Pil Joong Lee
Jung Hee Cheon (Eds.)

LNCS 5461

Information Security and Cryptology – ICISC 2008

11th International Conference
Seoul, Korea, December 2008
Revised Selected Papers

Springer

Lecture Notes in Computer Science 5461

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Pil Joong Lee   Jung Hee Cheon (Eds.)

Information Security and Cryptology – ICISC 2008

11th International Conference
Seoul, Korea, December 3-5, 2008
Revised Selected Papers

Springer

Volume Editors

Pil Joong Lee
Pohang University of Science and Technology (POSTECH)
Department of Electronic and Electrical Engineering
San 31 Hyoja-dong, Nam-gu, Pohang, Kyungbuk 790-784, Korea
E-mail: pjl@postech.ac.kr

Jung Hee Cheon
Seoul National University, Department of Mathematical Sciences
599 Gwanakno, Gwanak-gu, Seoul 151-742, Korea
E-mail: jhcheon@snu.ac.kr

Library of Congress Control Number: Applied for

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

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-642-00729-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-00729-3 Springer Berlin Heidelberg New York

springer.com

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12631810 06/3180 5 4 3 2 1 0

Preface

ICISC 2008, the 11th International Conference on Information Security and Cryptology, was held in Seoul, Korea, during December 3–5, 2008. It was organized by the Korea Institute of Information Security and Cryptology (KIISC). The aim of this conference was to provide a forum for the presentation of new results in research, development, and applications in the field of information security and cryptology. It also served as a place for research information exchange.

The conference received 131 submissions from 28 countries, covering all areas of information security and cryptology. The review and selection processes were carried out in two stages by the Program Committee (PC) of 62 prominent researchers via online meetings, using the We-Submission-and-Review software written by Shai Halevi, IBM. First, at least three PC members blind-reviewed each paper, and papers co-authored by the PC members were reviewed by at least five PC members. Second, individual review reports were revealed to PC members, followed by detailed interactive discussion on each paper. Through this process, the PC finally selected 26 papers from 14 countries. The acceptance rate was 19.8%. The authors of selected papers had a few weeks to prepare for their final versions based on the comments received from more than 136 external reviewers. These revised papers were not subject to editorial review and the authors bear full responsibility for their contents.

The conference featured one tutorial and two invited talks. The tutorial was given by Masayuki Abe from NTT. The invited speakers for two talks were Vincent Rijmen from K.U.L. & Graz University of Tech and Jong-Deok Choi from Samsung Electronics.

There are many people who contributed to the success of ICISC 2008. We would like to thank all the authors who submitted papers to this conference. We are deeply grateful to all 62 members of the PC members. It was a truly nice experience to work with such talented and hard-working researchers. We wish to thank all the external reviewers for assisting the PC in their particular areas of expertise. Thanks to Shai Halevi for allowing us to use their convenient software. Finally, we would like to thank all the participants of the conference who made this event an intellectually stimulating one through their active contribution.

December 2008

Pil Joong Lee
Jung Hee Cheon

ICISC 2008

The 11th Annual International Conference
on Information Security

December 3–5, 2008
Sungkyunkwan University, Seoul, Korea

Organized by
Korea Institute of Information Security and Cryptology (KIISC)
http://www.kiisc.or.kr

In cooperation with
Ministry of Knowledge Economy (MKE)
http://www.mke.go.kr

General Chair

Hong-sub Lee	KIISC, Korea

Program Co-chairs

Pil Joong Lee	POSTECH, Korea
Jung Hee Cheon	Seoul National University, Korea

Program Committee

Joonsang Baek	Institute for Infocomm Research, Singapore
Liqun Chen	Hewlett-Packard Laboratories, UK
Nicolas T. Courtois	University College London, UK
Michel Cukier	University of Maryland, USA
Frederic Cuppens	Telecom Bretagne, France
Bart De Decker	Katholieke Universiteit Leuven, Belgium
Mario Marques Freire	University of Beira Interior, Portugal
Philippe Golle	Palo Alto Research Center, USA
Guang Gong	University of Waterloo, Canada
Vipul Goyal	UCLA, USA
Kil-Chan Ha	Sejong University, Korea
Eduardo B. Fernandez	Florida Atlantic University, USA
Dowon Hong	ETRI, Korea
Jin Hong	Seoul National University, Korea
Seokhie Hong	Korea University, Korea
Stanislaw Jarecki	University of California, Irvine, USA
Jaeyeon Jung	Intel Research, USA

Kwangjo Kim	ICU, Korea
Yongdae Kim	University of Minnesota, Twin Cities, USA
Christopher Kruegel	University of California, Santa Barbara, USA
Taekyoung Kwon	Sejong University, Korea
Byoungcheon Lee	Joongbu University, Korea
Dong Hoon Lee	Korea University, Korea
Mun-Kyu Lee	Inha University, Korea
Yingjiu Li	Singapore Management University, Singapore
Chae Hoon Lim	Sejong University, Korea
Javier Lopez	University of Malaga, Spain
Keith Martin	Royal Holloway, University of London, UK
Sjouke Mauw	University of Luxembourg, Luxembourg
Atsuko Miyaji	JAIST, Japan
SangJae Moon	Kyungpook National University, Korea
David Naccache	Ecole Normale Superieure, France
Jesper Buus Nielsen	Aarhus University, Denmark
DaeHun Nyang	Inha University, Korea
Tatsuaki Okamoto	NTT, Japan
Rolf Oppliger	eSECURITY Technologies, Switzerland
Paolo D'Arco	University of Salerno, Italy
Je Hong Park	ETRI, Korea
Sangjoon Park	Sungkyunkwan University, Korea
Sangwoo Park	ETRI, Korea
Jacques Patarin	Versailles University, France
Raphael C.-W. Phan	Loughborough University, UK
Bart Preneel	Katholieke Universiteit Leuven,Belgium
Jean-Jacques Quisquater	UCL, Belgium
Vincent Rijmen	K.U.L., Belgium and Graz University of Technology, Austria
Ahmad-Reza Sadeghi	Ruhr University Bochum, Germany
Kouichi Sakurai	Kyushu University, Japan
Mahmoud Salmasizadeh	Sharif University of Technology, Iran
Palash Sarkar	Indian Statistical Institute, India
JungHwan Song	Hanyang University, Korea
Rainer Steinwandt	Florida Atlantic University, USA
Willy Susilo	University of Wollongong, Australia
Tsuyoshi Takagi	Future University Hakodate, Japan
Yukiyasu Tsunoo	NEC Corporation, Japan
Jozef Vyskoc	VaF s.r.o., Slovakia
Sung-Ming Yen	National Central University, Taiwan
Jeong Hyun Yi	Samsung, Korea
Hyunsoo Yoon	KAIST, Korea
Dae Hyun Yum	POSTECH, Korea
Moti Yung	Google Inc. and Columbia University, USA
Fangguo Zhang	Google Inc. and Columbia University, USA
Alf Zugenmaier	DoCoMo Euro-Labs, Germany

Organizing Chair

Seungjoo Kim	Sungkyunkwan University, Korea

Organizing Committee

Dong Kyue Kim	Hanyang University. Korea
Ki Young Moon	ETRI, Korea
Chang-Seop Park	Dankook University, Korea
Young Ik Eom	Sungkyunkwan University, Korea
Heekuck Oh	Hanyang University, Korea
Dong Hoon Lee	Korea University, Korea
Im-Yeong Lee	Soonchunhyang University, Korea

External Reviewers

Frederik Armknecht
Maryam Rajabzadeh Assar
Man Ho Au
Jean-Philippe Aumasson
Fabien Autrel
Behnam Bahrak
Shane Balfe
Lejla Batina
Aurelie Bauer
Robin Berthier
Jean-Luc Beuchat
Annalisa De Bonis
Antoon Bosselaers
Yacine Bouzida
Wouter Castryck
Dario Catalano
Donghoon Chang
Ku Young Chang
Byong-Deok Choi
Jae Tark Choi
Jeong Woon Choi
Sherman Chow
Danielle Chrun
Carlos Cid
Ed Condon
Nora Cuppens-Boulahia
M. Prem Laxman Das
Ton van Deursen
Gwenael Doerr
Dang Nguyen Duc
Thomas Eisenbarth
Yehia Elrakaiby
Jonathan Etrog
Kazuhide Fukushima
Xinxin Fan
Joaquin Garcia-Alfaro
Benedikt Gierlichs
Henri Gilbert
Yoshiaki Hori
JaeCheul Ha
Daewan Han
Honggang Hu
Xinyi Huang
Junbeom Hur
Jung Yeon Hwang
Sebastiaan Indesteege
Hoda Jannati
Keith Jarrin
Ikrae, Jeong
Nam-su Jho
Takeshi Kawabata
Chano Kim
Jihye Kim
Jongsung Kim
Takayuki Kimura
Divyan M. Konidala
Heejin Park
Hiroyasu Kubo
Jeong Ok Kwon
Jorn Lapon
HoonJae Lee
Ji-Seon Lee
JongHyup Lee
Jun Ho Lee
Jung-Keun Lee
Kwangsu Lee
Minsoo Lee
Youngsook Lee
Jin Li
Wei-Chih Lien
Hsi-Chung Lin
Kuan-Jen Lin
Hans Loehr
Carolin Lunemann
Florian Mendel
Hideyuki Miyake
Abedelaziz Mohaisen
Amir Moradi
Tomislav Nad
Vincent Naessens
Akira Nozawa
Satoshi Obana
Chihiro Ohyama

Katsuyuki Okeya
Claudio Orlandi
Kyosuke Osaka
Omkant Pandey
Jun Pang
Chanil Park
Hyun-A Park
YongSu Park
Young-Ho Park
Axel Poschmann
Roberto De Prisco
Sasa Radomirovic
Christian Rechberger
Mohammad Reza Reyhanitabar
Chunhua Su
Minoru Saeki
Teruo Saito
Somitra Kumar Sanadhya
Riccardo Scandariato
Martin Schläffer
Jae Woo Seo
Maki Shigeri
Masaaki Shirase
Haya Shulman
Masakazu Soshi
Miroslava Sotakova
Takahiko Syouji
Tamer
Julien Thomas
Joe-Kai Tsay
Etsuko Tsujihara
Frederik Vercauteren
Pieter Verhaeghe
Ivan Visconti
Camille Vuillaume
Christian Wachsmann
Yamin Wen
Jian Weng
Bo-Ching Wu
Chi-Dian Wu
Lingling Xu
OkYeon Yi
Yves Younan
Ng Ching Yu
Aaram Yun
Chang-An Zhao
Xingwen Zhao
Tieyan Li

Sponsoring Institutions

BCQRE
Chungnam National University Internet Intrusion Response Technology Research Center (CNU IIRTRC), Korea
Electronics and Telecommunications Research Institute (ETRI), Korea
IglooSecurity, Korea
Korea Electronics Technology Institute (KETI), Korea
Korea Information Security Agency (KISA), Korea
BK21 Information Security in Ubiquitous Environment, Korea
Mobile Network Security Technology Research Center (MSRC), Korea
LG-CNS, Korea
LOTTE Data Communication Company, Korea
SNU-BK21 Mathematical Sciences Division, Korea
Sungkyunkwan University Authentication Technology Research Center (SKKU ARTC), Korea

Table of Contents

Public Key Encryption

Key Management and Secret Sharing

Privacy and Digital Rights

Digital Signature and Voting

Side Channel Attack

Hash and MAC

Primitives and Foundations

Block and Stream

Simple CCA-Secure Public Key Encryption from Any Non-Malleable Identity-Based Encryption

Takahiro Matsuda[1], Goichiro Hanaoka[2], Kanta Matsuura[1], and Hideki Imai[2,3]

[1] The University of Tokyo, Tokyo, Japan
{tmatsuda,kanta}@iis.u-tokyo.ac.jp
[2] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
{hanaoka-goichiro,h-imai@}@aist.go.jp
[3] Chuo University, Tokyo, Japan

Abstract. In this paper, we present a simple and generic method for constructing public key encryption (PKE) secure against chosen ciphertext attacks (CCA) from identity-based encryption (IBE). Specifically, we show that a CCA-secure PKE scheme can be generically obtained by encrypting $(m||r)$ under identity "$f(r)$" with the encryption algorithm of the given IBE scheme, assuming that the IBE scheme is non-malleable and f is one-way. In contrast to the previous generic methods (such as Canetti-Halevi-Katz), our method requires stronger security for the underlying IBE schemes, non-malleability, and thus cannot be seen as a direct improvement of the previous methods. However, once we have an IBE scheme which is proved (or can be assumed) to be non-malleable, we will have a PKE scheme via our simple method, and we believe that the simpleness of our proposed transformation itself is theoretically interesting. Our proof technique for security of the proposed scheme is also novel. In the security proof, we show how to deal with certain types of decryption queries which cannot be handled by straightforwardly using conventional techniques.

Keywords: public key encryption, identity-based encryption, IND-CCA security, non-malleability, NM-sID-CPA security.

1 Introduction

1.1 Background

Studies on constructing and understanding efficient public key encryption (PKE) schemes secure against chosen ciphertext attacks (CCA) [42,21] are important research topics in the area of cryptography.

In [15], Canetti, Halevi, and Katz showed a generic construction of CCA secure PKE schemes from any semantically secure identity-based encryption (IBE) and one-time signature. This construction is fairly simple, and specifically, its ciphertext consists of (χ, vk, σ) where χ is a ciphertext of the underlying IBE scheme (under identity "vk"), vk is a verification key of a one-time signature scheme, and σ is a valid signature of χ (under verification key vk). However,

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 1–19, 2009.

due to the use of a one-time signature, ciphertext length of the resulting scheme becomes longer than that of the underlying IBE scheme for $|vk|$ and $|\sigma|$, which might result in significantly large ciphertexts. This method was later improved by Boneh and Katz [11], but its ciphertext length is still not sufficiently short compared with existing practical CCA secure PKE schemes, e.g. [19,32].

