rate is decreasing and FP, FN rates are rising.

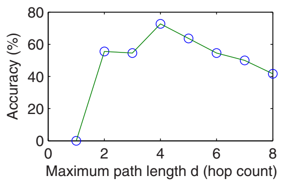


Fig. 7. AC vs. d

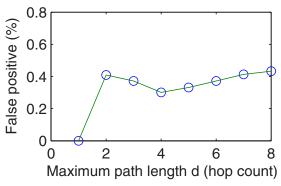


Fig. 8. FP vs. d

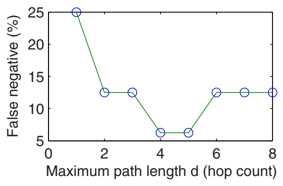


Fig. 9. FN vs. d

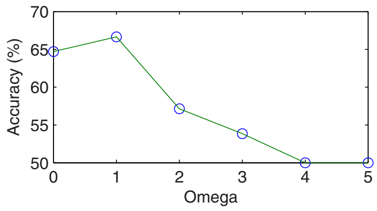


Fig. 10. AC rate vs. ω

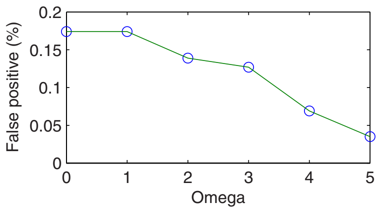


Fig. 11. FP vs. ω

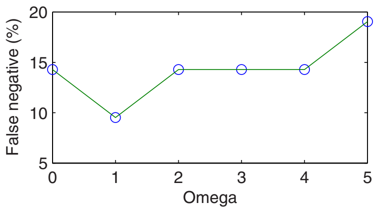


Fig. 12. FN vs. ω

Performance vs. ω. Too short walk path will be ignored, an appropriate value of ω help to achieve an acceptable FP rate and FN rate. From Fig. 10, 11 and 12 we can see that when effective path length (ω) is increasing, more short path is eliminated. Compare to long path, *normal edge* has a greater proportion in a short path. Therefore excluded of short paths makes FP rate lower, but FN rate is increasing and AC rate is falling down because excluded short path will also reducing opportunities for tracing some of the *causal edge*. Not all of the causal chains are very long, short causal chain may be directly excluded from the result set. So there is also a tradeoff when choosing value for ω.

Performance vs. λ. Algorithm's ultimate goal is to reconstruct the *causal tree*, and tracing the origin of worm propagation. Simple random walk algorithm only returns a forest or ordinary graph in most cases. When solving the conflict in reconstructing *causal tree*, we introduced a threshold λ.

P_m is defined as the probability that an *attack edge* is walked, while P_n as the probability that a *normal edge* is walked. We assume ΔP=P_m-P_n. The reference[12] has proved that ΔP >0, further indicate ΔP is pro rata with

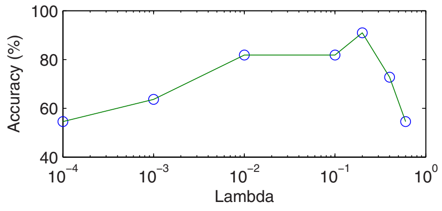


Fig. 13. AC rate vs. λ

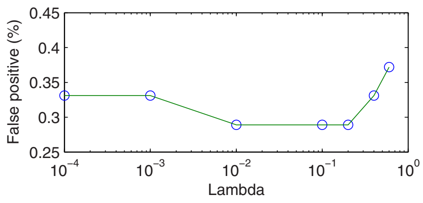


Fig. 14. False positive vs. λ

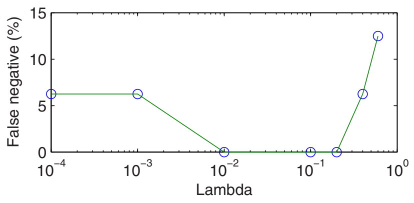


Fig. 15. False negative vs. λ

number of outgoing flows that a host initiates in unit time, and pro rate with valid host address rate of worm scanning. For scanning worms, due to its high scanning rate, infected host initiates a large outgoing flows, meanwhile some advanced scanning strategy will avoid scanning IP address that does not exist, this will lead to the increase of ΔP. As a conclusion, $\Delta P \gg 0$ for most scanning worms.

So when two edges' walk frequencies are fairly closed, they are more likely simultaneously to be two *attack edges* or two *normal edges*. If these two edges were selected into *Top-Z*, because of their high walk frequency, they have a higher probability to be two *attack edge*s. But *causal tree* can not simultaneously include a pair of *conflict edges*.

λ_c describes proximity degree of two edges. When $\lambda_c \leq \lambda$, the early edge is more likely to be causal edge. When $\lambda_c > \lambda$ the edges with a larger walk frequency is more likely to be *causal edge*. From Fig. 13, 14 and 15 can be seen that λ as a threshold that impact final results, too big or too small is not a good option, in general λ=0.1 is a better choice.

Performance vs. ϕ and Z. Parameter ϕ controls the number of edges in *Top-Z*. Less edge will be selected into *Top-Z* while ϕ is increasing, the reserved edges in *Top-Z* have a great walk frequency. Compare with the edges eliminate from *Top-Z* because of walk frequency less than $\phi \times W$, the reserved edges are more likely to be *causal edge*. Combined with Fig. 16 and 17 can be seen that, when ϕ is increasing, AC rate is climbing up and FP rate are lower down. But at the same time in Fig. 18 FN rate is climbing, due to the increased restriction of ϕ, more causal edges are lose.

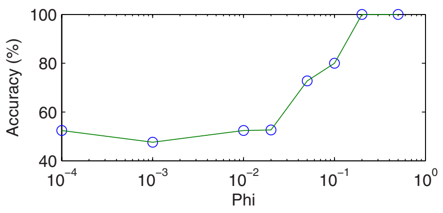
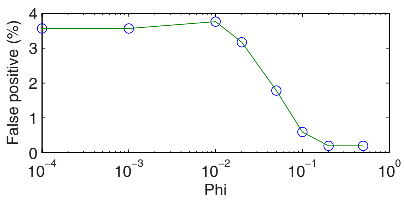


Fig. 16. Accurately rate vs. ϕ



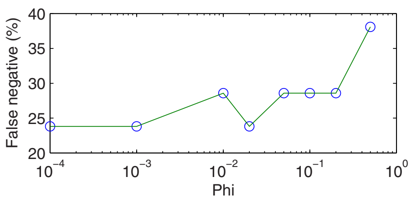


Fig. 17. False positive vs. ϕ

Fig. 18. False negative vs. ϕ

Offline random walk algorithm does not need to use ϕ control the size of the *Top-Z*, because we can change the value of parameter Z to achieve the same purpose.

However, in online algorithm, the worm outbreak time can not be predicted. When there is no worm in network, too big Z will results in selecting a large number of *normal edges*, but when the worm outbreaks, too small Z will cause the increasing of FN rate.

Introduce ϕ help to solve this problem, if the value of ϕ is appropriate. When there is no worm outbreak, walking path often very short, nearly no edge have a walk frequency larger than $\phi \times W$, so no edge was selected into *Top-Z*. When the worm outbreak, ϕ enable algorithm adapt to this change. Because more and more long paths are appeared, more *causal edges* will be selected into *Top-Z* because of its walk frequency exceeded $\phi \times W$.

Performance vs. *W*. For a random algorithm, a larger W means higher accuracy rate, but it also implies that the execution of algorithm use more time. Online algorithm has a high demand in real-time, need to receive rapid update. So there is tradeoff between accuracy rate and execution time. In the experiments we found that, when W and the number of flows in a slide window have the same magnitude, can achieve a better AC rate and an acceptable execution time.

Figure 19 and 20 show the relationship between W and accuracy or execution time when Δt=800 seconds, window size D=4000 seconds, number of flows F=4000 in a time window. From that can be seen when W is 5000, accuracy rate arrives 63.636% and algorithm execution time is 9.748 seconds.

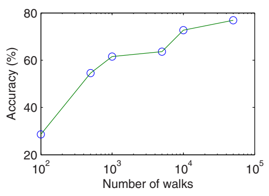


Fig. 19. Accurately rate vs. W

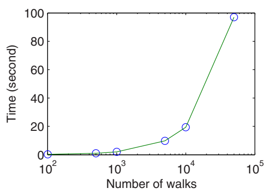


Fig. 20. Execution time vs. W

5 Conclusions

Online tracing the evolution of a worm outbreak reconstructs not only patient zero (i.e., the initial victim), but also the infection node list in evolution process. Even if proportion trails can be captured, it has significance in restraining evolution of worm in investigating and collecting evidence. Through experiments and analysis can be seen that, online tracing algorithm with sliding window can reflects worm evolution process. According to the network environment and worm characteristics, we dynamically change the running parameters in algorithm, advance the tracing accuracy. This paper's online tracing algorithm focuses on the basic scanning worms, and future work will study tracing more species of worm under complex network environment.

References

1. Wen, W.P., Qing, S.H., Jiang, J.C., Wang, Y.: Research and development of Internet worms. Journal of Software 15(8), 1208–1219 (2004)
2. Hernacki, B.: Emerging threats. In: Proceedings of the 2005 ACM workshop on Rapid malcode, Fairfax, VA, USA, November 2005. ACM Press, New York (2005)
3. Kienzle, D.M., Elder, M.C.: Recent worms: a survey and trends. In: WORM 2003: Proceedings of the 2003 ACM workshop on Rapid Malcode, pp. 1–10. ACM Press, New York (2003)
4. CERT. Code Red II: Another worm exploiting buffer overflow in IIS indexing service DLL (2001), `http://www.cert.org/incident_notes/in-2001-09.html`
5. Staniford, S., Paxson, V., Weaver, N.: Hwo to own the internet in your spare time. In: USENIX security Symposium, 11th. USENIX (2002)
6. Chen, Z.S., Gao, L.X., Kwiat, K.: Modeling the Spread of Active Worms. In: Proceedings of IEEE INFOCOM, San Francisco, CA (March 2003)
7. Abu Rajab, M., Monrose, F., Terzis, A.: Worm evolution tracking via timing analysis. In: Proceedings of the 2005 ACM Workshop on Rapid Malcode WORM 2005, Fairfax, VA, USA, November 11-11, 2005, pp. 52–59. ACM Press, New York (2005)
8. Savage, S., Wetherall, D., Karlin, A., Anderson, T.: Practical Network Support for IP Traceback. ACM/IEEE Transactions on Networking 9(3), 226–237 (2001)
9. Yaar, A., Perrig, A., Song, D.: FIT: Fast Internet Traceback. IEEE Infocom (2005)
10. Zhang, Y., Paxson, V.: Detecting Stepping Stones. In: Proc. of 9th USENIX Security Symposium (2001)

11. Peng, P., Ning, P., Reeves, D.S., Wang, X.: Active Timing-Based Correlation of Perturbed Traffic Flows with Chaff Packets. In: ICDCS Workshops 2005, pp. 107–113 (2005)
12. Xie, Y., Sckar, V., Maltz, D.A., Reiter, M.K., Zhang, H.: Worm Origin Identification Using Random Moonwalks. In: Proceedings of IEEE Symposium on Security and Privacy, May 2005, pp. 242–256 (2005)
13. Kumar, A., Paxson, V., Weaver, N.: Exploiting Underlying Structure for Detailed Reconstruction of an Internet Scale Event. In: Proc. ACM IMC (October 2005)
14. Dike, J.: User Mode Linux, `http://user-mode-linux.sourceforge.net`
15. Jiang, X., Xu, D., Wang, H.J., Spafford, E.H.: Virtual Playgrounds for Worm Behavior Investigation. In: Valdes, A., Zamboni, D. (eds.) RAID 2005. LNCS, vol. 3858, pp. 1–21. Springer, Heidelberg (2006)
16. Linux Lion Worms (2001), `http://www.whitehats.com/library/worms/lion/`

A New Proactive Defense Model Based on Intrusion Deception and Traceback

Junfeng Tian and Ning Li

Institute of Computer Network Technology, Hebei University, Baoding 071002, China
donnyli1229@yahoo.com.cn

Abstract. Along with the fast development of the Internet, the traditional passive defense measures have shortcomings and can not deal with the increasingly serious network security problems better. In this paper, a proactive network defense scheme is presented. And a new model of DTPM (Intrusion Deception and Traceback-based Proactive Defense Model) is established, which protects the precious network resources with the cooperation of intrusion deception and traceback. In the traceback module of DTPM, an improved approach APPM based on the PPM (Probabilistic Packet Marking) is developed, which makes up for the deficiency of the PPM in real-time capability and flexibility. By way of analyzing and comparing with other methods, this approach can decrease the overloads of many aspects and make traceback more efficient. The simulation experiment indicates the high performance and efficiency of this scheme.