Hence, it is still desired to further shorten ciphertext length of the Canetti-Halevi-Katz (CHK) transformation without loosing generality.

1.2 Our Contribution

In this paper, we present a new *simple* IBE-to-PKE transformation which is fairly generic and practical. In contrast to the previous transformations [15,11] which require semantic security [26] for the underlying IBE scheme, our proposed method requires *non-malleability* [21].

Informally, for a given IBE scheme IBE, we generate a ciphertext χ of a PKE scheme which is converted from IBE via our method as follows:

$$\chi = \langle f(r), \mathsf{IBE.Enc}(\mathsf{prm}, \text{“}f(r)\text{”}, (m||r))\rangle,$$

where the second component is an encryption of $(m||r)$ with the encryption algorithm of IBE under the identity “$f(r)$”, and f is a one-way function (OWF). [1] It should be noticed that only a OWF is directly used as an additional building block and thus fairly simple while in [15,11] more complicated tools, e.g. one-time signatures, are required (though these tools can be obtained from OWFs in theory). As seen in the above construction, ciphertext overhead of our construction is that of IBE plus $|r| + |f(r)| =$ (256-bit) for 128-bit security, and this is fairly efficient compared to the Boneh-Katz (BK) construction [11].

An obvious and crucial disadvantage of our proposed transformation is that it requires a stronger assumption for the underlying IBE scheme, non-malleability. It is well known that non-malleability is a significantly stronger notion of security than semantic security, and in fact, except for CCA secure IBE schemes, no practical non-malleable IBE scheme is currently known. [2] Thus, we have to honestly remark that our proposal cannot be seen as a direct improvement of the previous generic IBE-to-PKE transformations [15,11]. However, once we have an IBE scheme which is proved (or can be assumed) to be non-malleable, an efficient CCA secure PKE scheme can be immediately obtained via our transformation. Also, we believe that the simpleness of our transformation itself is theoretically interesting.

Our proof technique for the proposed method will be of another theoretical interest. Since in the security proof, there exists a non-trivial issue which cannot be treated by straightforward application of known techniques, we have to concurrently carry out a totally different proof strategy. Hence, we develop a dedicated proof technique for handling two different strategies simultaneously.

[1] Details of the notations are explained in Section 2.

[2] In theory, it is possible to construct non-malleable IBE schemes generically from any semantically secure IBE schemes (while it is not known how to generically construct CCA secure IBE schemes). We discuss this in the full version of our paper.

Though there are several definitions for non-malleability so far [21,2,6,39,1], non-malleability of IBE our transformation requires is the weakest among them, i.e, (comparison-based) non-malleability against selective identity, chosen plaintext attacks (NM-sID-CPA security).

1.3 Related Works

CCA Security of PKE. The notion of CCA security of PKE scheme was first introduced by Naor and Yung [37] and later extended by Rackoff and Simon [42] and Dolev, Dwork, and Naor [21]. Naor and Yung [37] proposed a generic construction of non-adaptive CCA secure PKE schemes from semantically secure PKE schemes, using *non-interactive zero knowledge proofs* which yield inefficient and complicated structure and are not practical. Based on the Naor-Yung paradigm, some generic constructions of fully CCA secure PKE schemes were also proposed [21,43,33]. The first practical CCA secure PKE scheme is proposed by Cramer and Shoup [19], and they also generalized their method as *universal hash proof* technique [20] as well as some other instantiations of it. Kurosawa and Desmedt [32] further improved efficiency of the Cramer-Shoup scheme.

In [15], Canetti, Halevi, and Katz proposed a novel methodology for achieving CCA security (see below), and from this methodology some practical CCA secure PKE schemes are also produced, e.g. [12].

Recently, Peikert and Waters [41] proposed yet another paradigm to obtain CCA secure PKE schemes using a new primitive called *lossy trapdoor functions.*

In the random oracle methodology [3], several generic methodologies (e.g., [4,22,38]) and concrete practical schemes are known. However, since the results from several papers, such as [13], have shown that this methodology has some problem, in this paper we focus only on the constructions in the standard model.

IBE and the CHK Transformation. As mentioned above, one promising approach for constructing CCA secure PKE schemes is to transform an IBE scheme via the CHK paradigm. Here, we briefly review IBE schemes and applications of the CHK paradigm.

The concept of the identity-based encryption was introduced by Shamir [46]. Roughly speaking, an IBE scheme is a PKE scheme where one can use an arbitrary string (e.g., an email address) as one's public key. Boneh and Franklin [10] proposed a first efficient scheme (in the random oracle model) as well as the security models for IBE schemes. Sakai, Ohgishi, and Kasahara [44] independently proposed an IBE scheme with basically the same structure as the Boneh-Franklin IBE scheme (without proper security discussions). In the same year, Cocks [18] also proposed an IBE scheme based on the decisional quadratic residuosity problem. Horwitz and Lynn [28] introduced a notion of the hierarchical IBE (HIBE) which supports hierarchical structure of identities and Gentry and Silverberg [25] achieved the first scheme in the random oracle model. Canetti, Halevi, and Katz [14] introduced a weaker security model called *selective identity security*, and proposed an IBE scheme with this security without using random oracles. Boneh and Boyen [8] proposed two efficient selective identity secure IBE schemes in the standard model, and following this work, Boneh and Boyen [9] and Waters [47] proposed fully secure IBE schemes. Based on the Waters scheme (and thus based on the Boneh-Boyen scheme), many variants are proposed [35,17,30,16,45,31]. Gentry [24] proposed a practical IBE scheme which provides tight security reduction as well as anonymity of identities. A variant of this scheme was proposed in [31].

Canetti, Halevi, and Katz [15] proposed a generic method for obtaining CCA secure PKE schemes. Following [15], there have been some attempts to construct practical CCA secure PKE schemes by using specific algebraic properties of underlying IBE schemes, and especially, based on this approach Boyen, Mei, and Waters [12] proposed the currently best known CCA secure PKE schemes in terms of ciphertext length by using certain specific IBE schemes [8,47]. Boneh and Katz [11] improved the efficiency of [15]. Kiltz [29] showed that the CHK paradigm can be generically applied to *tag-based encryption* (TBE) schemes [34], which are weaker primitives than IBE schemes.

2 Preliminaries

Here, we review the definitions of the terms and security used in this paper.

Notations. In this paper, "$x \leftarrow y$" denotes that x is chosen uniformly at random from y if y is a set or x is output from y if y is a function or an algorithm. "$x||y$" denotes a concatenation of x and y. "$|x|$" denotes the size of the set if x is a finite set or bit length of x if x is an element of some set. "x-$\mathsf{LSB}(y)$" denotes x-least significant bits of y. For simplicity, in most cases we drop the security parameter (1^κ) for input of the algorithms considered in this paper.

2.1 Public Key Encryption

A public key encryption (PKE) scheme Π consists of the following three (probabilistic) algorithms:

$\mathsf{PKE.KG}$: A key generation algorithm that takes 1^κ (security parameter κ) as input, and outputs a pair of a secret key sk and a public key pk.

$\mathsf{PKE.Enc}$: An encryption algorithm that takes a public key pk and a plaintext $m \in \mathcal{M}$ as input, and outputs a ciphertext $\chi \in \mathcal{X}$.

$\mathsf{PKE.Dec}$: A decryption algorithm that takes a secret key sk and a ciphertext χ as input, and outputs a plaintext $m \in \mathcal{M} \cup \{\perp\}$.

where $\mathcal{M}$ and $\mathcal{X}$ are a plaintext space and a ciphertext space of Π, respectively. We require $\mathsf{PKE.Dec}(sk, \mathsf{PKE.Enc}(pk, m)) = m$ hold for all (sk, pk) output from $\mathsf{PKE.KG}$ and all $m \in \mathcal{M}$.

IND-CCA Security. Indistinguishability against adaptive chosen ciphertext attacks (IND-CCA) of a PKE scheme Π is defined using the following IND-CCA game between an adversary $\mathcal{A}$ and the IND-CCA challenger $\mathcal{C}$:

Setup. $\mathcal{C}$ runs $\mathsf{PKE.KG}(1^\kappa)$ and obtains a pair of sk and pk. $\mathcal{C}$ gives pk to $\mathcal{A}$ and keeps sk to itself.

Phase 1. $\mathcal{A}$ can adaptively issue decryption queries $(\chi_1, \chi_2, \ldots, \chi_{q_1})$ to $\mathcal{C}$ (at most q_1 times). $\mathcal{C}$ responds to each query χ_i by running $\mathsf{PKE.Dec}(sk, \chi_i)$ to obtain $m_i \in \mathcal{M} \cup \{\perp\}$ and returning m_i to $\mathcal{A}$.

Challenge. $\mathcal{A}$ chooses two distinct plaintexts (m_0, m_1) of equal length and sends them to $\mathcal{C}$. $\mathcal{C}$ flips a coin $b_C \in \{0, 1\}$ uniformly at random, computes a challenge ciphertext $\chi^* \leftarrow \mathsf{PKE.Enc}(pk, m_{b_C})$, and sends χ^* to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ can issue decryption queries in the same way as Phase 1 (at most q_2 times), except that $\mathcal{A}$ is not allowed to issue χ^* as a decryption query.

Guess. $\mathcal{A}$ outputs a bit b_A as its guess for b_C.

Let $q_D = q_1 + q_2$ be the number of $\mathcal{A}$'s decryption queries. We define the IND-CCA advantage of $\mathcal{A}$ attacking Π as: $\mathsf{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} = |\Pr[b_A = b_C] - \frac{1}{2}|$.

Definition 1. *We say that a PKE scheme Π is (t, q_D, ϵ)-IND-CCA secure if we have $\mathsf{Adv}^{\text{IND-CCA}}_{\Pi,\mathcal{A}} \leq \epsilon$ for any algorithm $\mathcal{A}$ running in time less than t and making at most q_D decryption queries.*

2.2 Identity-Based Encryption

An identity-based encryption (IBE) scheme Π consists of the following four (probabilistic) algorithms.

$\mathsf{IBE.Setup}$: A setup algorithm that takes 1^κ (security parameter κ) as input, and outputs a pair of a master secret key msk and global parameters prm.

$\mathsf{IBE.Ext}$: A key extraction algorithm that takes global parameters prm, a master secret key msk, and an identity $\mathsf{ID} \in \mathcal{I}$ as input, and outputs a decryption key d_{ID} corresponding to ID.

$\mathsf{IBE.Enc}$: An encryption algorithm that takes global parameters prm, an identity $\mathsf{ID} \in \mathcal{I}$, and a plaintext $m \in \mathcal{M}$ as input, and outputs a ciphertext $\chi \in \mathcal{X}$.

$\mathsf{IBE.Dec}$: A decryption algorithm that takes a decryption key d_{ID} and a ciphertext χ as input, and outputs a plaintext $m \in \mathcal{M} \cup \{\perp\}$.

where $\mathcal{I}$, $\mathcal{M}$, and $\mathcal{X}$ are an identity space, a plaintext space and a ciphertext space of Π, respectively. We require $\mathsf{IBE.Dec}(\mathsf{IBE.Ext}(\mathsf{prm}, \mathsf{msk}, \mathsf{ID}), \mathsf{IBE.Enc}(\mathsf{prm}, \mathsf{ID}, m)) = m$ hold for all $(\mathsf{msk}, \mathsf{prm})$ output from $\mathsf{IBE.Setup}$, all $\mathsf{ID} \in \mathcal{I}$, and all $m \in \mathcal{M}$.

NM-sID-CPA Security. Non-malleability against selective identity, chosen plaintext attacks (NM-sID-CPA) of an IBE scheme Π is defined using the following NM-sID-CPA game between an adversary $\mathcal{A}$ and the NM-sID-CPA challenger $\mathcal{C}$:

Init. $\mathcal{A}$ commits the target identity ID^*.

Setup. $\mathcal{C}$ runs $\mathsf{IBE.Setup}(1^\kappa)$ and obtains a pair of msk and prm. $\mathcal{C}$ gives prm to $\mathcal{A}$ and keeps msk to itself.

Phase 1. $\mathcal{A}$ can adaptively issue extraction queries $(\mathsf{ID}_1, \mathsf{ID}_2, \ldots, \mathsf{ID}_{q_1})$ to $\mathcal{C}$ (at most q_1 times), except that $\mathcal{A}$ is not allowed to issue the target identity ID^* as an extraction query. $\mathcal{C}$ responds to each query ID_i by running $\mathsf{IBE.Ext}(\mathsf{prm}, \mathsf{msk}, \mathsf{ID}_i)$ to obtain d_{ID_i} and returning d_{ID_i} to $\mathcal{A}$.

Challenge. $\mathcal{A}$ chooses a probabilistic distribution over an arbitrary subset of the plaintext space $\mathcal{M}^*(\subseteq \mathcal{M})$ where all elements in $\mathcal{M}^*$ are of equal length, and sends $\mathcal{M}^*$ to $\mathcal{C}$. $\mathcal{C}$ chooses m^* and $m^{\bar{*}}$ in $\mathcal{M}^*$ according to its distribution, computes a challenge ciphertext $\chi^* \leftarrow \mathsf{IBE.Enc}(\mathsf{prm}, \mathsf{ID}^*, m^*)$, sends χ^* to $\mathcal{A}$, and keeps $m^{\bar{*}}$ to itself. [3]

Phase 2. $\mathcal{A}$ can issue extraction queries in the same way as Phase 1 (at most q_2 times).