Keywords: Intrusion deception; Trap environment; Honeypot; Traceback.

1 Introduction

Along with its fast development, the negative effect of the Internet is the serious network security problem. According to the Computer Security Institute(CSI), the result indicates that, the threats from computer crime including information stealingfinance deceivingabuse of internal usersvirus breaches continues unabated. Only the traditional passive defense measures such as firewall and IDS can not solve the network security problem better [1].What we need is to change from passivity to initiative and set up a dynamic proactive defense system to protect network resources efficiently.

As to the proactive defense, a representative technique is Honeypot. It is also called the network security trap which disguises as the real system with vulnerabilities, imitates vulnerable hosts, and attracts the hackers to enter and records their activities. By analyzing the data recorded by honeypot, we can understand the hacker's next action and the new attack methods they use [2]. Some product models of honeypot system appeared such as: the Deception Toolkit, CyberCop Sting, Spector, LaBrea Tarpit, Honeyd and ManTrap [3].

In the field of traceback, how to solve the problem of traceback and locate the attack source is becoming more and more desired. We can cut off the attack at the source using traceback and overawe the attackers to give up further attacks. In this way, we could launch the reverse attack toward the attacker, so that we can take the active role in attacking and defending. Some achievements have been made in traceback such as ICMP messages locating approach [4], Probabilistic Packet Marking (PPM) approach [5].

Currently, the researches on intrusion deception technique (Honeypot) and traceback technique are still independent. How to combine the intrusion deception and traceback together organically and to make them cooperate with each other better is the focus of this paper. Aiming that, a new scheme DTPM (Intrusion Deception and Traceback-based Proactive Defense Model) is presented. Based our model, when facing attack, we can respond to attack positively rather than passive and negative defense.

This paper is organized as follows: Section 2 introduces our new approach and DTPM model design in detail. In section 3, we analyze our new approach and give the simulation results. Finally, section 4 concludes the paper.

2 DTPM Model Design

2.1 DTPM Overview

Based on the thought of proactive defense and attack, we design the DTPM model utilizing the proactive entrapment and adopting the strategy of deceptionmonitorprotection and counterattack. Fig.1 shows the theory frame structure diagram of DTPM.

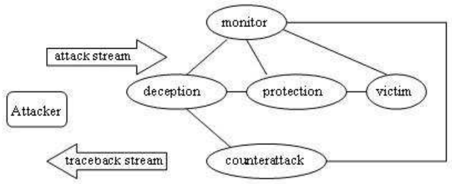


Fig. 1. DTPM theory frame structure diagram

DTPM model consists of Intrusion deception moduleTraceback module and Management and control module. Functions of each module are listed as follows: 1) Intrusion deception module. Implement deception to invader and guide the invader into the trap environment proactively. 2) Traceback module. Trace the attack back and locate the source of the attack. 3) Management and control module. It is responsible for controlling and cooperating with each module.

Even if the attacker intrudes into the real system that we protect, we can also transfer him from the real system to the trap environment via attack redirecting[6]. Meanwhile, we trap the attacker in the trap environment in order to strive for enough time to trace back and locate the attack source.

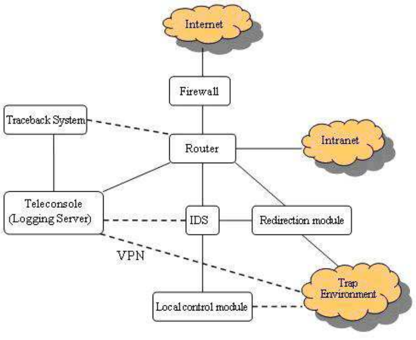


Fig. 2. DTPM system architecture graph

Above all, we can get the system architecture graph shown as Fig.2.

In DTPM, there are four types of data stream: network attack data stream management and control data streamcaptured data stream and traceback data stream. The control and flow of these four types data stream is shown as Fig.3:

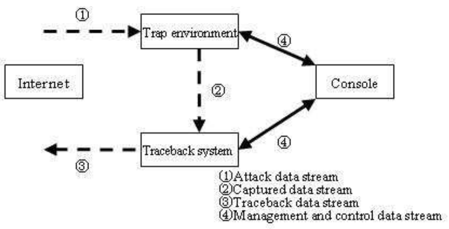


Fig. 3. Data stream in DTPM

2.2 Design of the Intrusion Deception Module

Spitzner thinks: honeypot is an information system resource whose value lies in unauthorized or illicit use[7]. Because honeypot does not expect real valuable services, any traffic to or from it is the most likely unauthorized and suspected activity. We make use of honeypots as the trap hosts, which imitate the real services of various internal servers and forge the IP address of the real system. The intrusion deception module is made up of firewallIDS and trap environment. There are more than one honeypot in the trap environment, each honeypot takes terms of the virtual honeypot(Honeyd). The whole trap environment adopts the architecture like Honeynets Gen1[8].

Firewall and IDS play the part of guiding the external invader into the trap environment and logging. Every action of the hacker is under controlled by interaction of firewall and trap environment. Firewall maintains a dynamic blacklist during the process. IDS renews the blacklist in time. For every connection from outside, if its source IP is in the blacklist, then it is led into trap environment; if it is a new IP, IDS judges whether the action from this source IP is an intruding behavior or not at first. If it is intrusion, IDS issues an order to the firewall. The firewall can stop the connection from this source IP or lead it to the trap environment, and adds its record to the blacklist. When it logs in again, it is led to the trap environment directly.

Placing router between firewall and trap environment for concealing firewall, it makes the attacker in the trap environment feel the network more real. The ACL (Access Control List) can be established on the router to filter the packet whose source IP address is not the address of the honeypots in the trap environment. It prevents the hacker from attacking other hosts by forging IP address after compromising the honeypot system.

Each honeypot host communicates with each other to work cooperatively. Local control centre manages the whole trap environment, and changes the configuration information in time. The teleconsole records the data information of the hacker under monitoring and the system log, cooperates with IDS to detect attack and triggers the traceback module when there is invader in the trap environment. All of the data streams take the terms of encrypting mode between each module. And the data transferring between them need to be authenticated. Fig.4 depicts the principle diagram of Intrusion deception module.

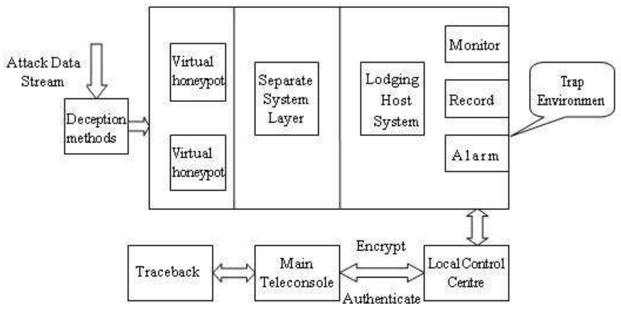


Fig. 4. Principle diagram of Intrusion deception module

2.3 Design of the Traceback Module

Current Internet infrastructure is exposed to many serious threats that can affect the availability of network services. Most attackers usually use incorrect or spoofed IP address in the attack packets to conceal their real location. When IP protocol was built up originally, there was limitation. The architecture of the Internet does not provide intrinsic support for identifying the real sources of IP packets. So it is very difficult to trace back and to locate the source. Therefore, the traceback system plays the vital role in the proactive security defense system.

Savage etc proposed PPM(Probabilistic Packet Marking)approach [5]. The core is to let routers add part of path information to the forward packets with certain probability. This information reaches the destination host with IP packets. After receiving large numbers of packets, the destination host can reconstruct the entire path with the partial path information included in the received IP packets, and then determines the attack source. This approach helps the victim distinguish the path of the attack traffic without the cooperation of ISPs(Internet Service Provider). However, the victim only starts traceback and receives large numbers of packets passively to compute the attacking path. The speed and effect of tracback are impacted by the uniform determined marking probability. Also the routers and networks don't cooperate with each other. Routers mark all the forward packets without selectivity, which increases themselves overloads.

Based on the PPM approach, we propose the APPM(Advanced Probabilistic Packet Marking) approach, which enables the attack source locating more exactfast and efficient. There are two key parts in APPM: the packets marking module and path reconstructing module. Two essential methods are compressed node sample marking method and compressed edge sample marking method[5], both of which operate on IP packet header. We adopt the combination of the two methods.

This paper makes use of the routing area dividing to partition the routers on the attack path into several AA (Autonomous Area) [9]. With the cooperation of the traceback system of DTPM and the backbone routers CR (Central Router) of each AA, it is to mark the packets only within the needed time and direction, which can decrease the overload of the midway routers greatly.

We make the following definitions for conveniently.

[Def.0] R1: denotes the compressed node sample marking method; R2: denotes the compressed edge sample marking method.C1,C2 denotes the path reconstructing method of R1,R2 respectively.
[Def.1] Ti: stands for marking the packets at Ti time. This time is computed by the traceback system and transfered to each marking router in terms of timestamp.
[Def.2] S: a kind of control switch. On the choice of marking method, S taking value 0 means choosing R1, and S taking value 1 means choosing R2.
[Def.3] CU: denotes the communicating information among traceback system-backbone router CR and marking routers. $CU =< S, P, Ti >$,P represents the marking probability.
[Def.4] D: stands for direction symbol. It means the direction of attack traffic which is contrary to the traceback request direction. D taking value 0 means following the downstream direction, and taking value 1 means following the upstream direction.

The traceback system consists of traceback server(TS)data analyzing server (AS) and path reconstructing server(CS). The functions of each component are described as follows.

(1)TS establishes communication with the backbone routers CR of respective upstream AA by means of IP tunneling technique. Among each CR and between CR and other marking routers of its own AA, the communication is also established by means of IP tunneling technique. The time stamp Ti used for communicating is added to the header's Optional field of IP packets. In Ti period the system performs traceback strategy and carries out marking. According to the traceback request received by CR, the attack traffic's direction D can be figured out. The CR of each AA on the attack path sends the received traceback information CU to other marking routers of its own AA through IP tunnel. CR does not participate in packets marking. Then the marking routers begin marking the forwarded packets within Ti period according to CU and D. When the time of Ti ends, the marking routers will finish the marking process.

(2)AS analyzes the data stream captured by trap environment and figures out which marking method to choose R1 or R2. According to the theoretical number of the marking packets needed to reconstruct the path, it computes the demanded P of this traceback process.

(3)CS carries on the following path reconstructing work using the received marked packets after the marking process is complete. CS chooses the different path reconstruction method C1 or C2 to locate the source.

After reconstructing the whole attack path, the source information is sent to TS. The TS submits this information to teleconsole which carries out log audit and forms the rule. The rule is submitted to firewall and IDS of the intrusion deception module in order to identify and defend the attack originated from this source. Towards the attack information, the teleconsole devises this kind of attack should be stopped or deceived. So the attack from this source would be resisted and defended efficiently.

The APPM approach can be described with flow chart as Fig.5 shows.