Output. $\mathcal{A}$ outputs a vector of ciphertexts $\overrightarrow{\chi}' = (\chi'_1, \chi'_2, \dots, \chi'_l)$, where each χ'_i is an encryption of m'_i under the target identity ID^* (i.e., $\chi'_i \leftarrow \mathsf{IBE.Enc}$ $(\mathsf{prm}, \mathsf{ID}^*, m'_i)$ for $1 \le i \le l$), and a description of a relation $R(\cdot,\cdot)$ of arity $(l+1)$, where the first input is a scalar and the second input is a vector of length l where l is polynomial in κ. $\mathcal{C}$ runs $d_{\mathsf{ID}^*} \leftarrow \mathsf{IBE.Ext}(\mathsf{prm}, \mathsf{msk}, \mathsf{ID}^*)$ to obtain d_{ID^*}, decrypts all elements in $\overrightarrow{\chi}'$ by running $m'_i \leftarrow \mathsf{IBE.Dec}(d_{\mathsf{ID}^*}, \chi'_i)$ for $1 \le i \le l$, and obtains $\overrightarrow{m}' = (m'_1, m'_2, \dots, m'_l)$.

Let $q_E = q_1 + q_2$ be the number of $\mathcal{A}$'s extraction queries. We define R^* as an event that $[\chi^* \notin \overrightarrow{\chi}' \wedge \bot \notin \overrightarrow{m}' \wedge R(m^*, \overrightarrow{m}') = \mathtt{true}]$. We also define $\mathsf{R}^{\bar{*}}$ in the same way as R^* except that m^* is replaced with $m^{\bar{*}}$. We then define the NM-sID-CPA advantage of $\mathcal{A}$ attacking Π as: $\mathsf{Adv}^{\text{NM-sID-CPA}}_{\Pi,\mathcal{A}} = \Pr[\mathsf{R}^*] - \Pr[\mathsf{R}^{\bar{*}}]$.

Definition 2. *We say that an IBE scheme Π is (t, q_E, ϵ)-NM-sID-CPA secure if we have $\mathsf{Adv}^{\text{NM-sID-CPA}}_{\Pi,\mathcal{A}} \le \epsilon$ for any algorithm $\mathcal{A}$ running in time less than t and making at most q_E extraction queries.*

Remark. NM-sID-CPA security of an IBE scheme we use in this paper is from [23,1]. This type of non-malleability is called *comparison-based* non-malleability [6,7], which was first introduced in [2] for PKE schemes and shown to be equivalent to or weaker than *simulation-based* non-malleability [21] depending on attacks. Note that our definition of the NM-sID-CPA game does not allow the adversary to output invalid ciphertexts (which decrypt to $\bot$). It was shown in [40] that this non-malleability is, depending on attacks, equivalent to or weaker than the one where the adversary is allowed to output invalid ciphertexts. Moreover, it was shown in [23] that the selective identity security is strictly weaker than adaptive identity security for IBE schemes. Thus, in summary, NM-sID-CPA security we use is the weakest among non-malleability for IBE schemes considered so far.

2.3 One-Way Function

Let $f : \mathcal{D} \to \mathcal{R}$ be a function. (In this paper, we only consider the case where all elements in $\mathcal{D}$ and $\mathcal{R}$ are of equal length.) We define the advantage of an adversary $\mathcal{A}$ against one-wayness of f as follows:

$$\mathsf{Adv}^{\text{OW}}_{f,\mathcal{A}} = \Pr[x \leftarrow \mathcal{D}; y \leftarrow f(x); x' \leftarrow \mathcal{A}(f, y) : f(x') = y].$$

[3] Without loss of generality, we assume $|\mathcal{M}^*| \ge 2$ and m^* and $m^{\bar{*}}$ are always distinct. The reason why we can assume this is similar to the case of the *indistinguishability* games where we can assume that an adversary always outputs two distinct messages in Challenge phase. For more details, see [2].

Definition 3. *We say that f is a (t, ϵ)-one-way function (OWF) if we have $\mathsf{Adv}^{\mathrm{OW}}_{f,\mathcal{A}} \leq \epsilon$ for any algorithm $\mathcal{A}$ running in time less than t.*

3 Proposed Transformation

In this section, we give the details of the construction of our simple IBE-to-PKE transformation from any NM-sID-CPA secure IBE scheme.

The idea behind the construction is as follows. Suppose f is a OWF. In our construction, a randomness r is encrypted as a part of a plaintext of the underlying non-malleable IBE scheme using $f(r)$ as an identity. In the decryption, the relation between r and $f(r)$ is then used to check the validity of the ciphertext. Constructed like this, it seems hard to make a valid ciphertext without knowing the exact value of r. Moreover, due to non-malleability of the IBE scheme and one-wayness of f, an adversary given a target ciphertext cannot make any alternation on it with keeping the consistency of r and $f(r)$.

3.1 Construction

Let $\Pi = (\mathsf{IBE.Setup}, \mathsf{IBE.Ext}, \mathsf{IBE.Enc}, \mathsf{IBE.Dec})$ be a non-malleable IBE scheme and $f : \{0,1\}^\gamma \to \mathcal{I}$ be a OWF, where $\mathcal{I}$ is the identity space of Π. Then we construct a PKE scheme $\Pi' = (\mathsf{PKE.KG}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ as in Fig. 1. Suppose the plaintext space of Π' is $\mathcal{M}_{\Pi'}$, then we require that the plaintext space $\mathcal{M}_\Pi$ of the underlying IBE scheme Π satisfy $\mathcal{M}_{\Pi'} \times \{0,1\}^\gamma \subseteq \mathcal{M}_\Pi$. We also require that length of all elements in $\mathcal{I}$, the output space of f as well as the identity space of Π, be of equal length and fixed. Typically, length γ of the randomness will be the security parameter κ.

In terms of the construction of the transformation, ours is fairly simpler compared to other generic IBE-to-PKE transformations [15,11], since only a OWF f, the weakest primitive, is directly used as an additional building block.

3.2 Security

Before going into a formal security proof, we give an intuitive explanation on how CCA security is proved. In the security proof, we construct a simulator which breaks NM-sID-CPA security using an IND-CCA adversary attacking the PKE scheme Π'. The simulator's task is to output a ciphertext y' and a relation R such that R holds between the plaintext of y' and that of simulator's challenge ciphertext y^*.

Roughly, the proof strategy of the previous generic IBE-to-PKE transformations [15,11] is that the decryption queries encrypted under identities different from the target identity of the simulator are responded perfectly due to simulator's own extraction queries, and the probability that an adversary issues a valid ciphertext under the target identity as a decryption query is bounded due to the properties of the underlying building blocks.

This "previous strategy" seems to work in our proof. But it is not sufficient because there seems to be a chance for the adversary to confuse our simulator by submitting a decryption query of the form $\langle f(r^*), y \rangle$ where f is a OWF, $f(r^*)$ is

<table>
<tr><td>$\mathsf{PKE.KG}(1^\kappa)$:
$(\mathsf{msk}, \mathsf{prm}) \leftarrow \mathsf{IBE.Setup}(1^\kappa)$
Pick a OWF f.
$SK \leftarrow \mathsf{msk}$, $PK \leftarrow (\mathsf{prm}, f)$
Output (SK, PK).</td><td>$\mathsf{PKE.Enc}(PK, m)$:
$r \leftarrow \{0,1\}^\gamma$; $\mathsf{ID} \leftarrow f(r)$
$y \leftarrow \mathsf{IBE.Enc}(\mathsf{prm}, \mathsf{ID}, (m||r))$
$\chi \leftarrow \langle \mathsf{ID}, y \rangle$
Output χ.</td></tr>
<tr><td colspan="2">$\mathsf{PKE.Dec}(SK, \chi)$:
Parse χ as $\langle \mathsf{ID}, y \rangle$.; $d_{\mathsf{ID}} \leftarrow \mathsf{IBE.Ext}(\mathsf{prm}, \mathsf{msk}, \mathsf{ID})$
$(m||r)$ / $\perp \leftarrow \mathsf{IBE.Dec}(d_{\mathsf{ID}}, y)$ (if $\perp$ then output $\perp$ and stop.)
Output m if $f(r) = \mathsf{ID}$. Otherwise output $\perp$.</td></tr>
</table>

Fig. 1. The Proposed IBE-to-PKE Transformation

submitted as a simulator's target identity, and $y \neq y^*$. Seeing such a query, the simulator cannot tell whether it is a valid ciphertext or not and only it can do is to return "$\perp$". If this query is a valid ciphertext, then the simulation for the adversary becomes imperfect by the improper response $\perp$ (if this is not the case, then the simulation is still perfect). However, notice that if the ciphertext of the form $\langle f(r^*), y \rangle$ where $y \neq y^*$ is valid, then the simulator can break NM-sID-CPA security by outputting y with a relation such that "the $|r^*|$-significant bits are mapped to the same value by f." Namely, suppose y is an encryption of $(m_A||r_A)$ under the target identity "$f(r^*)$", then a valid ciphertext satisfies $f(r_A) = f(r^*)$, which can be used for the relation R. (We call this "new strategy".)

The difficult point is that the simulator cannot know whether the decryption query under the target identity is a valid ciphertext or not when the adversary issues such a query. Therefore, we further show how to handle both the "previous" and "new" strategies so that our simulator can *always* gain the advantage of breaking NM-sID-CPA security from the adversary's IND-CCA advantage.

Theorem 1. *If the underlying IBE scheme Π is (t, q, ϵ_{nm})-NM-sID-CPA secure and f is a (t, ϵ_{ow})-OWF, then the proposed PKE scheme Π' is $(t, q, 2q(\epsilon_{nm} + \epsilon_{ow}))$-IND-CCA secure.*

Proof. Suppose $\mathcal{A}$ is an adversary that breaks (t, q, ϵ_{cca})-IND-CCA security of Π', which means that $\mathcal{A}$ with running time t makes at most q decryption queries and wins the IND-CCA game with probability $\frac{1}{2} + \epsilon_{cca}$. Then we construct a simulator $\mathcal{S}$ who can break $(t,\ q,\ \frac{1}{2q}\epsilon_{cca} - \epsilon_{ow})$-NM-sID-CPA security of the underlying IBE scheme Π using $\mathcal{A}$ and the (t, ϵ_{ow})-OWF f. We use the weakest case of NM-sID-CPA security where an attacker outputs a binary relation R and only a single ciphertext y' in Output phase, because it is sufficient for our proof. Without loss of generality, we assume $q > 0$. Our simulator $\mathcal{S}$, simulating the IND-CCA game for $\mathcal{A}$, plays the NM-sID-CPA game with the NM-sID-CPA challenger $\mathcal{C}$ as follows.

Setup. $\mathcal{S}$ generates a public key for $\mathcal{A}$ as follows. Pick a OWF f. Choose $r^* \in \{0,1\}^\gamma$ uniformly at random and compute $\mathsf{ID}^* \leftarrow f(r^*)$. Commit ID^* as $\mathcal{S}$'s target identity in the NM-sID-CPA game and obtain prm from $\mathcal{C}$. Give $PK = (\mathsf{prm}, f)$ to $\mathcal{A}$.

Phase 1. $\mathcal{S}$ responds to $\mathcal{A}$'s decryption queries $\chi_i = \langle \mathsf{ID}_i, y_i \rangle_{i \in \{1,\ldots,q_1\}}$ by returning m_i generated depending on ID_i as follows.

If $\mathsf{ID}_i = \mathsf{ID}^*$: Set $m_i = \bot$.

Otherwise: Issue ID_i as an extraction query to $\mathcal{C}$ and obtain d_{ID_i}. Compute $\mathsf{IBE.Dec}(d_{\mathsf{ID}_i}, y_i)$ and set $m_i = \bot$ if the decryption result is $\bot$. Otherwise, check whether $f(r_i) = \mathsf{ID}_i$ holds or not for the decryption result $(m_i||r_i)$. If this holds, then this m_i is used as a response to $\mathcal{A}$, otherwise, set $m_i = \bot$.

Challenge. When $\mathcal{A}$ submits (m_0, m_1) to $\mathcal{S}$, $\mathcal{S}$ returns the challenge ciphertext χ^* to $\mathcal{A}$ generated as follows. Flip a coin $b_S \in \{0,1\}$ uniformly at random. Choose a random message $m' \in \mathcal{M}_{\Pi'}$ (equal length to m_{b_S}). Choose $r' \in \{0,1\}^\gamma$ uniformly at random. Set $M_{b_S} = (m_{b_S}||r^*)$ and $M_{1-b_S} = (m'||r')$ $\in \mathcal{M}_\Pi$. Define $\mathcal{M}^*_\Pi$ as a uniform distribution over $\{M_0, M_1\}$. Submit $\mathcal{M}^*_\Pi$ to $\mathcal{C}$ as $\mathcal{S}$'s challenge and obtain y^* from $\mathcal{C}$. Give $\chi^* = \langle \mathsf{ID}^*, y^* \rangle$ to $\mathcal{A}$.

Phase 2. $\mathcal{S}$ responds to $\mathcal{A}$'s decryption queries in the same way as Phase 1.

Guess. $\mathcal{A}$ outputs b_A. $\mathcal{S}$ outputs a ciphertext y' and a description of a relation R depending on b_A as follows.

If $b_A = b_S$: Set a binary relation $R(\cdot,\cdot)$ as "$R(a,b) = \mathtt{true}$ iff $\gamma\text{-}\mathsf{LSB}(a) = \gamma\text{-}\mathsf{LSB}(b)$." Pick $m'' \in \mathcal{M}_{\Pi'}$ (equal length to m_{b_S}) randomly. Choose $r'' \in \{0,1\}^\gamma$ uniformly at random. Flip a biased coin $b_\alpha \in \{0,1\}$ where $b_\alpha = 1$ holds with probability α. If $b_\alpha = 1$, compute $y' \leftarrow \mathsf{IBE.Enc}(\mathsf{prm}, \mathsf{ID}^*, (m''||r^*))$, otherwise compute $y' \leftarrow \mathsf{IBE.Enc}(\mathsf{prm}, \mathsf{ID}^*, (m''||r''))$.

Otherwise: Set a binary relation $R(\cdot,\cdot)$ as "$R(a,b) = \mathtt{true}$ iff $f(\gamma\text{-}\mathsf{LSB}(a)) = f(\gamma\text{-}\mathsf{LSB}(b))$." Pick uniformly one ciphertext χ_i in $\mathcal{A}$'s decryption queries $\{\chi_j = \langle \mathsf{ID}_j, y_j \rangle\}_{j \in \{1,\ldots,q\}}$ and set $y' = y_i$.

We remain probability α unknown here, and discuss later in this proof. Note that $\Pr[b_\alpha = 1] = \alpha$ and $\Pr[b_\alpha = 0] = 1 - \alpha$, according to our definition. Note also that the description of the relation R that $\mathcal{S}$ uses is different depending on $\mathcal{A}$'s guess bit b_A.

Next, we estimate $\mathcal{S}$'s NM-sID-CPA advantage $\mathsf{Adv}^{\text{NM-sID-CPA}}_{\Pi,\mathcal{S}}$. In our construction of $\mathcal{S}$, $\mathcal{S}$'s challenge $\mathcal{M}^*_\Pi$ is always a uniform distribution over two messages. Thus, for convenience, we assume that the NM-sID-CPA challenger $\mathcal{C}$ flips its own coin $b_C \in \{0,1\}$ uniformly at random and chooses M_{b_C} as a challenge message M^* (and M_{1-b_C} as $\bar{M^*}$) in Challenge phase. Note that $\Pr[b_S = b_C] = \Pr[b_S \neq b_C] = \frac{1}{2}$ holds since b_C and b_S are independent. Note also that $\mathcal{S}$'s simulation for $\mathcal{A}$ becomes imperfect if $[b_S \neq b_C]$ occurs, since with overwhelming probability the challenge ciphertext given to $\mathcal{A}$ is not an encryption of either of (m_0, m_1) submitted by $\mathcal{A}$.

We say that a ciphertext χ is *valid* if χ decrypts to an element in the plaintext space $\mathcal{M}_{\Pi'}$ (i.e., not $\bot$) according to the decryption process of Π'. Let Valid be an event that $\mathcal{A}$ issues a decryption query which forms $\chi = \langle \mathsf{ID}^*, y \rangle$ and is *valid.* Note that $\mathcal{S}$'s simulation for $\mathcal{A}$ becomes imperfect if Valid occurs, because in this case $\mathcal{S}$ cannot return an appropriate plaintext to $\mathcal{A}$.

We also note that throughout the simulation $\mathcal{S}$ cannot know whether $[b_S \neq b_C]$ and Valid have occurred or not.

In the following, we consider six cases depending on b_A, b_S, b_C, and Valid:

- Case 1: $b_A = b_S \wedge b_S = b_C \wedge \overline{\mathsf{Valid}}$
- Case 2: $b_A = b_S \wedge b_S = b_C \wedge \mathsf{Valid}$
- Case 3: $b_A = b_S \wedge b_S \neq b_C$
- Case 4: $b_A \neq b_S \wedge b_S = b_C \wedge \overline{\mathsf{Valid}}$
- Case 5: $b_A \neq b_S \wedge b_S = b_C \wedge \mathsf{Valid}$
- Case 6: $b_A \neq b_S \wedge b_S \neq b_C$

These cases cover all possibilities. Let $[\textcircled{i}]$ denotes an event that Case i occurs. We denote $\mathcal{S}$'s advantage in Case i as Adv_i and define it as: $\mathsf{Adv}_i = \Pr[\mathsf{R}^* \wedge \textcircled{i}] - \Pr[\mathsf{R}^{\bar{*}} \wedge \textcircled{i}] = (\Pr[\mathsf{R}^* | \textcircled{i}] - \Pr[\mathsf{R}^{\bar{*}} | \textcircled{i}]) \cdot \Pr[\textcircled{i}]$, where R^* and $\mathsf{R}^{\bar{*}}$ are defined in Section 2.2. According to the definition of the NM-sID-CPA advantage, we obviously have $\mathsf{Adv}_{\Pi,\mathcal{S}}^{\text{NM-sID-CPA}} = \sum_{i=1}^{6} \mathsf{Adv}_i$.

Now, we introduce the following lemmas.[4] In the following, just for notational convenience, we define two conditional probabilities $P_v = \Pr[\mathsf{Valid} | b_S = b_C]$ and $P_k = \Pr[b_A = b_S | b_S = b_C \wedge \mathsf{Valid}]$, and use them for describing the lemmas.

Lemma 1. $\mathsf{Adv}_1 \geq \frac{1}{2}\alpha(\frac{1}{2} + \epsilon_{cca})(1 - P_v) - \frac{1}{2^\gamma} \Pr[\textcircled{1}]$.

Lemma 2. $\mathsf{Adv}_2 \geq \frac{1}{2}\alpha P_k P_v - \frac{1}{2^\gamma} \Pr[\textcircled{2}]$.

Lemma 3. $\mathsf{Adv}_3 \geq -\frac{1}{4}\alpha - \frac{1}{2^\gamma} \Pr[\textcircled{3}]$.

Lemma 4. $\mathsf{Adv}_4 \geq -\epsilon_{ow} \Pr[\textcircled{4}]$.

Lemma 5. $\mathsf{Adv}_5 \geq \frac{1}{2q}(1 - P_k)P_v - \epsilon_{ow} \Pr[\textcircled{5}]$.

Lemma 6. $\mathsf{Adv}_6 \geq -\epsilon_{ow} \Pr[\textcircled{6}]$.

Then, before proving the lemmas, we first calculate $\mathsf{Adv}_{\Pi,\mathcal{S}}^{\text{NM-sID-CPA}}$.

$$\begin{aligned}
\mathsf{Adv}_{\Pi,\mathcal{S}}^{\text{NM-sID-CPA}} &= \sum_{i=1}^{6} \mathsf{Adv}_i \\
&\geq \frac{1}{2}\alpha(\frac{1}{2} + \epsilon_{cca})(1 - P_v) + \frac{1}{2}\alpha P_k P_v - \frac{1}{4}\alpha + \frac{1}{2q}(1 - P_k)P_v \\
&\quad - \frac{1}{2^\gamma}(\Pr[\textcircled{1}] + \Pr[\textcircled{2}] + \Pr[\textcircled{3}]) - \epsilon_{ow}(\Pr[\textcircled{4}] + \Pr[\textcircled{5}] + \Pr[\textcircled{6}]) \\
&\geq \frac{1}{2}\alpha(\frac{1}{2} + \epsilon_{cca})(1 - P_v) + \frac{1}{2}\alpha P_k P_v - \frac{1}{4}\alpha + \frac{1}{2q}(1 - P_k)P_v - \epsilon_{ow},
\end{aligned}$$

where, in order to sum up the terms regarding $\Pr[\textcircled{i}]$ into one term ϵ_{ow} in the last inequality, we used the fact that $\sum_{i=1}^{6} \Pr[\textcircled{i}] = 1$ and the following claim.

Claim 1. $\frac{1}{2^\gamma} \leq \epsilon_{ow}$.

Proof of Claim 1. Consider the adversary $\mathcal{A}'$ against one-wayness of f who on input $f(r)$, where r is chosen uniformly from $\{0,1\}^\gamma$, runs as follows. Choose r' uniformly at random from $\{0,1\}^\gamma$ and output r' as the solution of the one-way experiment. Since r and r' are independent, we have

[4] Here, we purposely remain each $\Pr[\textcircled{i}]$ as it is for the later calculation of $\mathsf{Adv}_{\Pi,\mathcal{S}}^{\text{NM-sID-CPA}}$.

$\mathsf{Adv}^{\mathrm{OW}}_{f,\mathcal{A}'} = \Pr[f(r) = f(r')] \geq \Pr[r = r'] = \frac{1}{2^\gamma}$. According to the definition of ϵ_{ow}, we have $\mathsf{Adv}^{\mathrm{OW}}_{f,\mathcal{A}} \leq \epsilon_{ow}$ for any adversary $\mathcal{A}$ (including $\mathcal{A}'$). Thus, we have $\frac{1}{2^\gamma} \leq \epsilon_{ow}$. □

Now, focusing on the second and the fourth terms of the right side member of the above inequality, we can define $\alpha = \frac{1}{q}$, which will cancel out all the terms regarding P_k. Using this α, we have

$$\mathsf{Adv}^{\text{NM-sID-CPA}}_{\mathit{\Pi},\mathcal{S}} \geq \frac{1}{2q}(\epsilon_{cca} + P_v(\frac{1}{2} - \epsilon_{cca})) - \epsilon_{ow} \geq \frac{1}{2q}\epsilon_{cca} - \epsilon_{ow},$$

where the right side inequality is due to the definition of the IND-CCA advantage and the fact that $P_v \geq 0$.

Consequently, assuming $\mathcal{A}$ has advantage ϵ_{cca} in breaking IND-CCA security of the proposed PKE scheme $\mathit{\Pi}'$ and f is a (t, ϵ_{ow})-OWF, $\mathcal{S}$ can break NM-sID-CPA security of the underlying IBE scheme $\mathit{\Pi}$ with the above advantage, using above α.

To complete the proof, we prove the lemmas in order.

Proof of Lemma 1: In Case 1, an event $[\textcircled{1}] = [b_A = b_S \wedge b_S = b_C \wedge \overline{\mathsf{Valid}}]$ occurs. In this case, we have $M^* = (m_{b_S}||r^*)$ and $M^{\bar{*}} = (m'||r')$. First, we estimate $\Pr[\mathsf{R}^*|\textcircled{1}]$.

$$\Pr[\mathsf{R}^*|\textcircled{1}] \geq \Pr[\mathsf{R}^* \wedge b_\alpha = 1|\textcircled{1}] = \Pr[b_\alpha = 1] \cdot \Pr[\mathsf{R}^*|\textcircled{1} \wedge b_\alpha = 1] = \alpha,$$

where we used the following.

Claim 2. $\Pr[\mathsf{R}^*|\textcircled{1} \wedge b_\alpha = 1] = 1$.

Proof of Claim 2. $[b_S = b_C]$ implies that the plaintext of $\mathcal{S}$'s challenge ciphertext y^* is $M^* = (m_{b_S}||r^*)$, and thus γ-least significant bits of M^* is r^*. And when $[b_A = b_S \wedge b_\alpha = 1]$ occurs, $\mathcal{S}$ outputs y' such that γ-least significant bits of its plaintext is also r^*. Thus, $\mathcal{S}$ always outputs a ciphertext y' such that R^* occurs and we have $\Pr[\mathsf{R}^*|\textcircled{1}\wedge b_\alpha = 1] = 1$. □

Next, we estimate $\Pr[\mathsf{R}^{\bar{*}}|\textcircled{1}]$.

Claim 3. $\Pr[\mathsf{R}^{\bar{*}}|\textcircled{1}] = \frac{1}{2^\gamma}$.

Proof of Claim 3. Recall that γ-$\mathsf{LSB}(M^{\bar{*}}) = r'$. Recall also that γ-least significant bits of the plaintext of y' is either r^* or r'', depending on the value b_α. Since r', r^* and r'' are independently chosen by $\mathcal{S}$, we have

$$\Pr[\mathsf{R}^{\bar{*}}|\textcircled{1}] = \Pr[r' = r^* \wedge b_\alpha = 1] + \Pr[r' = r'' \wedge b_\alpha = 0] = \frac{1}{2^\gamma} \cdot \alpha + \frac{1}{2^\gamma} \cdot (1 - \alpha) = \frac{1}{2^\gamma}.$$

□

Finally, we estimate $\Pr[\textcircled{1}]$. We have

$$\begin{aligned}\Pr[\textcircled{1}] &= \Pr[b_A = b_S \wedge b_S = b_C \wedge \overline{\mathsf{Valid}}] \\ &= \frac{1}{2}\Pr[b_A = b_S | b_S = b_C \wedge \overline{\mathsf{Valid}}] \cdot \Pr[\overline{\mathsf{Valid}} | b_S = b_C] = \frac{1}{2}(\frac{1}{2} + \epsilon_{cca})(1 - P_v),\end{aligned}$$

where we used the following.

Claim 4. $\Pr[b_A = b_S | b_S = b_C \wedge \overline{\mathsf{Valid}}] = \frac{1}{2} + \epsilon_{cca}$.

Proof of Claim 4. If both Valid and $[b_S \neq b_C]$ do not occur, then $\mathcal{S}$ perfectly simulates the IND-CCA game for $\mathcal{A}$, and the view of $\mathcal{A}$ is identical to that when attacking Π'. Thus in this case $\mathcal{A}$ wins his IND-CCA game perfectly simulated by $\mathcal{S}$ with ordinary probability $\frac{1}{2} + \epsilon_{cca}$. □

Consequently, $\mathcal{S}$'s advantage in Case 1 is estimated as: $\mathsf{Adv}_1 = (\Pr[\mathsf{R}^* | \textcircled{1}] - \Pr[\mathsf{R}^{\bar{*}} | \textcircled{1}]) \cdot \Pr[\textcircled{1}] \geq \frac{1}{2}\alpha(\frac{1}{2} + \epsilon_{cca})(1 - P_v) - \frac{1}{2^\gamma}\Pr[\textcircled{1}]$, which completes the proof of Lemma 1. □

Proof of Lemma 2: In Case 2, an event $[\textcircled{2}] = [b_A = b_S \wedge b_S = b_C \wedge \mathsf{Valid}]$ occurs. In this case, we have $M^* = (m_{b_S} || r^*)$ and $M^{\bar{*}} = (m' || r')$. With the same discussion in the proof of Lemma 1, we have $\Pr[\mathsf{R}^* | \textcircled{2}] \geq \alpha$ and $\Pr[\mathsf{R}^{\bar{*}} | \textcircled{2}] = \frac{1}{2^\gamma}$. As for $\Pr[\textcircled{2}]$, we have $\Pr[\textcircled{2}] = \Pr[b_A = b_S \wedge b_S = b_C \wedge \mathsf{Valid}] = \frac{1}{2}\Pr[b_A = b_S | b_S = b_C \wedge \mathsf{Valid}] \cdot \Pr[\mathsf{Valid} | b_S = b_C] = \frac{1}{2}P_k P_v$.