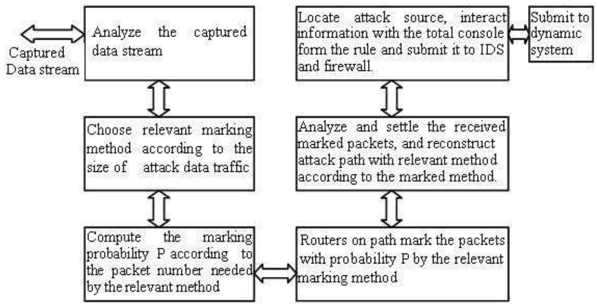

Captured Data stream
Analyze the captured data stream
Choose relevant marking method according to the size of attack data traffic
Compute the marking probability P according to the packet number needed by the relevant method
Routers on path mark the packets with probability P by the relevant marking method
Analyze and settle the received marked packets, and reconstruct attack path with relevant method according to the marked method.
Locate attack source, interact information with the total console form the rule and submit it to IDS and firewall.
Submit to dynamic system

Fig. 5. APPM approach flow chart

3 Performance Evaluation

3.1 Analysis

As the core of DTPM, the performance of APPM approach greatly influences the effect of whole DTPM. As the choice of marking field, the Identification field is used for fragmentation in IP header. The compressed partial path information could be encoded to this field by means of Hash value. There are three bits in the fragmentation flags field where there is 1 bit untapped. We could select that bit as the marking strategy choice switch S. The distance of almost all the Internet paths does not exceed 30 hops [10]. We set the first 5 bits of the 16-bit Identification field as the distance field.

In the existing approaches such as PPM[5]AMS[11], in order to avoid extra overload of the marking routers, the probability that the routers on the path mark forwarded packets is fixed, which generally is settled to 1/25. When a packet is marked by one router on the path, it may also be marked by the downstream routers, and the former marking information will be overloaded. We assume the distance of one marking router to the victim as d hops, and the probability that one packet is marked by one router and left unmolested by the downstream routers is $(1-p)^{d-1}$ .For the packets reached the victim, the probability that it is marked by this router and isn't marked by the downstream routers, is $p(1-p)^{d-1}$. Distance d is larger, this probability is smaller, i.e. the victim need receive quite more packets in order to receive the marked packets from the farthest router away from the victim. So the victim needs large numbers of packets to reconstruct the attack path. In R1, the inequation $Np(1-p)^{d-1} \geq 1$ should be satisfied, for instance, if p=0.5 and d=15, the victim must receive at least 32768 packets on average to reconstruct the path, among which there is only single sample packet marked by the furthest router. In R2, the number of attack packets needed to reconstruct the path decreases obviously, and the performance is optimized. The expected number of packets used for traceback is $(ln(d)+O(1))/p(1-p)^{d-1}$, if p=0.1 and d=16, the victim only need receive at 134 packets on average to reconstruct the path.

According to the number of packets captured by DTPM system, it selects relevant method and figures out one approximate marking probability so as to implements dynamic marking based on every different attack.

3.2 Simulation

In order to evaluate the performance of defense ability of our system, we carried out the simulation. We deployed the SUN's Solaris ftp server and the Fang Zheng's Yuanming web server on the local network in the lab, both of the operation systems took the default installation. Several deception hosts running the virtual honeypot software-Honeyd were deployed on the network, which imitate several virtual honeypot systems in one host. All of the above hosts compose the trap environment.The honeypots imitate the corresponding services of real servers respectively by the way of reflecting network ports and allocating ports and IP dynamically. The well-known Snort is selected as the IDS of the deception module. We configure the rules of firewall to control the data stream flowing in and out of it, and make sure the compromised honeypots can not be used to attack other external systems.

Boson NetSim software runs in one host to imitate Cisco routers. By configuring this software, the imitating routers achieve the marking function. The Tracerout software records the routing path passing through different roads from attacker to victim. One host running the CGI procedure established by Java plays the role of the teleconsole and logging server, which receives the data and logging records from the trap environment. The communication of each host adopts the MD5 to encrypt. One host as the traceback system launches the traceback and locates the attack source. The test environment is deployed in two different local area networks. They are 10.187.82.0∼254 and 10.187.84.0∼254, where we deploy the attack and defense parts respectively.

We use Honeyd as Honeypot system and Linux Red Hat 9.0 as OS. One configuration for the virtual honeypot of our system is as follows:

```
create windows
set windows personality "Microsoft Windows XP SP2"
set windows default tcp action reset
set windows default udp action reset
add windows tcp port 445 open
add windows tcp port 139 open
add windows tcp port 80 open
add windows tcp port 21 open
bind 10.187.82.200 windows
bind 10.187.82.170 windows
bind 10.187.82.177 windows
```

This instance creates a windows system template and imitates the Windows XP SP2 operation system. It binds 3 IP to the VMware and opens the above ports so as to deceive the attacker to intrude in. The other virtual honeypots are also configured following this instance.

The experiment used one host with real IP 10.187.84.122 as the attack part forging the IP source as 202.219.168.1. It started attack to the deployed defense system through the buffer overflowing vulnerability using TCP 139 and 445 ports, and it intruded into one honeypot of the trap environment successfully.This honeypot discovered the attack, recorded the logging of this attack, gave the alarm, and in the meantime sent attack logging records to the teleconsole with IP address 10.187.82.172. The teleconsole started the traceback system which began the marking function of imitating routers by means of interacting with the host running Boson software. Meanwhile, the whole attack path was reconstructed and the attack source was located according to the marked IP packets. Through the monitor of Tracerout, we find the location of our system is precise. And the whole process lasts 3 minutes and 36 seconds, which is in the reasonable scope.

We used the NS [12] to test the availability of the APPM. We establish the NS2 network topology by the Internet Mapping traceroute database obtained from Lucent Bell Lab [13]. We randomly selected 500 destination nodes as the attack nodes topology prototype. In the test we used the single source of the traceroute as the victim and the whole traceroute database as the upstream routers from the victim. In NS2, we configured the imitating routers to mark the packets and simulated the victim to carry on reconstructing the attack path. We launched 20 DoS attacks to the defense system including TCP SYN Flood and IP Spoofing attacks which went through different numbers of intermediate nodes. We take the average values of 500 independent results as the experiment result and figure out the average number of packets needed to reconstruct the attack path. The result is depicted as Fig.6.

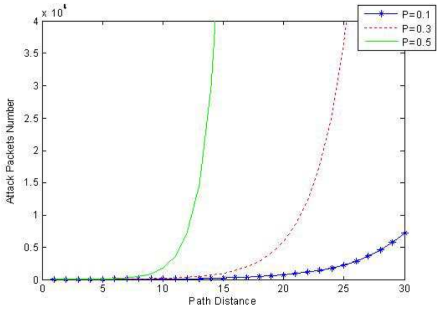


Fig. 6. The relationship of the path distance and the number of marked packets needed

As shown in Fig.6, when the data traffic per unit time is under 1000 and the path distance within 30, it can reconstruct the attack path only with low marking probability and fewer marked packets. When the path distance is within 25 and the attack data traffic per unit time is high, it needs more marked packets and high marking probability to reconstruct attack path.

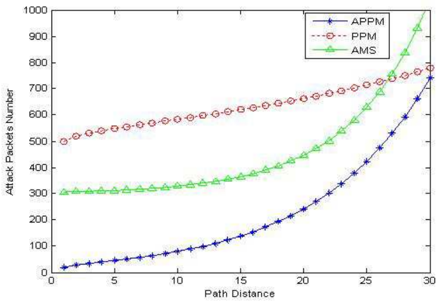


Fig. 7. Comparison with PPM and AMS approaches

In the simulation condition, the comparison of our approach with PPM and AMS approaches is shown as Fig.7. In the condition that marking probability is same such as p=0.1, the proposed APPM approach needs much fewer marked packets to reconstruct attack path than PPM and AMS.

The simulation result indicates that our approach needs fewer packets to reconstruct the attack path. It has good flexibility of selecting and adjusting traceback strategy dynamically according to the data traffic. The simulation experiment proves that our DTPM system can perform proactive defense efficiently.

4 Conclusion

This paper proposes the DTPM model which provides a viable solution for the network security problem. It combines the intrusion deception and traceback techniques together to construct the proactive network security defense system. The simulation result indicates the model has high quality of real-time and cooperation. It can resist attack effectively and make the real systems under protection. Locating the source and finding out the real attacker also can play a well role in early warning.

References

1. Shu-fan, Y., Fang-min, L., Jian-qiu, J., et al.: Constitute the passive infrastructure in network [J]. Journal of China Institute of Communications 24(7), 170–175 (2003)
2. Gubbels, K.: Hands in the Honeypot (2002–03), `http://www.sans.Org/rr/-white-papers/detection/365.php`
3. Honeypots Solutions: So you want to build your own honeypot, `http://www.tracking-hacker.com/solutions/`
4. Bellovin, S.M.: ICMP traceback messages, Internet draft (February 2003), `ftp://ftp.ietf.org/internet-drafts/draft-ietf-itrace-04.txt`
5. Savage, S., Wetherall, D., Karlin, A., et al.: Network support for IP traceback[J]. ACM/IEEE Transactions on Networking 9(3), 226–237 (2001)
6. Chun-he, X., Xiao-jian, L., Xin-ping, Z.: Research on Intrusion-Deception-Based Dynamic Network Defense[J]. Chinese Journal of Computers 27(12), 1585–1592 (2004)
7. Spitzner, L.: Honeypot: Definitions and Value of Honeypots (2005-5), `http://www.Tracking-hackers.com/papers/honeypot.html`
8. Honeynet Project: Know Your Enemy: Honeynets, `http://www.honeynet.org`
9. Qiang, L., Hong-zi, Z., Meng, Z., et al.: CoMM: Real-Time IP Traceback Model Based on Cooperative Marking and Mitigation[J]. Mini-Micro Systems 27(5), 769–773 (2006)
10. Theilmann, W., Rothermel, K.: Dynamic distance maps of the Internet[C]. In: Proceedings of the 2000 IEEE INFOCOM Conference, March 2000, pp. 275–284 (2000)
11. Song, D.X., Perrig, A.: Advanced and Authenticated Marking Schemes for IP Traceback[C]. In: Proceedings of the IEEE INFOCOM, pp. 878–886 (2001)
12. The Network Simulator-ns-2.[EB/OL] (2003), `http://www.isi.edu/nsnam/ns`
13. Internet mapping[EB/OL] (1999), `http://cm.bell-labs.com/who/ches/map/dbs/index.html`

On Modeling Post Decryption Error Processes in UMTS Air Interface

Fouz Sattar and Muid Mufti

Department of Electrical Engineering,
University of Engineering and Technology, Taxila, Pakistan
fouz@ieee.org, muid@uettaxila.edu.pk

Abstract. This paper analyzes the impact of encryption over the UMTS air interface. Using a finite state Markov characterization of the UMTS decryption process, a stochastic model has been developed that quantifies the impact of bit errors in the ciphertext and cipher synchronization counter. The effects of residual bit errors, UMTS air interface power budget, interleaving span and channel coding rate on the decryption process are analyzed.

Keywords: 3G Security, Wireless Confidentiality, Markov Modeling, Error probability, UMTS Encryption.

1 Introduction

Communications channels are prone to errors due to various physical impairments. Although application of error correcting codes overcomes or reduces the impact of these errors, residual errors can pass through undetected in some cases. These residual errors can in turn have significant impact on the transmitted data if it is block encrypted prior to transmission. Characterization of the effect of encrypting data before transmission over error prone channel and quantifying the impact of residual errors on decryption process is one of the key technical problems.

In [1], the authors have shown through empirical simulations that the use of data encryption over an error prone channel significantly increases the post decryption bit error rate (BER) at the receiver.