Consequently, $\mathcal{S}$'s advantage in Case 2 is estimated as: $\mathsf{Adv}_2 = (\Pr[\mathsf{R}^* | \textcircled{2}] - \Pr[\mathsf{R}^{\bar{*}} | \textcircled{2}]) \cdot \Pr[\textcircled{2}] \geq \frac{1}{2}\alpha P_k P_v - \frac{1}{2^\gamma}\Pr[\textcircled{2}]$, which completes the proof of Lemma 2. □

Proof of Lemma 3: In Case 3, an event $[\textcircled{3}] = [b_A = b_S \wedge b_S \neq b_C]$ occurs. In this case, we have $M^* = (m' || r')$ and $M^{\bar{*}} = (m_{b_S} || r^*)$. We first estimate $\Pr[\mathsf{R}^{\bar{*}} | \textcircled{3}]$.

$$\begin{aligned}\Pr[\mathsf{R}^{\bar{*}} | \textcircled{3}] &= \Pr[\mathsf{R}^{\bar{*}} \wedge b_\alpha = 1 | \textcircled{3}] + \Pr[\mathsf{R}^{\bar{*}} \wedge b_\alpha = 0 | \textcircled{3}] \\ &= \Pr[b_\alpha = 1] \cdot \Pr[\mathsf{R}^{\bar{*}} | \textcircled{3} \wedge b_\alpha = 1] + \Pr[b_\alpha = 0] \cdot \Pr[\mathsf{R}^{\bar{*}} | \textcircled{3} \wedge b_\alpha = 0] \\ &\leq \alpha + \Pr[\mathsf{R}^{\bar{*}} | \textcircled{3} \wedge b_\alpha = 0] \leq \alpha + \frac{1}{2^\gamma},\end{aligned}$$

where we used the following.

Claim 5. $\Pr[\mathsf{R}^{\bar{*}} | \textcircled{3} \wedge b_\alpha = 0] \leq \frac{1}{2^\gamma}$.

Proof of Claim 5. When $[b_\alpha = 0]$ occurs, $\mathcal{S}$ chooses r'' uniformly from $\{0,1\}^\gamma$, independently of r^*, generates y' such that γ-least significant bits of the plaintext is r'', and outputs it with the relation R in Output phase. Therefore, the probability that $\mathsf{R}^{\bar{*}}$ occurs in this scenario is identical to the probability that $r'' = r^*$ occurs and is at most $\frac{1}{2^\gamma}$. □

As for $\Pr[\textcircled{3}]$, we have $\Pr[\textcircled{3}] = \Pr[b_A = b_S \wedge b_S \neq b_C] = \frac{1}{2}\Pr[b_A = b_S | b_S \neq b_C] = \frac{1}{4}$, where we used the following.

Claim 6. $\Pr[b_A = b_S | b_S \neq b_C] = \frac{1}{2}$.

Proof of Claim 6. When $[b_S \neq b_C]$ occurs, since the challenge ciphertext χ^* given to $\mathcal{A}$ is not an encryption of either m_0 nor m_1 that are submitted by $\mathcal{A}$, $\mathcal{S}$'s simulation for $\mathcal{A}$ becomes imperfect. Thus, $\mathcal{A}$'s behavior may become unknown to $\mathcal{S}$ after Challenge phase. However, since b_S is information-theoretically hidden to $\mathcal{A}$, the probability that $[b_A = b_S]$ occurs is exactly $\frac{1}{2}$. □

Consequently, $\mathcal{S}$'s advantage in Case 3 is estimated as: $\mathsf{Adv}_3 \geq -\Pr[\mathsf{R}^{\bar{*}}|\textcircled{3}] \cdot \Pr[\textcircled{3}] \geq -\frac{1}{4}\alpha - \frac{1}{2^\gamma}\Pr[\textcircled{3}]$, which completes the proof of Lemma 3. □

Proof of Lemma 4: In Case 4, an event $[\textcircled{4}] = [b_A \neq b_S \wedge b_S = b_C \wedge \overline{\mathsf{Valid}}]$ occurs. In this case, we have $M^* = (m_{b_S}||r^*)$ and $M^{\bar{*}} = (m'||r')$. According to our construction of $\mathcal{S}$, $[b_A \neq b_S]$ implies that $\mathcal{S}$ chooses y' uniformly from $\mathcal{A}$'s decryption queries. We estimate $\Pr[\mathsf{R}^{\bar{*}}|\textcircled{4}]$ in the following.

Claim 7. $\Pr[\mathsf{R}^{\bar{*}}|\textcircled{4}] \leq \epsilon_{ow}$.

Proof of Claim 7. Since in this case r' is information-theoretically hidden to $\mathcal{A}$, the value $f(r')$ is also information-theoretically hidden to $\mathcal{A}$. Thus, probability that $\mathcal{A}$, without seeing $f(r')$, happens to issue decryption queries such that the image with f of γ-least significant bits of the plaintext becomes identical to $f(r')$ is at most ϵ_{ow}. □

Consequently, $\mathcal{S}$'s advantage in Case 4 is estimated as: $\mathsf{Adv}_4 \geq -\Pr[\mathsf{R}^{\bar{*}}|\textcircled{4}] \cdot \Pr[\textcircled{4}] \geq -\epsilon_{ow}\Pr[\textcircled{4}]$, which completes the proof of Lemma 4. □

Proof of Lemma 5: In Case 5, an event $[\textcircled{5}] = [b_A \neq b_S \wedge b_S = b_C \wedge \mathsf{Valid}]$ occurs. In this case, we have $M^* = (m_{b_S}||r^*)$ and $M^{\bar{*}} = (m'||r')$. Due to $[b_A \neq b_S]$, $\mathcal{S}$ chooses one ciphertext uniformly from $\mathcal{A}$'s decryption queries and uses it as its final output in Output phase of the NM-sID-CPA game. First, we estimate $\Pr[\mathsf{R}^*|\textcircled{5}]$. We prove the following claim.

Claim 8. $\Pr[\mathsf{R}^*|\textcircled{5}] \geq \frac{1}{q}$.

Proof of Claim 8. Since $\mathcal{S}$ cannot correctly respond to the decryption query that causes Valid, $\mathcal{S}$'s simulation for $\mathcal{A}$ becomes imperfect and $\mathcal{A}$'s behavior may become unknown to $\mathcal{S}$ after such query. However, when Valid occurs, $\mathcal{A}$ must have issued at least one query of the form $\langle \mathsf{ID}^*, y_A \rangle$ such that the plaintext of y_A forms $M_A = (m_A||r_A)$ and $f(r_A) = f(r^*) = \mathsf{ID}^*$ holds. Thus, if $\mathcal{S}$ chooses such y_A for y' from $\mathcal{A}$'s decryption queries, R^* occurs. Since the number of $\mathcal{A}$'s decryption queries is at most q, the above probability is at least $\frac{1}{q}$. □

With the same discussion in the proof of Lemma 4, we have $\Pr[\mathsf{R}^{\bar{*}}|\textcircled{5}] \leq \epsilon_{ow}$. We also have $\Pr[\textcircled{5}] = \Pr[b_A \neq b_S \wedge b_S = b_C \wedge \mathsf{Valid}] = \frac{1}{2}(1 - P_k)P_v$.

Consequently, $\mathcal{S}$'s advantage in Case 5 is estimated as: $\mathsf{Adv}_5 = (\Pr[\mathsf{R}^*|\textcircled{5}] - \Pr[\mathsf{R}^{\bar{*}}|\textcircled{5}]) \cdot \Pr[\textcircled{5}] \geq \frac{1}{2q}(1 - P_k)P_v - \epsilon_{ow}\Pr[\textcircled{5}]$, which completes the proof of Lemma 5. □

Proof of Lemma 6: In Case 6, an event $[\textcircled{6}] = [b_A \neq b_S \wedge b_S \neq b_C]$ occurs. In this case, we have $M^* = (m'||r')$ and $M^{\bar{*}} = (m_{b_S}||r^*)$. Again, due to $[b_A \neq b_S]$, $\mathcal{S}$ chooses one ciphertext from $\mathcal{A}$'s decryption query. We estimate $\Pr[\mathsf{R}^{\bar{*}}|\textcircled{6}]$.

Claim 9. $\Pr[\mathsf{R}^{\bar{*}}|\textcircled{6}] \leq \epsilon_{ow}$.

Proof of Claim 9. Recall that we defined that, in the case $[b_A \neq b_S]$, the relation $R(a, b)$ tests whether $f(\gamma\text{-}\mathsf{LSB}(a)) = f(\gamma\text{-}\mathsf{LSB}(b))$ holds. Thus, the event $\mathsf{R}^{\bar{*}}$ in this case consists of the following two events: (1) $\mathcal{A}$ issues at least one decryption query $\langle \mathsf{ID}, y\rangle$ which satisfies the conditions $\mathsf{IBE.Dec}(d_{\mathsf{ID}^*}, y) = (m||r) \neq \bot$ and $f(r^*) = f(r)$, where $f(r^*) = \mathsf{ID}^*$ is the first component of the challenge ciphertext given to $\mathcal{A}$, and (2) $\mathcal{S}$ chooses such a query. Note that the first event above is exactly the same event as Valid. For notational convenience, we denote by Choice the second event above. Moreover, according to our construction of $\mathcal{S}$, in the case $[b_S \neq b_C]$, the challenge ciphertext given to $\mathcal{A}$ is a "garbage" ciphertext which forms $\langle f(r^*), \mathsf{IBE.Enc}(\mathsf{prm}, f(r^*), (m'||r'))\rangle$ where r^* and r' are chosen independently and uniformly from $\{0,1\}^\gamma$ and m' is also chosen randomly. Also for notational convenience, we denote by $\mathsf{RandomCT}$ an event that $\mathcal{A}$ is given a garbage challenge ciphertext of this form. Suppose that, as $\mathcal{S}$'s final output, one ciphertext $\langle \mathsf{ID}, y\rangle$ is chosen from $\mathcal{A}$'s queries. Using these notations, we have

$$\begin{aligned}
&\Pr[\mathsf{R}^{\bar{*}}|\textcircled{6}]\\
&=\Pr[\mathsf{Choice} \wedge \mathsf{IBE.Dec}(d_{\mathsf{ID}^*}, y) = (m||r) \neq \bot \wedge R((m_{b_S}||r^*), (m||r))|b_A \neq b_S \wedge b_S \neq b_C]\\
&=\Pr[\mathsf{Choice} \wedge \mathsf{IBE.Dec}(d_{\mathsf{ID}^*}, y) = (m||r) \neq \bot \wedge f(r^*) = f(r)|\mathsf{RandomCT}]\\
&=\Pr[\mathsf{Choice} \wedge \mathsf{Valid}|\mathsf{RandomCT}] \leq \Pr[\mathsf{Valid}|\mathsf{RandomCT}].
\end{aligned}$$

Thus, all we have to do is to show $\Pr[\mathsf{Valid}|\mathsf{RandomCT}] \leq \epsilon_{ow}$. Now, towards a contradiction, we assume $\Pr[\mathsf{Valid}|\mathsf{RandomCT}] > \epsilon_{ow}$. We construct another simulator $\mathcal{S}'$ which, using $\mathcal{A}$, breaks one-wayness of f. The description of $\mathcal{S}'$ is as follows.

Given f and $f(r^*)$ (where r^* is uniformly chosen from $\{0,1\}^\gamma$ and unknown to $\mathcal{S}'$), $\mathcal{S}'$ first sets $\mathsf{ID}^* = f(r^*)$, then runs $(\mathsf{msk}, \mathsf{prm}) \leftarrow \mathsf{IBE.Setup}$ and $d_{\mathsf{ID}^*} \leftarrow \mathsf{IBE.Ext}(\mathsf{msk}, \mathsf{ID}^*)$. It gives $PK = (\mathsf{prm}, f)$ to $\mathcal{A}$. Since $\mathcal{S}'$ possesses $SK = \mathsf{msk}$, it can perfectly respond to the decryption queries. When $\mathcal{A}$ submits two plaintexts as a challenge, $\mathcal{S}'$ ignores it and generates the challenge ciphertext $\chi^* = \langle \mathsf{ID}^*, y^*\rangle = \langle f(r^*), \mathsf{IBE.Enc}(\mathsf{prm}, f(r^*), (m'||r'))\rangle$ where r' is uniformly chosen from $\{0,1\}^\gamma$ and m' is also chosen randomly, and gives χ^* to $\mathcal{A}$. After $\mathcal{A}$ outputs a guess bit, from $\mathcal{A}$'s decryption queries $\mathcal{S}'$ finds one ciphertext $\langle \mathsf{ID}, y\rangle$ whose second component y satisfies $\mathsf{IBE.Dec}(d_{\mathsf{ID}^*}, y) = (m||r) \neq \bot$ and $f(r) = f(r^*)$, and outputs such r (if no such query is found then $\mathcal{S}'$ simply aborts).

It is easy to see that $\mathcal{S}'$ perfectly simulates the scenario $\mathsf{RandomCT}$ for $\mathcal{A}$. Moreover, whenever Valid occurs, $\mathcal{S}'$ can find a preimage of $f(r^*)$ and thus breaks the one-wayness of f. Therefore, $\mathsf{Adv}^{\mathrm{OW}}_{f,\mathcal{S}'} = \Pr[\mathsf{Valid}|\mathsf{RandomCT}] > \epsilon_{ow}$, which contradicts that f is a OWF, and thus we must have $\Pr[\mathsf{Valid}|\mathsf{RandomCT}] \leq \epsilon_{ow}$. This completes the proof of Claim 9. □

Consequently, $\mathcal{S}$'s advantage in Case 6 is estimated as: $\mathsf{Adv}_6 \geq -\Pr[\mathsf{R}^{\bar{*}}|\textcircled{6}] \cdot \Pr[\textcircled{6}] \geq -\epsilon_{ow} \Pr[\textcircled{6}]$, which completes the proof of Lemma 6. □

Above completes the proof of Theorem 1. □

3.3 Extensions

As is the same with the previous generic IBE-to-PKE transformations, our transformation can be applied to TBE schemes if we appropriately define non-malleability for TBE schemes. (We discuss this in the full version of our paper.)

Moreover, if we consider non-malleability for HIBE schemes in the same way as in Section 2.2, then our transformation can be used to obtain adaptive (resp., selective) identity CCA-secure t-level HIBE from $(t+1)$-level HIBE that is non-malleable against adaptive (resp., selective) identity, CPA.

4 Comparison

Table 1 compares our transformation in Section 3 and other generic IBE-to-PKE transformations including the CHK transformation (CHK) [15], the BK transformation (BK) [11], and the BK transformation where the encapsulation scheme[5] used in the BK transformation is instantiated by using a target collision resistant hash function (TCRHF) [36,5] and a pairwise-independent hash function (PIHF) (BK*).[6] In Table 1, the column "IBE" denotes the security requirement for the underlying IBE schemes ("-sID-CPA" is omitted), the column "Overhead by Transformation" denotes how much the ciphertext size increases from that of the underlying IBE scheme (typical sizes for 128-bit security are given as numerical examples), the column "Required Size for $\mathcal{M}_{IBE}$" denotes how much size is necessary for the plaintext space of the underlying IBE scheme, the column "Reduction" denotes the ratios of the advantage of breaking the transformed PKE schemes and that of the underlying IBE schemes, and the column "Publicly Verifiable?" denotes whether we can check the validity of ciphertexts publicly (without any secret keys) assuming that of the underlying IBE scheme can be also checked publicly.

Ciphertext Overhead by Transformations. Here, we mainly compare ours with BK* scheme, because the ciphertext overhead of the original CHK and BK transformations depend on how we construct their additional building blocks.

In our transformation, the size overhead from the ciphertext of the underlying IBE scheme is caused by a randomness r and its image $f(r)$ with a OWF f. If we require 128-bit security, we can set each to be 128-bit, and thus we have 256-bit overhead in total. In BK*, on the other hand, overhead is caused by a TCRHF (TCR), a MAC, and a large randomness r' (because of the use of the Leftover Hash Lemma [27] with the use of a PIHF in order to get an almost uniformly distributed value for a MAC key). Because of r', though size of $\mathsf{TCR}(r')$ and the tag from MAC can be 128-bit, we need at least 448-bit for the randomness r', and the overhead in total needs to be 704-bit.

Observation: IND vs. NM. As we can see, there exists a trade-off between assumptions on security of the underlying IBE schemes and ciphertext overhead.

[5] An *encapsulation* scheme is a special kind of commitment scheme. See [11] for definitions.

[6] This construction of the encapsulation scheme is introduced in [11].

Table 1. Comparison among Generic IBE-to-PKE Transformations

	IBE	Overhead by Transformation † (Numerical Example (bit))	Required Size for $\mathcal{M}_{IBE}$	Reduction	Publicly Verifiable?
CHK	IND	$\lvert\mathsf{vk}\rvert + \lvert\mathsf{sig}\rvert$ (—)	$\lvert m_{PKE}\rvert$	tight	yes
BK	IND	$\lvert\mathsf{com}\rvert + \lvert\mathsf{dec}\rvert + \lvert\mathsf{MAC}\rvert$ (—)	$\lvert m_{PKE}\rvert + \lvert\mathsf{dec}\rvert$	tight	—
BK*	IND	$\lvert\mathsf{TCR}(r')\rvert + \lvert r'\rvert + \lvert\mathsf{MAC}\rvert$ (704)	$\lvert m_{PKE}\rvert + \lvert r'\rvert$	tight	—
Ours	NM	$\lvert f(r)\rvert + \lvert r\rvert$ (256)	$\lvert m_{PKE}\rvert + \lvert r\rvert$	$1/2q$	—

† $|\mathsf{vk}|$ and $|\mathsf{sig}|$ denote the size of the verification key and the signature of the one-time signature in the CHK transformation. $|\mathsf{com}|$ and $|\mathsf{dec}|$ denote the size of the commitment and the decommitment of the encapsulation scheme, and $|\mathsf{MAC}|$ denotes the size of the tag of MAC in the BK transformation. $|m_{PKE}|$ is a plaintext size of a transformed PKE scheme.

Roughly speaking, if we see the OWF in our transformation as a hash function, then our transformation is obtained by getting rid of the PIHF and the MAC from BK*. And the lost power is supplied by the property of non-malleability of the underlying IBE scheme. But it is not easy with a brief consideration to come up with an efficient NM-sID-CPA secure IBE scheme from a combination of an IND-sID-CPA secure IBE scheme, a PIHF, and a MAC. Thus, this relation between ours and BK* could be seen as a concrete (but qualitative) evidence that shows the (huge) gap between what IND achieves and what NM achieves (at least, for selective identity, chosen plaintext attacks for IBE schemes).

References

1. Attrapadung, N., Cui, Y., Galindo, D., Hanaoka, G., Hasuo, I., Imai, H., Matsuura, K., Yang, P., Zhang, R.: Relations among notions of security for identity based encryption schemes. In: Correa, J.R., Hevia, A., Kiwi, M. (eds.) LATIN 2006. LNCS, vol. 3887, pp. 130–141. Springer, Heidelberg (2006)
2. 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)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proc. of CCS 1993, pp. 62–73. ACM, New York (1993)
4. 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)
5. Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making UOWHFs practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 320–335. Springer, Heidelberg (1997)
6. Bellare, M., Sahai, A.: Non-malleable encryption: Equivalence between two notions, and indistinguishability-based characterization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 519–536. Springer, Heidelberg (1999)
7. Bellare, M., Sahai, A.: Non-malleable encryption: Equivalence between two notions, and indistinguishability-based characterization (2006); full version of [6], `eprint.iacr.org/2006/228`

8. 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)
9. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
10. 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)
11. Boneh, D., Katz, J.: Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
12. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: Proc. of CCS 2005, pp. 320–329. ACM Press, New York (2005)
13. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. In: Proc. of STOC 1998, pp. 209–218. ACM, New York (1998)
14. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
15. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
16. Chatterjee, S., Sarkar, P.: HIBE with short public prarameters without random oracle. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 145–160. Springer, Heidelberg (2006)
17. Chatterjee, S., Sarkar, P.: Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 424–440. Springer, Heidelberg (2006)
18. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, p. 360. Springer, Heidelberg (2001)
19. 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)
20. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
21. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. In: Proc. of STOC 1991, pp. 542–552. ACM Press, New York (1991)
22. 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)
23. Galindo, D.: A separation between selective and full-identity security notions for identity-based encryption. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3982, pp. 318–326. Springer, Heidelberg (2006)
24. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)

25. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
26. Goldwasser, S., Micali, S.: Probabilistic encryption. J. of Computer and System Sciences 28(2), 270–299 (1984)
27. Håstad, J., Impagliazzo, R., Levin, L., Luby, M.: Construction of a pseudorandom generator from any one-way function. SIAM J. Computing 28(4), 1364–1396 (1999)
28. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
29. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)
30. Kiltz, E., Galindo, D.: Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 336–347. Springer, Heidelberg (2006)
31. Kiltz, E., Vahlis, Y.: CCA2 secure IBE: Standard model efficiency through authenticased symmetric encryption. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 221–238. Springer, Heidelberg (2008)
32. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
33. Lindell, Y.: A simpler construction of CCA2-secure public-key encryption under general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 241–254. Springer, Heidelberg (2003)
34. MacKenzie, P., Reiter, M.K., Yang, K.: Alternatives to non-malleability: Definitions, constructions and applications. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 171–190. Springer, Heidelberg (2004)
35. Naccache, D.: Secure and practical identity-based encryption (2005), `eprint.iacr.org/2005/369`
36. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: Proc. of STOC 1989, pp. 33–43. ACM, New York (1989)
37. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proc. of STOC 1990, pp. 427–437. ACM, New York (1990)
38. Okamoto, T., Pointcheval, D.: REACT: Rapid enhanced-security asymmetric cryptosystem transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–174. Springer, Heidelberg (2001)
39. Pass, R., Shelat, A., Vaikuntanathan, V.: Construction of a Non-malleable Encryption Scheme from Any Semantically Secure One. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 271–289. Springer, Heidelberg (2006)
40. Pass, R., Shelat, A., Vaikuntanathan, V.: Relations Among Notions of Non-malleability for Encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 519–535. Springer, Heidelberg (2007)
41. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Proc. of STOC 2008, pp. 187–196. ACM, New York (2008)
42. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
43. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: Proc. of FOCS 1999, pp. 543–553. IEEE Computer Society Press, Los Alamitos (1999)

44. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing over elliptic curve (in japanese). In: Proc. of SCIS 2001 (2001)
45. Sarkar, P., Chatterjee, S.: Construction of a hybrid HIBE protocol secure against adaptive attacks (without random oracle). In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 51–67. Springer, Heidelberg (2007)
46. 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)
47. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Distributed Attribute-Based Encryption

Sascha Müller, Stefan Katzenbeisser, and Claudia Eckert

Technische Universität Darmstadt
Hochschulstr. 10
D – 64289 Darmstadt
{mueller,eckert}@sec.informatik.tu-darmstadt.de,
katzenbeisser@seceng.informatik.tu-darmstadt.de

Abstract. Ciphertext-Policy Attribute-Based Encryption (CP-ABE) allows to encrypt data under an access policy, specified as a logical combination of attributes. Such ciphertexts can be decrypted by anyone with a set of attributes that fits the policy. In this paper, we introduce the concept of Distributed Attribute-Based Encryption (DABE), where an arbitrary number of parties can be present to maintain attributes and their corresponding secret keys. This is in stark contrast to the classic CP-ABE schemes, where all secret keys are distributed by one central trusted party. We provide the first construction of a DABE scheme; the construction is very efficient, as it requires only a constant number of pairing operations during encryption and decryption.

1 Introduction

Emerging ubiquitous computing environments need flexible access control mechanisms. With a large and dynamic set of users, access rules for objects cannot easily be based on identities, and the conditions under which access to an object is granted need to take into account information like the context and the history of a subject. Due to these shortcomings of traditional access control mechanisms, cryptographically enforced access control receives increasing attention.

One of the most interesting approaches is Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [1]. In this scheme, users possess sets of attributes (and corresponding secret attribute keys) that describe certain properties. Ciphertexts are encrypted according to an access control policy, formulated as a Boolean formula over the attributes. The construction assures that only users whose attributes satisfy the access control policy are able to decrypt the ciphertext with their secret attribute keys. The construction is required to satisfy a *collusion-resistance* property: It must be impossible for several users to pool their attribute keys such that they are able to decrypt a ciphertext which they would not be able to decrypt individually.

Common to most previous ABE schemes is the existence of a central trusted authority (*master*) that knows a secret master key and distributes secret attribute keys to eligible users. However, for many attribute-based scenarios, it is much more natural to support multiple authorities [8,7]. The limitation to a

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 20–36, 2009.

```
http://db.mycompany.org : isAdmin OR
http://db.mycompany.org : hasFullAccess OR
( http://www.openid.org : is18OrOlder AND
 ( http://www.contprov1.com : article1234.hasPaidFor OR
   http://www.contprov2.com : article4325.hasPaidFor OR
   http://www.contprov3.com : articleABC.hasPurchased ) )
```

Fig. 1. An example policy

single central authority for attribute generation is neither realistic nor desirable in applications where no single entity has the authority to grant secret keys for arbitrary attributes.

We can, for exemplary purposes, illustrate one such scenario as follows. Consider a company that hosts DRM protected media files. Users can purchase licenses from various content providers that issue usage licenses which contain the keys required to decrypt the protected files. Let us assume that three such content providers are `contprov1.com`, `contprov2.com`, and `contprov3.com`. The usage license (see Figure 1) can be expressed as a Boolean formula over attributes. Here, attributes consist of an URL that specifies a party who has authority over an attribute and an identifier describing the attribute itself, both represented as strings and concatenated with a single colon character as separator. The intuition behind this sample policy is that the protected file should only be decrypted by someone who either is an administrator of the company database `db.mycompany.org`, has the rights to download all files, or is at least 18 years old (which is established by an identification service `www.openid.org`) and has purchased licenses from at least one of the given content providers. Note that the same media file might be identified by different product codes in different providers' databases.

It is difficult to use this policy in a standard CP-ABE scheme, since there is no central authority who maintains and controls all attributes; in the above example, `www.contprov1.com` is solely responsible for maintaining the attribute `article1234.hasPaidFor`, while `db.mycompany.org` has authority over the attribute `isAdmin`. While it is possible that a third party is set up to which the maintenance of all attributes is delegated, this solution obviously does not scale. In addition, this solution is problematic if the entities mutually distrust each other.

1.1 Distributed ABE

We propose Distributed Attribute-Based Encryption (DABE) to mitigate this problem. DABE allows an arbitrary number of authorities to independently maintain attributes. There are three different types of entites in a DABE scheme: a master, attribute authorities and users.

The *master* is responsible for the distribution of secret user keys. However, in contrast to standard CP-ABE schemes, this party is *not* involved in the creation of secret attribute keys; the latter task can independently be performed by the attribute authorities.

Attribute authorities are responsible to verify whether a user is eligible of a specific attribute; in this case they distribute a secret attribute key to the user. (Note that determining the users' eligibility is application-dependent and thus beyond the scope of this work.) In our scheme every attribute is associated with a single attribute authority, but each attribute authority can be responsible for an arbitrary number of attributes. Every attribute authority has full control over the structure and semantics of its attributes. An attribute authority generates a *public attribute key* for each attribute it maintains; this public key is available to every participant. Eligible users receive a personalized secret attribute key over an authenticated and trusted channel. This secret key, which is personalized to prevent collusion attacks, is required to decrypt a ciphertext.