In [2], the author has developed stochastic models to describe the error structures of secret key ciphers. By deriving the first order statistics of these models such as the mean lengths of the error events and mean lengths of time between error events, the author has shown that the end-to-end confidentiality can increase the average post decryption BER by more than an order of magnitude.

With the proliferation of high speed wireless networking and on going deployments of third generation mobile communication systems, the demand for efficient, robust and secure encryption modes is ever increasing. One of the popular third generation mobile communication systems is Universal Mobile Telecommunication System (UMTS) which is a aimed at providing global mobility and wide range of services like broadband data, video streaming, multimedia applications etc. The confidentiality scheme adopted in UMTS is based on $f8$ algorithm [3]. While most research efforts on $f8$ focus on investigating its security properties and efficient implementations, the characterization of effect of residual errors on this algorithm and analysis of its performance in error prone channels still remains an open research subject. The results presented in this paper are a contribution towards addressing this problem.

The paper has following organization: Section 2 briefly describes the UMTS system architecture, the radio interface protocol structure and the underlying radio access encryption algorithms. In Section 3, an analytical model of the UMTS air interface is discussed. This model quantifies the residual bit error process on the physical channel taking into account various air interface parameters such as modulation, channel coding and interleaving. The impact of residual bit errors is then subsequently analyzed using a stochastic model of $f8$ decryption process based on a finite state Markov chain. Section 4 discusses the analytical results and finally in section 5, conclusions are given.

2 UMTS System Overview

2.1 UMTS System Architecture

A typical UMTS system architecture consists of user equipment (UE), UMTS terrestrial radio access network (UTRAN) and a core network. The interfaces between UTRAN and the core network and UTRAN and the UE are referred to as Iu and Uu interfaces respectively.

The Uu interface is layered into three protocol layers: the physical layer, the data link layer and the network layer. The physical layer is based on Wideband Code Division Multiple Access (WCDMA) radio technology. The data link layer is divided into two main sub-layers, namely Medium Access Control (MAC) [5] and Radio Link Control (RLC) [6]. RLC operates in three different modes: Transparent Mode (TM), Unacknowledged Mode (UM), and Acknowledged Mode (AM). UMTS defines two types of dedicated physical channels: dedicated physical data channel (DPDCH) and dedicated physical control channel (DPCCH). Information is transmitted over the physical interface in 10ms duration radio frames. Each frame is divided into 16 slots of length 0.625 ms. To mitigate the effect of burst errors, a two-stage (inter-frame and intra-frame) interleaving is applied on the scale of a transmission time interval (TTI), which is equal to 1,2,4 or 8 radio frames (10-80ms).

2.2 UMTS Radio Access Confidetiality

UMTS radio access confidentiality is based on $f8$ encryption algorithm. $f8$ is built around KASUMI block cipher [4] using combination of OFB and CTR modes and pre-whitening of feedback data. Figure 1 depicts the ciphering algorithm. The $f8$ algorithm takes a 128-bit key CK, a 32-bit counter $COUNT$, a 5-bit radio identifier $BEARER$, a 1-bit direction identifier $DIRECTION$,

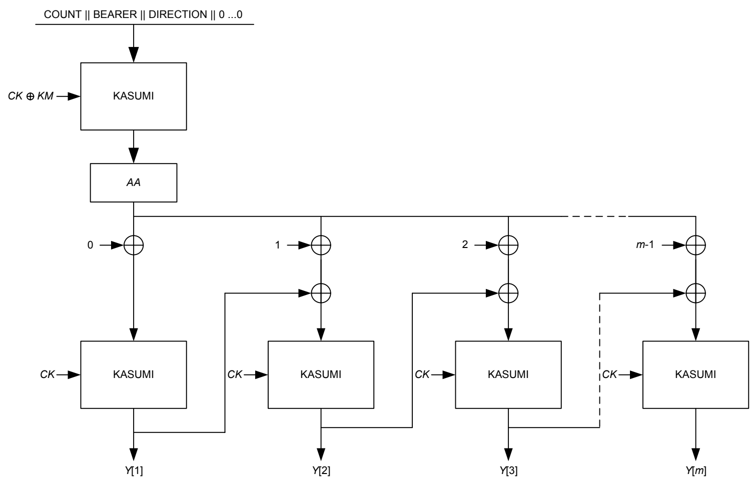


Fig. 1. $f8$ Ciphering Algorithm

and a message $M\{0,1\}$ to generate a keystream KS (the leftmost $|M|$ bits of $Y[1]||...||Y[m]$) which is exclusive-ORed with M to return ciphertext C of same length as M. Decryption operation is identical to the encryption.

In practical implementations, ciphering is performed either in RLC sub-layer or in the Medium Access Control (MAC) sub-layer.

3 Error Model of Decryption Process

The channel seen by decryptor is the physical channel as modified by the error correcting mechanisms used at the physical level. In most of the cases, the error correcting mechanisms provide less than perfect protection and some amount of residual errors pass through undetected resulting in a residual bit error probability P_{res}.Our goal is to determine the impact of residual bit errors seen on the physical channel on the decryption process. To this end, we first discuss a concise mathematical model of the UMTS Uu interface that quantifies the residual bit error process on the physical channel.

3.1 UMTS Uu Interface Residual Bit Error Model

In [10], the authors have developed an analytical expression to model P_{res} on the DPDCH uplink as:

$$P_{res} \approx \frac{1}{2} \sum_{d=d_{free}}^{\infty} c_d \; erfc\left(\sqrt{d \;R_c \left(\frac{E_b}{N_o}\right)_e}\;\right) \tag{1}$$

where R_c is the coding rate of the convolution code and d_{free} is the minimum free Hamming distance of the convolutional code. If the decoder makes an error in the pairwise comparison of two paths through its state space, c_d denotes the total number of information bit errors produced by the wrong paths of Hamming weight $d \geq d_{free}$ that diverge from the correct path and remerge to it at some stage later.

The term $\left(\frac{E_b}{N_o}\right)_e$ denotes the effective SNR per bit and is defined as the value of $\frac{E_b}{N_o}$ in an equivalent constant signal-to-noise ratio additive white Gaussian noise (AWGN) channel which would yield the same P_{res} as the fading channel with prior bit level interleaving. For soft-decision Viterbi decoding of the de-interleaved bit stream of length N, $\left(\frac{E_b}{N_o}\right)_e$ can be estimated as [11]:

$$\left(\frac{E_b}{N_o}\right)_e \approx min_{\,k=0}^{\,N-D} \left\{ \sum_{i=0}^{D-1} \frac{\left(\frac{E_b}{N_o}\right)_{S(k+i)}}{D} \right\} \tag{2}$$

Here D denotes the ratio of constraint length and coding rate of the convolutional coder and $\left(\frac{E_b}{N_o}\right)_{S(x)}$ stands for the $\frac{E_b}{N_o}$ in time slot $S(x)$.

$S(x)$ models the interleaving scheme adopted in the UMTS. It denotes the number of the time slot relative to the beginning of the TTI in which bit x of the bit stream $\forall\ x = 0 \cdots N-1$ is transmitted. For an interleaving scheme working with TTIs of N_I radio frames, with each radio frame containing B_R bits and N_S time slots, $S(x)$ can be described by the following expression:

$$\begin{aligned} S(x) = N_S \left[N_I \left\lfloor \frac{x}{N_I B_R} \right\rfloor + \sum_{i=0}^{2} \left(\left\lfloor \frac{N_I}{2^{i+1}} \right\rfloor \left\lfloor \frac{x - N_I B_R \left\lfloor \frac{x}{N_I B_R} \right\rfloor}{2^i} \right\rfloor_{mod\ 2} \right) \right] \\ + \left\lfloor \frac{1}{2} \sum_{i=0}^{2} \left(\left\lfloor \frac{2N_S}{2^{i+1}} \right\rfloor \left\lfloor \frac{\left\lfloor \frac{x}{N_I} - B_R \left\lfloor \frac{x}{N_I B_R} \right\rfloor \right\rfloor}{2^i} \right\rfloor_{mod\ 2} \right) \right\rfloor \end{aligned} \tag{3}$$

3.2 Post Decryption Bit Error Probability

Analogous to encryption, decryption is performed by the exclusive-OR of the ciphertext with the generated keystream. Observation of figure 1 shows that this process involves using block cipher KASUMI in output feedback mode. The feedback path is however modified by block counter and static data held in register AA. Clearly, if bit errors occur in the ciphertext, then the recovered plaintext will have the same number of bit errors in the same bit positions as in the ciphertext. In this condition, the decryptor is said to be propagating bit errors. Furthermore, if there is a bit error in the cipher synchronization counter $COUNT$, then a bit error may occur independently, in any bit position of the decryption of the corresponding ciphertext, with an expected error rate of fifty percent [7]. In this state, decryptor is said to be expanding errors. Bit error expansion is because of the fact that the underlying block cipher KASUMI adheres to strict avalanche criterion (SAC) [8] implying that each bit of its output function changes with probability one half, whenever an input bit is complemented.

If N_c denotes the length of physical layer counter block and L is the length of corresponding ciphertext block, then we can associate four possible events with the reception of these blocks as follows:

Suppose D is the event when both counter block and ciphertext block are in error. C is the event when ciphertext block is correct while counter block is in error. B is the event when counter block is correct while ciphertext block is in error. A is the event when both counter block and ciphertext block are correct. When event D or C happens, the decryptor is in the state of error expansion. When event B takes place, propagation of bit errors occurs. When event A happens, the decryptor is free of errror expansion and propagation. Hence depending on the state of the received ciphertext and counter blocks at each decryption cycle, the decryptor is in one of the three states namely "error free", "error propagation" and "error expansion". If we refer to these states as **0**, **1** and **2** respectively, then the state diagram of the stochastic error model for the decryption process can be expressed as illustrated in figure 2.

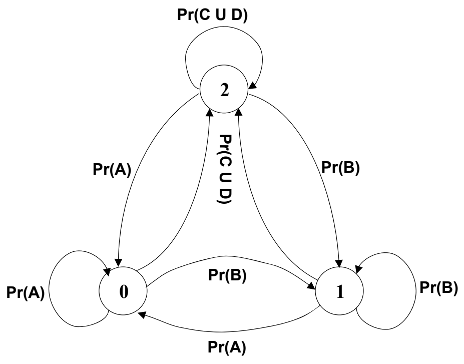


Fig. 2. State diagram of the decryption process

The stochastic process updates its state every decryption cycle with the transition probabilities indicated in figure 2 whereby $Pr(X)$ denotes the probability of occurrence of event X. Since the transition probability of the next state only depends on the value of the current state, the stochastic error model is a Markov chain. The state transition matrix of the model can be determined from $Pr(A), Pr(B), Pr(C)$ and $Pr(D)$. We can also express $Pr(A)$, $Pr(B)$, $Pr(C)$ and $Pr(D)$ in terms of P_{res} as follows:

Let C_i denote the i-th bit of the counter block $\forall\ i = 1..N_c$. Let $C_1C_2C_3...C_{N_c}$ and $C_1^{'}C_2^{'}C_3^{'}...C_{N_c}^{'}$ be the transmitted and received counter blocks respectively. Let $P(C_i^{'}|C_i)$ denote the probability of receiving $C_i^{'}$ when C_i is transmitted and $P(C_1^{'}C_2^{'}C_3^{'}...C_{N_c}^{'}|C_1C_2C_3...C_{N_c})$ denote the probability that that the received counter block is $C_1^{'}C_2^{'}C_3^{'}...C_{N_c}^{'}$ when the transmitted block is $C_1C_2C_3...C_{N_c}$. Assuming reception of each bit is independent of all the remaining bits then:

$$P(C_1^{'}C_2^{'}C_3^{'}...C_{N_c}^{'}|C_1C_2C_3...C_{N_c}) = P(C_1^{'}|C_1)\cdot P(C_2^{'}|C'_2)\cdot P(C_3^{'}|C'_3)...P(C_{N_c}^{'}|C_{N_c}^{'}) \tag{4}$$