Users can encrypt and decrypt messages. To encrypt a message, a user first formulates his access policy in the form of a Boolean formula over some attributes, which in our construction is assumed to be in Disjunctive Normal Form (DNF). The party finally uses the public keys corresponding to the attributes occurring in the policy to encrypt. In DNF, all negations are atomic, so attribute authorities should be able to issue negative attributes as well in order to make use of the full expressive power of DNF formulas.

To decrypt a ciphertext, a user needs at least access to some set of attributes (and their associated secret keys) which satisfies the access policy. If he does not already possess these keys, he may query the attribute authorities for the secret keys corresponding to the attributes he is eligible of.

To illustrate the use of a DABE scheme, we return to the above mentioned example of the protection of media files. Figure 2 shows the policy of Figure 1 in DNF. The policy consists of five conjunctions over different sets of attributes. A user needs all secret attribute keys of at least one of the conjunctive terms to be able to decrypt a ciphertext that was encrypted with this access policy.

A user who downloads the ciphertext analyzes the policy and tests if he has a sufficient set of attributes to decrypt. The user may contact attribute authorities for secret attribute keys he does not already have in his possession but he is eligible of. For instance, he may query `www.openid.org` for a secret attribute key corresponding to `is18OrOlder` and `contprov3.com` for a secret attribute key corresponding to the attribute `articleABC.hasPurchased`. In this case he is able to satisfy the last conjunction. It may be necessary for him to perform additional steps if he is not yet eligible for an attribute. For example, he might decide to buy the article `articleABC` from `contprov3.com` to get the respective attribute.

Note that every attribute authority independently decides on the structure and semantics of its attributes. For instance, the authority `db.mycompany.org` offers the attribute `isAdmin`. The meaning of this attribute and the semantics (i.e., the decision who is eligible of it) is entirely up to `db.mycompany.org`. Whoever includes the attribute in an access policy needs to trust the respective authority to correctly determine eligibility.

Note that a DABE scheme must be collusion-resistant: if a user u has a friend v who possesses an attribute that u does not have, it is not possible for u to

```
   http://db.mycompany.org : isAdmin
OR
   http://db.mycompany.org : hasFullAccess
OR
   ( http://www.openid.org : is18OrOlder AND
     http://www.contprov1.com : article1234.hasPaidFor )
OR
   ( http://www.openid.org : is18OrOlder AND
     http://www.contprov2.com : article4325.hasPaidFor)
OR
   ( http://www.openid.org : is18OrOlder AND
     http://www.contprov3.com : articleABC.hasPurchased )
```

Fig. 2. Policy of Figure 1 in DNF

use the corresponding secret attribute key of v. Neither can u give any of his attribute keys to v. All secret attribute keys are bound to their owner, making them unusable with keys issued for other users.

1.2 Our Contribution

In this paper we introduce the concept of Distributed Attribute-Based Encryption (DABE), i.e., a fully distributed version of CP-ABE, where multiple attribute authorities may be present and distribute secret attribute keys. Furthermore, we give the first construction of a DABE scheme, which supports policies written in DNF; the ciphertexts grow linearly with the number of conjunctive terms in the policy. Our scheme is very simple and efficient, demonstrating the practical viability of DABE. We furthermore provide a proof of security in the generic group model, introduced by [2]; even though this proof is weaker than the proofs of some more recent CP-ABE schemes [3,4,5], our scheme is much more efficient, requiring only $O(1)$ pairing operations during encryption and decryption.

The remainder of this document is structured as follows: In Section 2 we discuss related work. Section 3 contains a description of DABE as well as a formal definition of the required security property. Our construction is detailed in Section 4. We discuss its security and performance in Section 5. Finally we conclude in Section 6. A detailed security proof in the generic bilinear group model is given in the appendix.

2 Related Work

Attribute-Based Encryption was first proposed by Goyal et al. [6] in the form of *key-policy* attribute-based encryption (KP-ABE), based on the work of Sahai and Waters [7]. In KP-ABE, users are associated with access policies and ciphertexts are encrypted with sets of attributes. The access policies describe which ciphertexts users can decrypt.

The first CP-ABE scheme was presented by Bethencourt, Sahai and Waters [1], followed by some cryptographically stronger CP-ABE constructions that allowed reductions to the Decisional Bilinear Diffie Hellman Problem [5,4], but imposed restrictions that the original CP-ABE does not have. Recently, Waters proposed three CP-ABE schemes that are as expressive as [1], rather efficient and provably secure under strong cryptographic assumptions [3].

There is only one attempt at multi-authority CP-ABE, proposed by Chase [8] as an extension of her multi-authority threshold ABE construction. This extension is rather limited. The policy is written as a set of threshold gates, which are connected by another outer threshold gate. The threshold of the outer gate is fixed. (However, one could run several parallel instances to support different thresholds.) Each of the inner threshold gates is controlled by one of the authorities, and contains only attributes of that authority. The threshold of each inner gate is fixed, even though dummy attributes can be used to support different thresholds, as described in [7]. If the policy can only be formulated in a way where some attributes occur in more than one of the inner threshold gates, these attributes must be copied between the respective authorities, so in this case the involved authorities need to mutually trust each other.

Another restriction of Chase's scheme is that all authorities are managed centrally by a trusted authority. Whenever a new authority is added, the global system key changes and has to be propagated to all users who want to use attributes of the authority. This includes encryptors who want to use attributes from that authority in the policy.

Techniques similar to CP-ABE were proposed for many applications like Attribute-Based Access Control (ABAC, used in SOA) [9], Property-Based Broadcast Encryption (used in DRM) [10], and Hidden Credentials [11,12]. Note that the techniques used in these applications are not collusion-resistant, so they can not be classified as ABE. It remains to be examined if ABE techniques can be used to improve the solutions.

3 DABE

In this section we formally define the concept of DABE and introduce the required keys and algorithms. Our construction will be detailed in Section 4. Table 1 on the next page provides a quick reference of the most relevant keys.

3.1 Users, Attributes and Keys

During setup, a public master key PK and a secret master key MK are generated; PK is available to every party, whereas MK is only known to the master. Every user u maintains a public user key PK_u, which is used by attribute authorities to generate personalized secret attribute keys, and a secret key SK_u, which is used in the decryption operation. Generation and distribution of PK_u and SK_u is the task of the master, who is also required to verify the identity of the users before keys are issued. The keys SK_u and PK_u of a user u are bound to the

Table 1. Summary of DABE keys

Key	Description	Usage
PK	Global key	Input for all operations
MK	Master key	Creation of user keys
SK_a	Secret key of attribute authority a	Creation of attribute keys
$\mathrm{PK}_{\mathcal{A}}$	Public key of attribute $\mathcal{A}$	Encryption
$\mathrm{SK}_{\mathcal{A},}$	Secret key of attribute $\mathcal{A}$ for user u	Decryption
PK	Public key of user u	Key Request
SK	Secret key of user u	Decryption

identity and/or pseudonyms of the user by the master. This binding is crucial for the verification of the user's attributes.

Every attribute authority maintains a secret key SK_a which is used to issue secret attribute keys to users. An attribute is a tuple consisting of an identifier of an attribute authority (e.g. an URL) and an identifier describing the attribute itself (an arbitrary string). We will denote the public representation of the attribute as $\mathcal{A}$ and use $a_{\mathcal{A}}$ as the identifier of the attribute authority present within $\mathcal{A}$. For every attribute with representation $\mathcal{A}$ there is a public key, denoted $\mathrm{PK}_{\mathcal{A}}$, which is issued by the respective attribute authority and is used to encrypt messages. The corresponding secret attribute keys, personalized for eligible users, are issued by the attribute authorities to users who request them (after determining their eligibility). To prevent collusions, every user gets a different secret attribute key that only he can use. A secret attribute key of an attribute $\mathcal{A}$, issued for a user u is denoted by $\mathrm{SK}_{\mathcal{A},u}$. We call the set of secret keys that a user has (i.e., the key SK_u and all keys $\mathrm{SK}_{\mathcal{A},u}$) his *key ring.*

3.2 The DABE Scheme

The DABE scheme consists of seven fundamental algorithms: *Setup*, *CreateUser*, *CreateAuthority*, *RequestAttributePK*, *RequestAttributeSK*, *Encrypt* and *Decrypt.* The description of the seven algorithms is as follows:

Setup. The *Setup* algorithm takes as input the implicit security parameter 1^k. It outputs the public key PK and the master key MK.

CreateUser$(\mathrm{PK}, \mathrm{MK}, u)$**.** The *CreateUser* algorithm takes as input the public key PK, the master key MK, and a user name u. It outputs a public user key PK_u, that will be used by attribute authorities to issue secret attribute keys for u, and a secret user key SK_u, used for the decryption of ciphertexts.

CreateAuthority(PK, a)**.** The *CreateAuthority* algorithm is executed by the attribute authority with identifier a once during initialization. It outputs a secret authority key SK_a.

RequestAttributePK$(\mathrm{PK}, \mathcal{A}, \mathrm{SK}_a)$**.** The *RequestAttributePK* algorithm is executed by attribute authorities whenever they receive a request for a public attribute key. The algorithm checks whether the authority identifier $a_{\mathcal{A}}$

of $\mathcal{A}$ equals a. If this is the case, the algorithm outputs a public attribute key for attribute $\mathcal{A}$, denoted $\mathrm{PK}_{\mathcal{A}}$, otherwise NULL.

RequestAttributeSK$(\mathrm{PK}, \mathcal{A}, \mathrm{SK}_a, u, \mathrm{PK}_u)$**.** The *RequestAttributeSK* algorithm is executed by the attribute authority with identifier a whenever it receives a request for a secret attribute key. The algorithm checks whether the authority identifier $a_{\mathcal{A}}$ of $\mathcal{A}$ equals a and whether the user u with public key PK_u is eligible of the attribute $\mathcal{A}$. If this is the case, *RequestAttributeSK* outputs a secret attribute key $\mathrm{SK}_{\mathcal{A},u}$ for user u. Otherwise, the algorithm outputs NULL.

Encrypt$(\mathrm{PK}, M, \mathbb{A}, \mathrm{PK}_{\mathcal{A}_1}, \ldots, \mathrm{PK}_{\mathcal{A}_N})$**.** The *Encrypt* algorithm takes as input the public key PK, a message M, an access policy $\mathbb{A}$ and the public keys $\mathrm{PK}_{\mathcal{A}_1}, \ldots, \mathrm{PK}_{\mathcal{A}_N}$ corresponding to all attributes occurring in the policy $\mathbb{A}$. The algorithm encrypts M with $\mathbb{A}$ and outputs the ciphertext CT.

Decrypt$(\mathrm{PK}, \mathrm{CT}, \mathbb{A}, \mathrm{SK}_u, \mathrm{SK}_{\mathcal{A}_1,u}, \ldots, \mathrm{SK}_{\mathcal{A}_N,u})$**.** The *Decrypt* algorithm takes as input a ciphertext produced by the *Encrypt* algorithm, an access policy $\mathbb{A}$, under which CT was encrypted, and a key ring $\mathrm{SK}_u, \mathrm{SK}_{\mathcal{A}_1,u}, \ldots, \mathrm{SK}_{\mathcal{A}_N,u}$ for user u. The algorithm *Decrypt* decrypts the ciphertext CT and outputs the corresponding plaintext M if the attributes were sufficient to satisfy $\mathbb{A}$; otherwise it outputs NULL.

Note that this scheme differs from CP-ABE [1] in that the two algorithms *CreateAuthority* and *RequestAttributePK* were added, and CP-ABE's algorithm *KeyGen* is split up into *CreateUser* and *RequestAttributeSK*. It is crucial that *RequestAttributeSK* does not need any components of the master key MK as input, so that every attribute authority is able to independently create attributes. However, we still require that a trusted central party maintains users (executes *CreateUser*), as otherwise collusion attacks would be possible.

3.3 Security Model

We model the security of DABE in terms of a game between a challenger and an adversary, where the challenger plays the role of the master and all attribute authorities.

Setup. The challenger runs the *Setup* algorithm and gives the global key PK to the adversary.

Phase 1. The adversary asks the challenger for an abitrary number of user keys. The challenger calls *CreateUser* for each requested user and returns the resulting public and private user keys to the adversary. For each user the adversary can request an arbitrary number of secret and public attribute keys, that the challenger creates by calling *RequestAttributeSK* or *RequestAttributePK*, respectively. Whenever the challenger receives a request for an attribute $\mathcal{A}$ of authority a, he tests whether he has already created a secret key SK_a for a. If not, he first calls *CreateAuthority* to create the appropriate authority key (note that SK_a will not be made available to the adversary).

Challenge. The adversary submits two messages M_0 and M_1 and an access policy $\mathbb{A}$ such that none of the users that he created in Phase 1 satisfy $\mathbb{A}$. (If any user from Phase 1 satisfies $\mathbb{A}$, the challenger aborts.) As before, the challenger may have to call *CreateAuthority* to initialize attribute authorities. The challenger flips a coin b, encrypts M_b under $\mathbb{A}$, and gives the ciphertext CT to the adversary.

Phase 2. Like in Phase 1, the adversary may create an arbitrary number of users. He can also request more secret attribute keys for the users he created in Phase 1 and 2, but if any secret attribute key would give the respective user a set of attributes needed to satisfy $\mathbb{A}$, the challenger aborts. As before, the adversary can always request any public attribute key.

Guess. The adversary outputs a guess b' of b.

The advantage of the adversary in this game is defined as $\Pr[b' = b] - \frac{1}{2}$, where the probability is taken over all coin tosses of both challenger and adversary. A DABE scheme is secure if all polynomial time adversaries have at most a negligible advantage in the above game.