If in a certain received block, j bits are in error, then $N_c - j$ bits are correct. Then the probability of receiving this block is $P_{res}^{j} \cdot (1 - P_{res})^{N_c - j}$. There are $\binom{N_c}{j}$ different ways in which j errors can occur in N_c bits. Hence:

$$P(receiving\ j\ out\ of\ N_c\ bits\ in\ error) = \binom{N_c}{j} P_{res}^{j} \cdot (1 - P_{res})^{N_c - j} \tag{5}$$

and the probability P_c of receiving correct counter block is:

$$P_c = (1 - P_{res})^{N_c} \tag{6}$$

The probability P_{ct} of receiving correct ciphertext block can similarly be given as:

$$P_{ct} = (1 - P_{res})^{L} \tag{7}$$

Since $Pr(A)$ denotes the probability of event when both counter block and ciphertext block are correct, therefore:

$$Pr(A) = P_c \cdot P_{ct} = (1 - P_{res})^{N_c} \cdot (1 - P_{res})^{L} \tag{8}$$

Similarly:

$$Pr(B) = (1 - (1 - P_{res})^{L}) \cdot (1 - P_{res})^{N_c} \tag{9}$$

$$Pr(C) = (1 - P_{res})^{L} \cdot (1 - (1 - P_{res})^{N_c}) \tag{10}$$

$$Pr(D) = (1 - (1 - P_{res})^{L}) \cdot (1 - (1 - P_{res})^{N_c}) \tag{11}$$

Also, events C and D are mutually exclusive, therefore we can write:

$$P(C \cup D) = P(C) + Pr(D) = 1 - (1 - P_{res})^{N_c} \tag{12}$$

Hence the transition probability matrix can be written as follows:

$$\mathbf{P} = \begin{bmatrix} (1 - P_{res})^{N_c}(1 - P_{res})^{L} & (1 - (1 - P_{res})^{L})(1 - P_{res})^{N_c} & 1 - (1 - P_{res})^{N_c} \\ (1 - P_{res})^{N_c}(1 - P_{res})^{L} & (1 - (1 - P_{res})^{L})(1 - P_{res})^{N_c} & 1 - (1 - P_{res})^{N_c} \\ (1 - P_{res})^{N_c}(1 - P_{res})^{L} & (1 - (1 - P_{res})^{L})(1 - P_{res})^{N_c} & 1 - (1 - P_{res})^{N_c} \end{bmatrix} \tag{13}$$

In the above equation the element p_{ij} of the transition probability matrix denotes the probability of moving from state i to j $\forall\ i, j = 0, 1, 2$.

If π_0 , π_1 and π_2 are the steady-state probabilities of being in states **0**, **1** and **2** respectively and e_0, e_1 and e_2 are the respective bit error rates associated with these states then the mean probability of error can be calculated as:

$$P_e = \pi_0 \cdot e_0 + \pi_1 \cdot e_1 + \pi_2 \cdot e_2 \tag{14}$$

As block cipher KASUMI adheres to SAC, the bit error rate in state **2** is $\frac{1}{2}$. Also, the bit error rates in states **0** and **1** are 0 and P_{res} respectively. Furthermore π_0 , π_1 and π_2 can be determined using the Markov chain steady state properties [9]. Using these results, the post decryption error probability can be written as:

$$P_e = P_{res}(1-(1-P_{res})^L)(1-P_{res})^{N_c} + \frac{1}{2}(1-(1-P_{res})^{N_c}) \tag{15}$$

4 Numerical Results

Figure 3 shows the analytical post decryption error probability plotted against the channel bit error probability for RLC AM, RLC UM and MAC layer ciphering modes. The dotted line represents the situation when plaintext stream is passed unencrypted through the channel i.e. $P_e = P_{res}$. As evident from (15), the post decryption error probability is proportional to the size of the counter and is therefore maximum for RLC AM and minimal for MAC layer ciphering.

4.1 Bit Error Expansion

The post decryption effect of inherent bit expansion property of the KASUMI is clearly visible. For a given P_{res}, it can be measured by the vertical distance between the curves representing encrypted and unencrypted cases in Figure 3. For a fixed N_c, the error expansion is approximately constant for P_{res} less than 10^{-1}. Furthermore, P_e saturates at $\frac{1}{2}$ when $P_{res} = 0.5$. At this saturation point, event D occurs predominantly i.e. received counter block is always in error causing decryptor to garble output with an error rate of fifty percent.

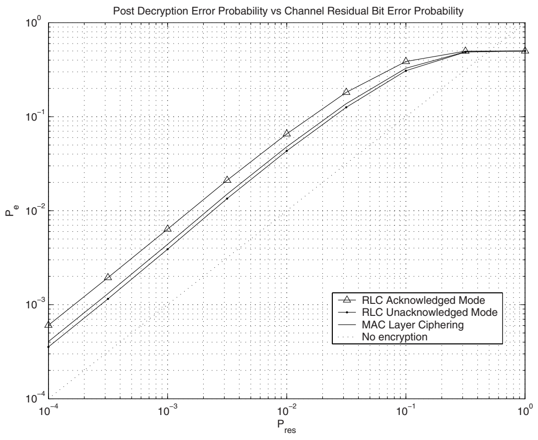


Fig. 3. Analytical post decryption error probability for various encryption modes

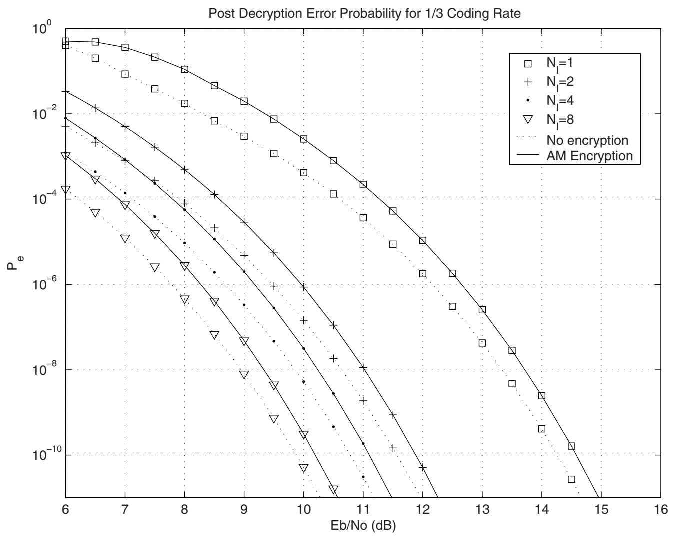


Fig. 4. Post decryption error probability for coding rate $\frac{1}{3}$ and different interleaving spans

4.2 Effects of UMTS Uu Interface Parameters on Decryption

We next determine the effects of UMTS Uu interface parameters (interleaving span and coding rate) on the post decryption performance. The pre and post decryption bit error probabilities are determined under following assumptions:

1. The mobile call is conducted from a vehicle driving at a speed of 50 km/hr i.e. the Doppler frequency is 100 Hz and the fades frequently span more than a full radio frame (10 ms).This implies that interleaving mitigates the effect of correlated burst errors thereby resulting in independent bit errors
2. The payload size is 512 bits and it is encrypted in RLC AM.
3. Radio frame duration is 10ms with 16 slots per frame.
4. Rate $\frac{1}{3}$ and $\frac{1}{2}$ convolutional codes of constraint length 9 are considered.
5. Interleaving spans over 1,2,4 and 8 TTIs (10,20,40 and 80ms respectively).

With effective $\frac{E_b}{N_o}$ as the parameter, the pre and post decryption probabilities are plotted based on the above stated assumptions. Figure 5 shows the results for coding rate $\frac{1}{3}$ and different interleaving spans whereas Figure 6 shows the results for coding rate $\frac{1}{2}$. From these results we note that:

1. For a fixed coding rate and interleaving span, applying encryption while maintaining the same bit error rate as without encryption induces an $\frac{E_b}{N_o}$ penalty. The additional $\frac{E_b}{N_o}$ can be determined from the horizontal distance between pre and post decryption curves.

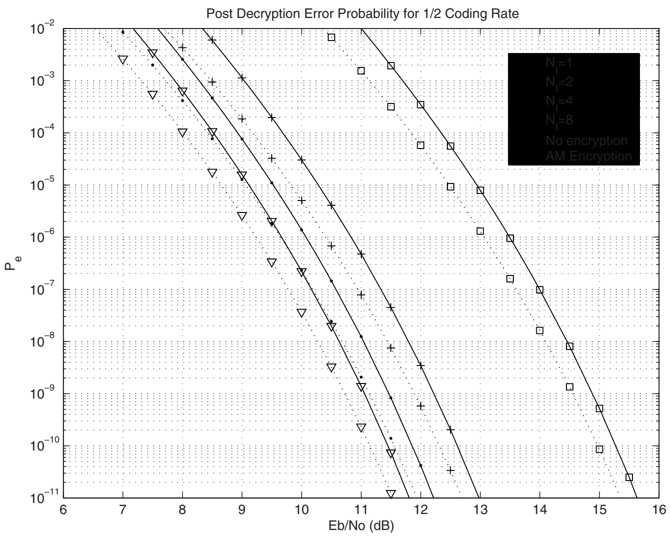


Fig. 5. Post decryption error probability for coding rate $\frac{1}{2}$ and different interleaving spans

2. Decreasing the rate of the convolutional code mitigates the post decryption error effect to some extent. This however comes at the expense of decrease in the bandwidth efficiency. For $\frac{1}{2}$ channel coding, the bandwidth efficiency is approximately 50%, whereas it reduces to approximately 25% with rate $\frac{1}{3}$ code.
3. Increasing the interleaving span also mitigates the post decryption error effect. This however brings on the processing and consequently packet delay.

5 Conclusions

In this paper, UMTS decryption process is modeled as a finite-state Markov chain and post decryption probability of error is investigated. The error probability is found to be proportional to the size of the synchronization counter transmitted in the cleartext headers of UMTS payloads. Impact of Uu parameters such as coding rate, interleaving span and $\frac{E_b}{No}$ on the post decryption errors has been analyzed. Results can be used to determine an optimum combination of transmission SNR, coding rate and interleaving span, to achieve target bit error rate while applying enryption.

References

1. Ferland, G., Chouinard, J.-Y.: Error Rate Performance Analysis of Stream and Block Ciphers in a Digital Mobile Communication Channel. In: 1992 Vehicle Navigation and Information Systems Conference (VNIS 1992), Oslo, Norway, pp. 426–433 (1992)

2. Reason, J.: End-to-end Confidentiality for Continuous-media Applications in Wireless Systems, Doctoral Dissertation. UC Berkeley (2000)
3. 3GPP Technical Specification Group Services and System Aspects, 3G Security: Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 1: f8 and f9 Specification, 3GPP TS 35.201 V6.0.0 (2004)
4. 3GPP Technical Specification Group Services and System Aspects, 3G Security: Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification, 3GPP TS 35.202 V6.0.0 (2004)
5. 3GPP Technical Specification Group Radio Access Network: Medium Access Control (MAC) protocol specification, 3GPP TS 25.321 V6.3.0 (2004)
6. 3GPP Technical Specification Group Radio Access Network: Radio Link Control (RLC) protocol specification, 3GPP TS 25.322 V6.2.0 (2004)
7. 3GPP Security Algorithms Group of Experts (SAGE): Report on the Evaluation of 3GPP Standard Confidentiality and Integrity Algorithms (SAGE version 2.0), 3GPP KASUMI Evaluation Report (2001)
8. Webster, A.F., Tavares, S.E.: On the design of S-boxes. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 523–534. Springer, Heidelberg (1986)
9. Ochi, M.K.: Applied Probability and Stochastic Processes in Engineering and Physical Sciences. John Wiley and Sons Inc., Chichester (1992)
10. Poppe, F., De Vleeschauwer, D., Petit, G.H.: Guaranteeing Quality of Service to Packetised Voice over the UMTS Air Interface. In: Proceedings of the Eight International Workshop on Quality of Service (IWQoS 2000), Pittsburgh (2000)
11. Nanda, S., Rege, K.M.: Frame Error Rates for Convolutional Codes on Fading Channels and the Concept of Effective Eb/No. IEEE Transactions on Vehicular Technology 47(4), 1245–1250 (1998)

A Simple, Smart and Extensible Framework for Network Security Measurement

Feng Cheng[1], Christian Wolter[1,2], and Christoph Meinel[1]

[1] Hasso Plattner Institute (HPI), University of Potsdam, 14482, Potsdam, Germany
{feng.cheng,christoph.meinel}@hpi.uni-potsdam.de
[2] SAP Research, Vincenz-Priessnitz-Str. 1, 76131, Karlsruhe, Germany
christian.wolter@sap.com

Abstract. Several efficient tools have emerged to aim at auditing and measuring the security of a computer system or an internal network. Along with the increasing complexity of network attacks, these tools become more and more complicated. Even so, most of them can only do simple snapshot analysis of the current system and are incapable of identifying possible attacks whose preconditions are not fulfilled at the beginning but may be possible during the further attack progression. This paper proposes a new framework for the security measurement that commits complex attack sequences and does stateful inspection of the target environment. The framework consists of five core components: Information Gatherer, Knowledge Base, Interaction Agent, Evaluation Engine and User Interface. An easy-to-use tool, called SNAPP, is realized based on the proposed framework. The dependencies among each attack step in an attack sequence revealed by SNAPP can be easily expressed using Attack Graphs which assist to make security evaluations of the testing environment. Several experiments that actually simulate and perform some well known penetration attacks using SNAPP are presented and analyzed for comparison and measurement of current security methods, such as the conventional filtering-based firewalls and our patented Lock-Keeper technology, which is an implementation of the high-level security concept "Physical Separation".

1 Introduction

Modern business models depend on providing applications to external customers and business partners while these applications, varying from the simple e-mail services to collaborative business processes, are commonly required to be exposed to untrusted networks, e.g. the Internet. As a result, malicious attackers may also gain the possibilities to intrude the network by misusing the providing service. In general, the more services an internal network exposes, the more attack opportunities, and the more likely it will be a target of attacks [1], [2].

The industry has responded to this problem by increasing efforts on deploying more powerful security methods. However, how can they determine if such efforts are paying off, and how can consumers perceive if such efforts have made a difference? Performing a throughout security audit has been recognized as an efficient solution for this demand [2]. Several open source software as well as some commercial products, such as Nessus [3], Core Impact [4], Metasploit [5], and SAINT [6], etc., have been developed for auditing the security of systems and networks. Most of them rely on huge vulnerability databases in which all the registered vulnerabilities are treated and inspected individually. Regarding the complex nature of some sophisticated network attack sequences, the limitation of these available tools to identify attacks that are based on a combination of known vulnerabilities has been a main challenge.

On the other hand, to match the need of testing some newly appeared and more sophisticated network attacks, the security auditing tools have to be designed and realized in more complex and intricate ways. It made using (installing, configuring, executing and extending) such tools more and more difficult. To this effect, the security auditing tools normally can only be operated by security domain experts. For general requirements, e.g., to assist a customer to test and choose a personal firewall, these tools would be relatively hard to use. The technical analysis and evaluation results provided by most of these tools are overly complex and hard to be understood by nonprofessionals. In addition, regular usage of target systems or networks may be disturbed due to some fatal configuration changes done by the security auditing tools or the simulated attacks itself.

In our own case, we need an efficient and specific tool to measure the security of our proposed high-level security solution, Lock-Keeper [7], [8], which is realized based on the simple security concept "Physical Separation" (PS). Theoretically, it's clear and easy to understand the security benefits of the PS idea because it can completely prevent any kinds of direct connections between the remote hackers with their vulnerable targets, e.g. a high-level secure internal network or a sensitive host. This connection-based communication is the basic model for most network attacks. However, to practically evaluate the Lock-Keeper solution, especially to measure and compare it with conventional filtering-based firewalls, we intend to simulate and perform some popular network penetration attacks in an easy and controllable way.

This paper addresses the above mentioned questions by proposing a framework that can automatically identify and commit complex network attack sequences. Based on the proposed framework and its concepts on "information gathering", "attack method selection" and "attack progression"[2], an simple and easy-to-use tool, called SNAPP, i.e., Smart Network Attack and Penetration Platform, is implemented. In comparison to some other available security auditing tools, SNAPP has a more flexible architecture and a simpler user interface that can even be managed by non-security experts. In particular, the simple *Knowledge Base* data structure in SNAPP makes it easier to construct Attack Graphs ([9], [10]) accordingly, which helps to make final security evaluations.

The rest of this paper is organized as follows. Popular security auditing tools are introduced as related works in the next section. Then Section 3 proposes the new framework of network security measurement. The SNAPP implementation is described in brief as well in this section. Several experiments are shown in

Section 4 to demonstrate applicability of the proposed SNAPP tool. A short conclusion and some possible future works are given in the last section.

2 Related Works

In order to evaluate the security of their systems or networks, security experts prefer to use the same tools which hackers would use on attacking. These tools can not only simplify the detection of security holes but also educate the user on different security topics. Some integrated and menu-driven auditing tools, including high-quality commercial products, have been proposed in this area. This section briefly describes two of the well-known tools: Nessus and Core Impact.

Nessus is a free program designed to automate the testing and discovery of known security vulnerabilities [3]. The first alpha release was published on April 4th, 1998. The extendability through the plugin mechanism is one of the key features of Nessus. Plugins are written in the scripting language NASL (Nessus Attack Scripting Language) [11]. Nessus comes with a huge community that regularly provides new security testing plugins. In practice, Nessus has been a popular and state-of-the-art tool widely used in the area of security consulting. It analyzes every host in a network for exploitable services, misconfiguration, or information leakage.

Nessus supports lots of simple attacks, but is not able to express dependencies between each attack step or construct more complex attacks by evaluating the situation after an attack takes place. The new attacks that were impossible at the beginning are ignored completely. The overall analysis is always related to a static snapshot of the environment. Figure 1 shows a typical Nessus testing process. Nessus works with the initial information about the target host, including the installed services or the patch level, which will be matched against a database of known vulnerabilities. No repetitive scanning is performed by Nessus. All generated information is based only on a certain snapshot taken at a certain moment. In spite of resulting in a lot of security related information, Nessus does not perform further evaluation of these information.

Core Impact [4] is a commercial product for security auditing. It can construct complex attack sequences by interacting with its environment at runtime. This is indicated by a loop shown in Figure 2. The strength of Core Impact lies in the ability to commit the evaluation not only for the original attacked host, but to transplant agents to the penetrated host and start further evaluation. This is achieved via *Syscall Proxying* [12].

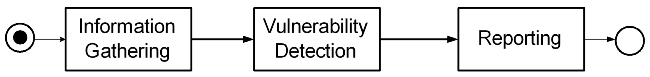


Fig. 1. Testing Process - Nessus

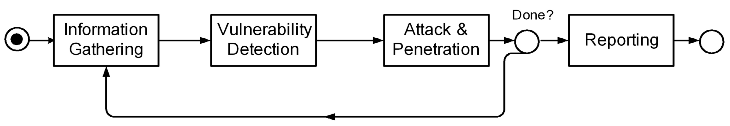


Fig. 2. Testing Process - CORE IMPACT

One of the biggest drawbacks of Core Impact is its incapability to monitor the environment changes during runtime. Similar to Nessus, Core Impact starts from an initial snapshot of the target environment, but may create complex attack sequences than Nessus does. Unfortunately, it does not consider the normal network traffic as an information source during the attack process. Thus, if no service is detected at the beginning, Core Impact will consider the network as safe and stop testing. In this case, network monitoring is completely missing.

In addition, such available tools as Nessus and Core Impact are mostly designed to expectedly be used by professionals. It is hard for regular users without strong security knowledge and enough IT-related skills to install, perform and maintain them.

3 A New Framework for Network Security Measurement

Unlike other security auditing tools mentioned previously, our proposed framework is capable of generating and evaluating complex attack sequences which consist of small interaction steps. The dependencies between them can also be stored accordingly. Our work is initially motivated by the intention on finding an easy way to practically demonstrate the security benefits of our patented "Physical Separation" based Lock-Keeper technology. For this reason, the architecture of the framework is designed as simple as possible. To be capable of easily adding new attacks or vulnerabilities, the architecture should be extensible.

3.1 Framework Overview

As shown in Figure 3, the new security measurement framework consists of five main parts: Information Gatherer, Knowledge Base, Interaction Agent, Evaluation Engine, and User Interface.

The Information Gatherer is responsible for network monitoring. It runs during the whole testing process and aims at collecting information about network topology, hosts, dataflow, and available services. Every piece of information gathered will be saved in Knowledge Base. The Knowledge Base is a data structure containing all information related to the target networks or hosts. The Evaluation Engine is informed whenever a change to the Knowledge Base occurs and tests if one or more Interaction Agent can perform an activity by using the currently available information. If an Interaction Agent confirms that all his prerequisites are satisfied, it propagates its capabilities to the Knowledge Base.

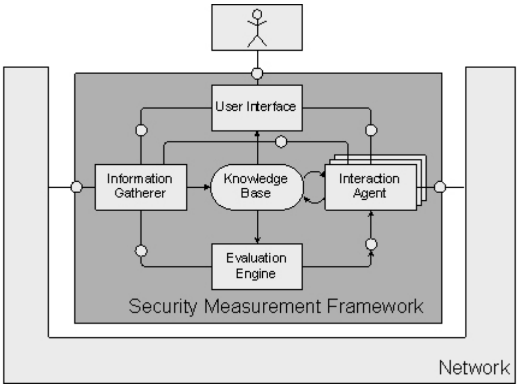


Fig. 3. New Framework

Each successful agent interaction would possibly enlarge the Knowledge Base. The User Interface allows to visualize the whole process and makes the usability of the framework easier.

3.2 Information Gatherer

Every discovered information, such as available services, opened ports, and user passwords, etc., will be gathered by this Information Gatherer module and then stored in a global Knowledge Base. As shown in Figure 4, Information Gatherer consists of a number of small collectors connected to the target network. Each collector is specialized on revealing a certain kind of information, such as a service detection or a password sniffing. All collectors are controlled by a coordinator. To avoid spamming the same information multiple times, each collector owns a local Knowledge Base (Local KB). No local information will be reported more than twice to the coordinator process. A *collector interface* that contains a method to start and stop the collector process has to be implemented each single collector. The coordinator calls these methods and manages a process list to guarantee complete control over the collector processes. In addition, the interface defines a report method that allows information exchange between the collectors and the coordinator. The coordinator assigns a special communication channel to each created collector. This channel is used to report newly discovered information. Because the collectors and the coordinator have to know which kinds of information have been known and what can be reported, both need to access a collection of information type descriptors. A descriptor defines how a certain information type has to be computed and reported. Every collector comes with a configuration file that allows fine grained setup of the collector's behavior.

3.3 Knowledge Base

The Knowledge Base contains all available information about the target network. The structure of the Knowledge Base can be illustrated by the entity relationship

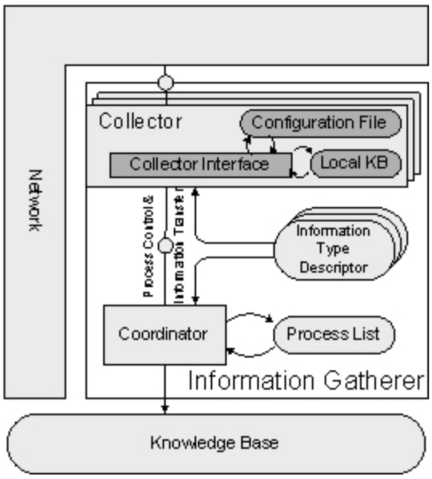


Fig. 4. Information Gatherer

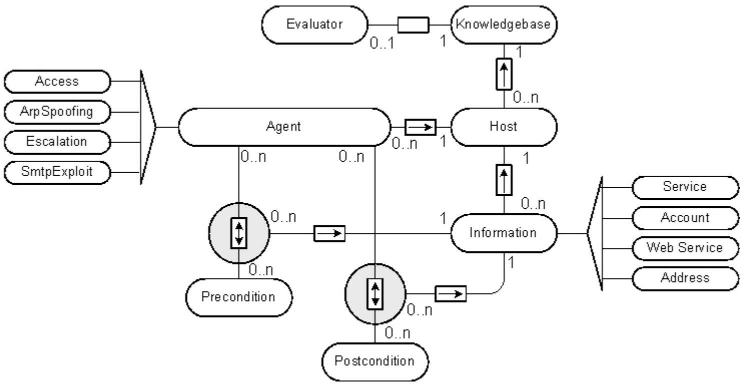


Fig. 5. Knowledge Base: Entity Relation

diagram shown in Figure 5. Every possible transit into a new state needs to be evaluated according to the information stored in the Knowledge Base.

The Knowledge Base is a structured storage that again consists of 0..n information containers, because these containers store information per host. They are called *host* containers. Each host container is assigned to a single Knowledge Base and holds 0..n detailed information units, which may consist of general information about network addresses, user accounts, and available services on that host. Additionally, every host container could have 0..n registered attack agents, such as as File Access Agent, or an Escalation Agent, which might be possibly executed based on the currently discovered host information stored information units. Figure 6 shows a simple example of the Knowledge Base. The

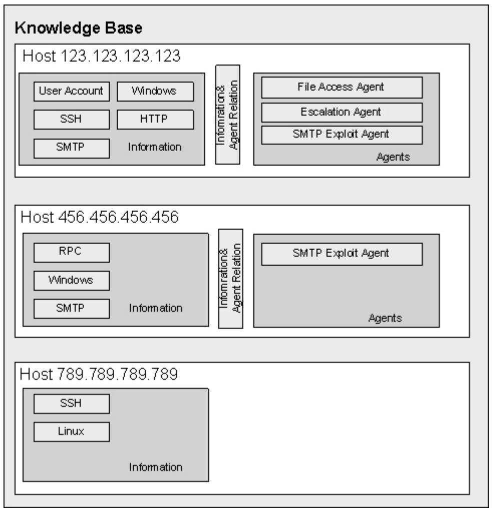


Fig. 6. Knowledge Base - An Example

clear structure of Knowledge Base makes it easier to construct an Attack Graph accordingly.

3.4 Interaction Agent

As mentioned above, an attack sequence consists of one or more interactions. Each agent proposes a set of preconditions that have to be fulfilled to execute the interaction. Additionally, an agent stores the outcome as its postconditions in Knowledge Base, so that it can be used as input for other agents. Agents can be classified and assigned to a special interaction domain by their postconditions. Therefore, agents within the same domain provide identical postconditions, but the requirements (i.e., preconditions) and the concrete implementations are

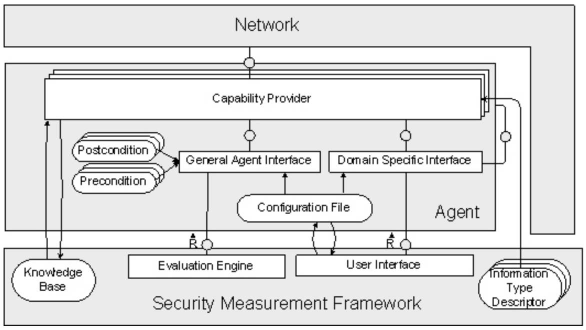


Fig. 7. Interaction Agent - Architecture

different. For instance, one agent requires an available secure shell service, while another may require an available FTP service, but both may provide the information that *file access* is available on that host and share the same agent domain. The detailed architecture of an agent is shown in Figure 7. Capability Provider is the core element of each agent and realizes the concrete interaction. To execute the interaction, the Capability Provider gets the required information that is defined in the agent's precondition set from the global Knowledge Base. After a successful interaction the agent propagates additional information to the global Knowledge Base, such as what have been defined in the agent's set as postconditions. The General Agent Interface is used by the Evaluation Engine to determine if all of this agent's preconditions are currently satisfied. Other agents and the user can utilize the information by calling the domain specific functionality through the Domain Specific Interface.

3.5 Evaluation Engine

The Evaluation Engine checks if the current set of information matches with one or more sets of preconditions defined by interaction agents. Whenever the

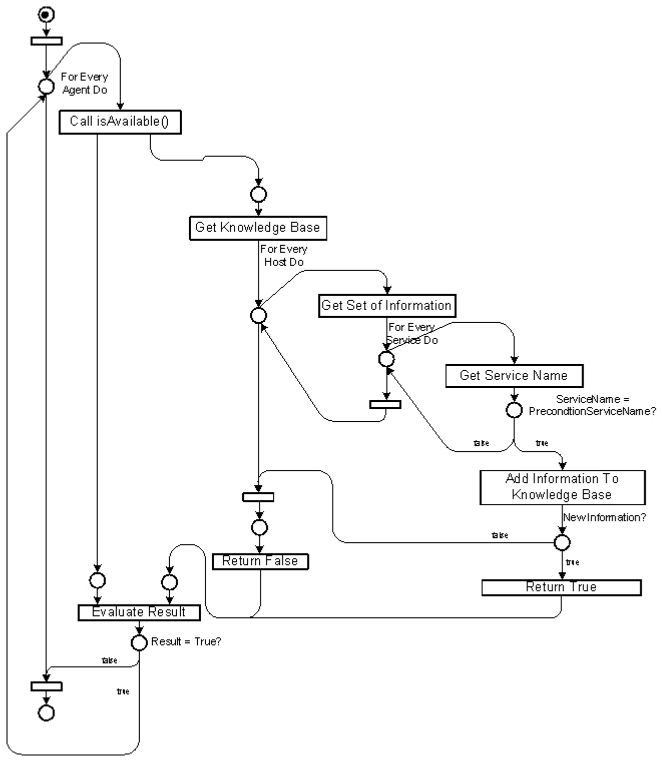


Fig. 8. Precondition Evaluation

preconditions of an agent are fulfilled, the agent is created and the corresponding information generated by its interaction will be propagated to the global Knowledge Base. Because this increased Knowledge Base may enable other agents, the Evaluation Engine will repeat this step until no more information can be added. To determine if an agent could be enabled, all information of the host are used to match the preconditions of an agent. If preconditions are satisfied, the test returns true and the postconditions of this agent will be added in the host's field in the Knowledge Base. This process can be simplified in two steps: Firstly, only one single precondition is tested, not the whole set. Secondly, after new information is added the evaluation process is repeated recursively to exclude already tested agents rather than in an iterative way, such as the process shown in Figure 8.

3.6 User Interface

The User Interface displays all available information extracted from the global Knowledge Base. Attack Graphs are also expected to be drawn in this User Interface to visualize the global Knowledge Base and help further security evaluation. Using an event-driven approach, the User Interface is updated whenever the current state changes. The user can start or stop information gathering and available interaction agents. Every important functionality can be accessed directly through the User Interface. Because the whole framework is extensible, the User Interface adopts to the current setup at framework initialization. Menus and dialogs could be adjusted, if new agents or collectors are added without changing the source codes. To support this dynamic adoption, the User Interface contains a dialog generator. As described earlier, every collector or agent comes with a

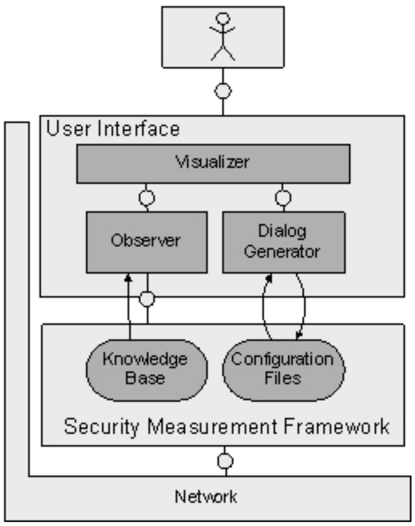


Fig. 9. User Interface

configuration file. The configuration files contain information about what parameters a collector or an agent may understand, for instance, a packet sniffer may filter certain ports or protocol types. Therefore, *port* and *protocol* are used as parameters to setup this filter. The dialog generator reads all available configuration files and renders a specific configuration dialog that shows the possible parameters during framework initialization (cf. Figure 9). These parameters will be passed to the collectors and agents when they are invoked. The user interface simply renders the current Knowledge Base. An observer is registered to be informed whenever the Knowledge Base changes so that new information changes can be rendered immediately.

3.7 SNAPP Implementation

The SNAPP (Smart Network Attack and Penetration Platform) is the implementation of the proposed security measurement framework. Main parts of the framework are written in Perl (Practical Extraction and Report Language). The graphical user interface (GUI) is implemented using the C++ library Qt [13]. The collectors and agents are either written in pure Perl or written in any other language which can be accessed through a Perl wrapper. SNAPP works in an Unix environment and is practically developed under SuSE Linux [14]. The GUI provided by SNAPP comes with three generators for a dynamic generation of dialogs at runtime, a global Knowledge Base and a gatherer object as a network monitor. The gatherer holds a reference to the global Knowledge Base. The generated dialogs are used to manipulate the default settings of all available agents, collector classes, information units, and SNAPP itself. Because the number of agents, collectors, and information units may be enlarged by future extensions, the dialog generators dynamically adjust to these changes at system start-up.

The gatherer consists of several collectors, each with its own local Knowledge Base. A collector's local Knowledge Base information will additionally be reported to the global Knowledge Base. The collector class is an abstract class. Specialized monitoring classes derive from the abstract collector and implement their own information retrieval behavior. Currently, three different collectors are realized, i.e. Pdump [15], Arpdiscover [16] and Nmap [17]. Pdump uses packet sniffing to be aware of existing hosts and discover available services, such as HTTP, e-mail, or Web Services. It is capable of discovering e-mail accounts or passwords transmitted in plain texts. In addition, Web Service messages can be intercepted and the available services extracted. Arpdiscover is able to detect hosts by using ARP (Address Resolution Protocol) packet sniffing in case of new hosts appear in the network. The well known fingerprinting and scanning tool, Nmap, has also been integrated into SNAPP to perform in-depth host analysis for service identification. Nmap is accessed by a special Perl interface that allows direct usage of Nmap's functionalities.

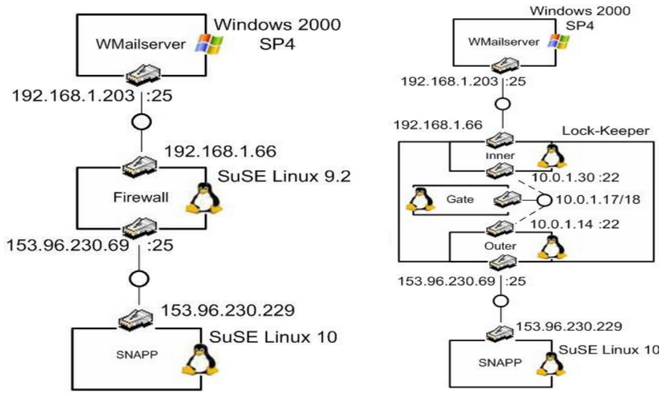
The Knowledge Base consists of host-based information units and a list of activated agents. Whenever the information content of the global Knowledge Base is changed, an associated evaluator will check the available agent implementations and create a new agent to perform a certain interaction with the available information of the global Knowledge Base. Currently, four different interaction agents are implemented. Besides the assigned agent information, each host object contains a list of host-related information. SNAPP implements information units by creating specialized assurance objects, derived from the abstract class assurance. The host contains information about available services, user account, web services, and addresses. Each assurance class contains detailed information about the host. This information will be accessed by the agents to perform their interactions.

4 Experiments

In this section, several experiments are illustrated to demonstrate the applicability of the proposed security measurement framework. Two testing scenarios are proposed to compare the effectiveness of two different security solutions, i. e. the packet-filtering firewall and the Lock-Keeper. Both mechanisms are intended to secure an internal network, which exposes an e-mail service to the untrusted external network. Two Attack Graphs are drawn according to the corresponding Knowledge Base.

4.1 General Setup

The test environment consists of two separate networks. A trusted internal network *192.168.1.0* and an untrusted external network *153.96.230.0*. Both networks are separated by a certain security mechanism to protect the internal network. Scenario I uses a packet-filtering firewall. Scenario II utilizes the Lock-Keeper. In both scenarios the internal network consists of an e-mail server and the external network consists of a host running the SNAPP application. This



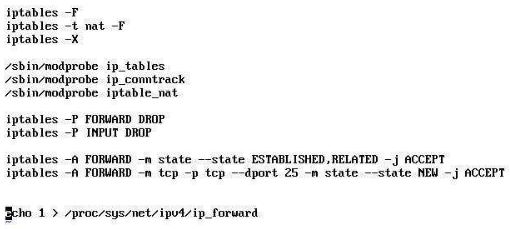
(a) Scenario I: Firewall Protection (b) Scenario II: Lock-Keeper Protection

Fig. 10. General Setup of Testing

general setup is shown in Figure 10. The WMailserver [18] used as the e-mail providing server, i.e. a mail server, is running on a Windows host (Win2000 SP4). The SNAPP application is running on a SuSE Linux operating system.

4.2 Scenario I

This scenario uses a packet-filtering firewall to protect the exposed e-mail service. Thus, the firewall defines an *iptables* [19] rule to drop all kinds of packets that are not arriving on port 25. All traffic on port 25 that emerges from an internal address is allowed to pass through, as shown in Figure 11.



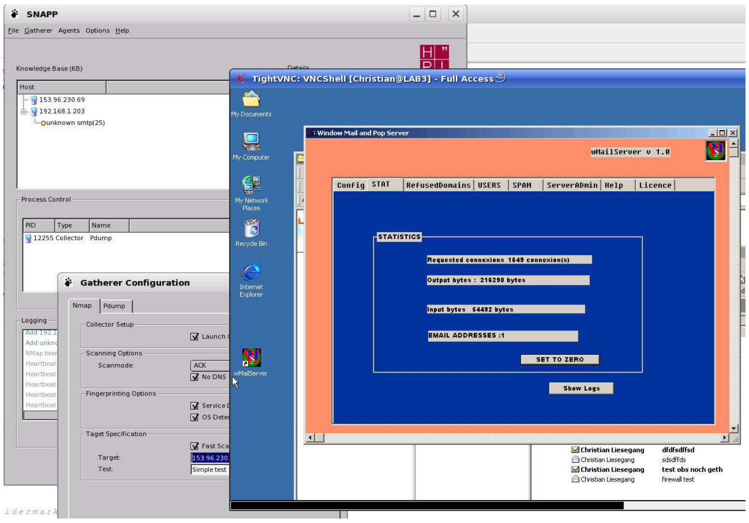
```
iptables -F
iptables -t nat -F
iptables -X

/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_nat

iptables -P FORWARD DROP
iptables -P INPUT DROP

iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m tcp -p tcp --dport 25 -m state --state NEW -j ACCEPT

echo 1 > /proc/sys/net/ipv4/ip_forward
```

Fig. 11. Firewall Configuration

Figure 12 shows the attack progression of the SNAPP application. At the beginning, the packet sniffer fetches the e-mail sent from the untrusted network to the internal e-mail server. The new information will be added to the Knowledge Base. The evaluation engine enables an agent that needs a host with open port 25 and a SMTP service running and commits a buffer overflow. The intercepted e-mail indicates that the target host fulfills all preconditions. The agent is invoked and tries to exploit the e-mail service using a corrupted SMTP packet to the e-mail server. The firewall will handle this packet as regular SMTP traffic. The SMTP packet contains a special payload that produces a buffer overflow. The payload contains an instruction to establish a reverse TCP connection over port 25 to the SNAPP application. This connection is used by a VNC (Virtual Network Computing) [20] server loaded into the memory of the windows host. As indicated in Figure 12, the attack results in a VNC session with the rights of the exploited e-mail service.

4.3 Scenario II

The second scenario uses the Lock-Keeper, instead of a firewall, to protect the internal e-mail server. The Lock-Keeper acts as a proxy. All e-mails are sent to the Lock-Keeper and will be forwarded to the internal e-mail server. Again, SNAPP intercepts an e-mail send to the e-mail server. Because the proxy is not transparent to its environment, SNAPP discovers the Lock-Keeper and the available e-mail deamon accepting e-mails and forwarding them to the internal network. At this time, the escalation agent will be activated for the Lock-Keeper, because



Requested connexions 1649 connexion(s)
Output bytes : 216290 bytes
Input bytes 54492 bytes
EMAIL ADDRESSES :1

Christian Liesegang dfdfsdffsd
Christian Liesegang sdsdffds
Christian Liesegang test obs noch geth
Christian Liesegang firewall test

Fig. 12. Scenario I - VNC Session

the Lock-Keeper fulfills all required preconditions. The special crafted e-mail is sent to the Lock-Keeper and forwarded into the internal network. The malicious payload tries to establish a backward connection to the SNAPP application. The connection establishment fails. The Lock-Keeper prevents a successful connection establishment, due to its inherent feature on TCP isolation. The exploitation interaction fails.

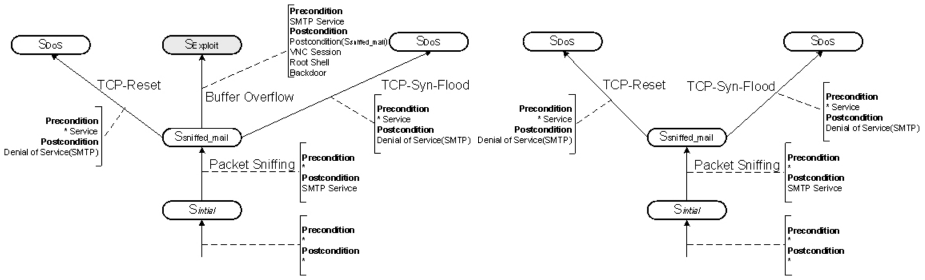
4.4 Result Analysis

These two simple scenarios are designed to verify the applicability of the proposed security measurement framework and the possibility to use *Attack Graph* as a metric to represent the realtime Knowledge Base generated by SNAPP.

Figure 13(a) depicts the resulting graph for scenario I. The critical state is $s_{exploit}$, because it enlarges the current Knowledge Base with several dangerous new capabilities. The successful exploit establishes a VNC session for direct access and a root shell for listening on a specific port. In comparison, there is no $s_{exploit}$ state in Figure 13(b) because the Lock-Keeper successfully prevents the backward connection.

The above mentioned two experiments performed by SNAPP, have demonstrated that our newly proposed security measurement framework is smart and has the capability to evaluate complex attack sequences and execute real time network monitoring to gather information during the attack progression.

The testing process of SNAPP shown in Figure 14, can make more thorough measurement than the previously mentioned Nessus and Core Impact solution. The simple User Interface makes it an easy-to-use tool to compare network architectures with different security mechanisms, such as firewalls and Lock-Keepers



(a) Scenario I: Firewall Exploiting (b) Scenario II: Lock-Keeper Exploiting

Fig. 13. Attack Graphs Results

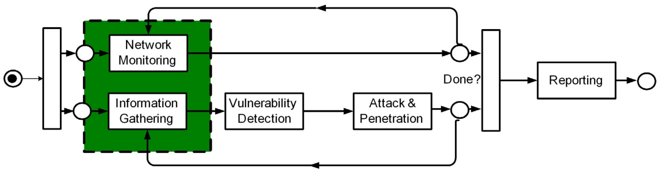


Fig. 14. Testing Process – SNAPP

in our experiments. Thanks to the clear structure of the Knowledge Base, Attack Graphs can be easily constructed to help us make further theoretical analysis and evaluation. Moreover, the flexible and comprehensible architecture makes it easy to extend the framework, for example, introducing new Information Gatherer, adding new atomic attack agents, etc.

5 Conclusions

This paper proposes a simple, smart, and extensible framework for network security measurement. Besides, an easy-to-use tool, SNAPP, is introduced as an implementation of the proposed framework. Using this framework, all the information exploited from former executed attack can be added into the global Knowledge Base at runtime so that new attacks can be performed based on it. After such throughout auditing, the vulnerabilities of the target system or network can easily represented by the Knowledge Base. With this complete and well-structured Knowledge Base, formal attack graphs can be expressed to help us make the security evaluation of the target system. However, there are still some further works to be done around this framework. More information gatherers and attack agents, are always useful to extend the functionality and performance of SNAPP. A graph-related module to visualize the global Knowledge Base, for instance automatically constructing and analyzing an Attack Graph, can be integrated into the framework. A security module, e.g. a user access control system, to manage different levels of SNAPP users is also necessary.

References

1. Tanenbaum, A.S.: Computer Networks, 4th edn. Prentice-Hall, Englewood Cliffs (2003)
2. Internet Security Systems (ISS): Web Application Protection: Using Existing Protection Solutions. Technical White Paper, Atlanta, U.S.A (July 2002)
3. Nessus Website (1998-2007), `http://www.nessus.org`
4. Core Impact Website (2003-2007), `http://www.coresecurity.com`
5. Metasploit Website (2003-2007), `http://www.metasploit.com`
6. SAINT Website (2007), `http://www.saintcorporation.com`
7. Cheng, F., Meinel, Ch.: Research on the Lock-Keeper Technology: Architectures, Applications and Advancements. International Journal of Computer & Information Science 5(3), 236–245 (2004)
8. Lock-Keeper WebSite of Siemens Switzerland (2005-2007), `http://www.siemens.ch`
9. Jha, S., Wing, J.: Survivability Analysis of Networked Systems. In: Proceedings of the ICSE 2001, Toronto, Canada (May 2001)
10. Sheyner, O.: Scenario Graphs and Attack Graphs. Ph.D. Dissertation, CMU-CS-04-122, Computer Science Department, Carnegie Mellon (April 2004)
11. Deraison, R.: The Nessus Attack Scripting Language Reference Guide (2002)
12. Caceres, M.: Syscall Proxying-Simulating Remote Execution, Technical Report, CORE SECURITY Technologies (2002)
13. Trolltech: The Qt C++ Class Library (2006), `http://www.trolltech.com`
14. Novell Tech: Introducing SUSE Linux Enterprise 10 (2006), `http://www.novell.com`
15. packetstormsecurity.org: Packet Sniffer: Pdump (2000), `http://www.packetstormsecurity.org/sniffers/pdump/`
16. freshmeat.net: Arp Tools: ARP Discover, ARP Flood, and ARP Poison (2007), `http://www.freshmeat.net/projects/arptools/`
17. Insercure.org: Nmap Security (2007), `http://www.insecure.org/nmap/`
18. SoftiaCom: WMailserver (2007), `http://www.softiacom.com`
19. Netfilter.org: Netfilter - IP Tables (1999-2007), `http://www.netfilter.org`
20. Richardson, T., Stafford-Fraser, Q., et al.: Virtual Network Computing. IEEE Internet Computing 2(1), 33–38 (1998)

Author Index