4 Our Construction

We construct an efficient DABE scheme as follows:

Setup. The *Setup* algorithm chooses a bilinear group $\mathbb{G}$ of order p and a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ [13]. Next it chooses a generator $g \in \mathbb{G}$, a random point $P \in \mathbb{G}$ and a random exponent $y \in \mathbb{Z}_p$. The public key of the system is $\mathrm{PK} = \{\mathbb{G}, \mathbb{G}_T, e, g, P, e(g, g)^y\}$, while the secret master key is given by $\mathrm{MK} = g^y$.

CreateUser$(\mathrm{PK}, \mathrm{MK}, u)$*.* The algorithm *CreateUser* chooses a secret $\mathrm{mk}_u \in \mathbb{Z}_p$ and outputs the public key $\mathrm{PK}_u := g^{\mathrm{mk}_u}$ and the private key $\mathrm{SK}_u := \mathrm{MK} \cdot P^{\mathrm{mk}_u} = g^y \cdot P^{\mathrm{mk}_u}$ for user u.

CreateAuthority(PK, a)*.* The algorithm *CreateAuthority* chooses uniformly and randomly a hash function $H_{x_a} : \{0,1\}^* \to \mathbb{Z}_p$ from a finite family of hash functions, which we model as random oracles. It returns as secret key the index of the hash function $\mathrm{SK}_a := x_a$.

RequestAttributePK$(\mathrm{PK}, \mathcal{A}, \mathrm{SK}_a)$*.* If $\mathcal{A}$ is handled by the attribute authority a (i.e., $a_{\mathcal{A}} = a$), *RequestAttributePK* returns the public attribute key of $\mathcal{A}$, which consists of two parts:

$$\mathrm{PK}_A := \left\langle\ \mathrm{PK}'_{\mathcal{A}} := g^{H_{\mathrm{SK}_a}(\mathcal{A})}, \quad \mathrm{PK}''_{\mathcal{A}} := e(g,g)^{y\, H_{\mathrm{SK}_a}(\mathcal{A})} \right\rangle \ .$$

This public key can be requested from the attribute authority by anyone, but *RequestAttributePK* can only be executed by the respective authority, because it requires the index of the hash function SK_a as input.

RequestAttributeSK$(\mathrm{PK}, \mathcal{A}, \mathrm{SK}_a, u, \mathrm{PK}_u)$. After determining that the attribute $\mathcal{A}$ is handled by a (i.e., $a_{\mathcal{A}} = a$), the authority tests whether user u is eligible for the attribute $\mathcal{A}$. If this is not the case, *RequestAttributeSK* returns `NULL`, else it outputs the secret attribute key

$$\mathrm{SK}_{\mathcal{A},u} := {\mathrm{PK}_u}^{H_{\mathrm{SK}_a}(\mathcal{A})} = g^{\mathrm{mk}_u\, H_{\mathrm{SK}_a}(\mathcal{A})}.$$

Note that the recipient u can check the validity of this secret key by testing if

$$e(\mathrm{SK}_u, \mathrm{PK}'_{\mathcal{A}}) = \mathrm{PK}''_{\mathcal{A}} \cdot e(P, \mathrm{SK}_{\mathcal{A},u}) \ .$$

Encrypt$(\mathrm{PK}, M, \mathbb{A}, \mathrm{PK}_{\mathcal{A}_1}, \ldots, \mathrm{PK}_{\mathcal{A}_N})$. A policy in DNF can be written as

$$\mathbb{A} = \bigvee_{j=1}^{n} \left(\bigwedge_{\mathcal{A} \in S_j} \mathcal{A} \right),$$

where n (not pairwise disjoint) sets $S_1, \ldots, S_n$ denote attributes that occur in the j-th conjunction of $\mathbb{A}$. The encryption algorithm iterates over all $j = 1, \ldots, n$, generates for each conjunction a random value $R_j \in \mathbb{Z}_p$ and constructs CT_j as

$$\begin{aligned}\mathrm{CT}_j := \big\langle\, E_j &:= M \cdot (\prod_{\mathcal{A} \in S_j} \mathrm{PK}''_{\mathcal{A}})^{R_j}, \\ E'_j &:= P^{R_j}, \\ E''_j &:= (\prod_{\mathcal{A} \in S_j} \mathrm{PK}'_{\mathcal{A}})^{R_j} \,\big\rangle \ . \end{aligned} \tag{1}$$

The ciphertext CT is obtained as tuple $\mathrm{CT} := \langle \mathrm{CT}_1, \ldots, \mathrm{CT}_n \rangle$.

Decrypt$(\mathrm{PK}, \mathrm{CT}, \mathbb{A}, \mathrm{SK}_u, \mathrm{SK}_{\mathcal{A}_1,u}, \ldots, \mathrm{SK}_{\mathcal{A}_N,u})$. To decrypt a ciphertext CT, *Decrypt* first checks whether any conjunction of $\mathbb{A}$ can be satisfied by the given attributes, i.e., whether the input $\mathrm{SK}_{\mathcal{A}_1,u}, \ldots, \mathrm{SK}_{\mathcal{A}_N,u}$ contains secret keys for all attributes occurring in a set S_j for some $1 \leq j \leq n$. If this is not the case, the algorithm outputs `NULL`, otherwise

$$M = E_j \cdot \frac{e(E'_j, \prod_{i \in S_j} \mathrm{SK}_{i,u})}{e(E''_j, \mathrm{SK}_u)} \ .$$

It is easy to see that the decryption is correct. Let $a_j := \sum_{\mathcal{A} \in S_j} H_{\mathrm{SK}_{a_{\mathcal{A}}}}(\mathcal{A})$. Then $E_j = M \cdot e(g,g)^{y a_j R_j}$, $E''_j = g^{a_j R_j}$ and

$$\begin{aligned} E_j \cdot \frac{e(E'_j, \prod_{i \in S_j} \mathrm{SK}_{i,u})}{e(E''_j, \mathrm{SK}_u)} &= M \cdot e(g,g)^{y a_j R_j} \cdot \frac{e(P^{R_j}, g^{\mathrm{mk}_u\, a_j})}{e(g^{a_j R_j}, g^y \cdot P^{\mathrm{mk}_u})} \\ &= M \cdot e(g,g)^{y a_j R_j} \cdot \frac{e(P,g)^{R_j\, \mathrm{mk}_u\, a_j}}{e(P,g)^{R_j\, \mathrm{mk}_u\, a_j} \cdot e(g,g)^{y a_j R_j}} = M \ . \end{aligned}$$

5 Discussion

In this section, we first comment on the performance of the proposed DABE scheme, give a security proof in the generic group model and finally comment on the delegation property.

5.1 Performance

Compared to other ABE schemes, the proposed DABE construction is very efficient. Nearly all operations are group operations in $\mathbb{G}$ and $\mathbb{G}_T$. The only computationally expensive operation—the pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$—is needed during decryption exactly two times, no matter how complex the access policy is. No pairings are needed for any other algorithms. In all other known ABE schemes, the number of pairings grows at least linearly with the minimum number of distinct attributes needed for decryption.

5.2 Security

We first give an intuitive security argument. Clearly, to decrypt a ciphertext CT without having access to a sufficient set of secret attribute keys, an adversary needs to find $e(g, g)^{y a_j R_j}$ for some $1 \le j \le n$ which allows him to obtain M from E_j (note that M only occurs in E_j). He thus must compute a pairing of g^α and $g^{y\beta}$ for some $\alpha, \beta \in \mathbb{Z}_p$, such that $\alpha\beta = a_j R_j$.

To create such a pairing, the adversary can only use keys that he has obtained before in a security game as defined in Section 3.3. We will show that, assuming the adversary has not enough secret keys to satisfy $\mathbb{A}$, he is not able to compute this value.

The only occurrence of g^y (aside from $e(g, g)^y$) is in the secret user keys, so the adversary has to use some SK_u for the pairing, yielding

$$e(g^\alpha, \gamma\,\mathrm{SK}_u) = e(g^\alpha, g^y) \cdot e(g^\alpha, P^{\mathrm{mk}_u}) \cdot e(g^\alpha, \gamma) \ ,$$

for some γ. Given all values that the adversary knows, the only useful choice for g^α is $E''_j = g^{a_j R_j}$ for some conjunction $\bigwedge_{\mathcal{A} \in S_j} \mathcal{A}$. Pairing E''_j with SK_u gives:

$$e(E''_j, \mathrm{SK}_u) = e(g, g)^{y a_j R_j} \cdot e(g, P)^{\mathrm{mk}_u\, a_j R_j} \ .$$

To obtain $e(g, g)^{y a_j R_j}$, the second factor has to be eliminated. However, all three exponents of $e(g, P)^{\mathrm{mk}_u\, a_j R_j}$ are unknown to the adversary, and no combination of two publicly known values or secret user keys holds exactly the desired components (assuming that the adversary does not have all required secret attribute keys), so this value cannot be computed by the adversary.

For a more thorough security proof of our construction, we will use the generic group model [2]. In this model, the elements of $\mathbb{G}$ and $\mathbb{G}_T$ are encoded as arbitrary strings that (from the adversary's point of view) appear random. All operations are computed by oracles, so the adversary is limited to the group operations, the pairing, and equality tests. A scheme proven secure this way is called *generically secure* and can only be broken by exploiting specific properties of the groups that are used to implement it.

Theorem 1. *Let Adv be a generic adversary who plays the DABE security game and makes q oracle queries. Then Adv has advantage at most $O(q^2/p)$ in the generic group model, where p is the order of the bilinear group.*

A proof of this theorem is given in the appendix.

5.3 Delegation

The CP-ABE schemes [1] and [4] support an additional mechanism called *Delegate* that allows a user to create a new key ring that contains a subset of his secret attribute keys. In a DABE scenario with separate *CreateUser* and *RequestAttributeSK* algorithms, delegation between users cannot be supported since it allows collusions. To see this, consider a user u who is eligible of a set of attributes S_u and a user v who is eligible of a set of attributes $S_v \neq S_u$. Now let S be a set of attributes such that $S \subset S_u \cup S_v$, but $S_u \not\subseteq S$ and $S_v \not\subseteq S$. To decrypt a ciphertext encrypted with a conjunction consisting of all attributes in S, the user u would use *CreateUser* and *RequestAttributeSK* to get all attributes of S_u, then call *Delegate* to create a key ring for v that contains $S \cap S_u$. Finally, v would then use *RequestAttributeSK* to add all remaining attributes $S_v' := S_v \setminus (S \cap S_u)$. This is possible as in a DABE scheme, the *RequestAttributeSK* can be called at any time to add private attribute keys to key rings. Subsequently, v could decrypt any ciphertext encrypted with S. For this reason, delegation is not allowed in DABE. In our scheme, key rings can be re-randomized in the same way as [1] and [3], so a new keyring containing a subset of the attributes of the old keyring can be generated that is usable for decryption. However, all values of the resulting keyring will contain a random mk_u that is not bound to any identity, so the user will not be able to add new attributes to it.

6 Conclusion

In this paper, we proposed the concept of Distributed Attribute-Based Encryption (DABE) as an extension of Ciphertext-Policy Attribute-Based Encryption (CP-ABE) that supports an arbitrary number of attribute authorities and allows to dynamically add new users and authorities at any time. We provided an efficient construction of DABE that uses only two pairing operations in the decryption algorithm and no pairing operation in any other algorithm.

A limitation of our construction is that access policies need to be in DNF form. We leave it as an open question to design a more expressive DABE scheme, while preferably maintaining the $O(1)$ number of pairings that our construction offers.

Acknowledgements

The authors wish to thank the reviewers of this paper on the ICISC 2008 program committee for some very helpful comments and suggestions.

References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society, Los Alamitos (2007)
2. 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)
3. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Technical report, SRI International (2008) (to appear)
4. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: ICALP (2008)
5. Cheung, L., Newport, C.C.: Provably secure ciphertext policy ABE. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM Conference on Computer and Communications Security, pp. 456–465. ACM, New York (2007)
6. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security, pp. 89–98. ACM, New York (2006)
7. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
8. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
9. Yuan, E., Tong, J.: Attributed based access control (ABAC) for web services. In: ICWS, pp. 561–569. IEEE Computer Society, Los Alamitos (2005)
10. Adelsbach, A., Huber, U., Sadeghi, A.R.: Property-based broadcast encryption for multi-level security policies. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 15–31. Springer, Heidelberg (2006)
11. Kapadia, A., Tsang, P.P., Smith, S.W.: Attribute-based publishing with hidden credentials and hidden policies. In: Proceedings of The 14th Annual Network and Distributed System Security Symposium (NDSS), pp. 179–192 (March 2007)
12. Bradshaw, R.W., Holt, J.E., Seamons, K.E.: Concealing complex policies with hidden credentials. In: Atluri, V., Pfitzmann, B., McDaniel, P.D. (eds.) ACM Conference on Computer and Communications Security, pp. 146–157. ACM, New York (2004)
13. Boneh, D.: A brief look at pairings based cryptography. In: FOCS, pp. 19–26. IEEE Computer Society, Los Alamitos (2007)
14. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptology 21(2), 149–177 (2008)

A Security Proof

We closely follow the structure of the proof of [1]: First we show how to reduce any adversary who plays the DABE game of Section 3.3 (denoted $\mathtt{Adv}_1$ in the following) to an adversary in a modified game (denoted $\mathtt{Adv}_2$). Then we show that no $\mathtt{Adv}_2$ has non-negligible advantage, so there can be no $\mathtt{Adv}_1$ with non-negligible advantage, either.

Let adversary $\mathtt{Adv}_1$ be an adversary who plays the DABE game defined in Section 3.3 using our construction from Section 4. We define a modified game as follows: The phases **Setup**, **Phase 1**, and **Phase 2** are equal to the DABE game. In the **Challenge** phase, the adversary submits an access policy $\mathbb{A}$ such that none of the users that he created in Phase 1 satisfy $\mathbb{A}$. The challenger flips a coin b, and creates a ciphertext for the access policy $\mathbb{A}$ according to Equation 1, but instead of computing $E_j := M \cdot e(g,g)^{y a_j R_j}$, he computes E_j as

$$E_j = \begin{cases} e(g,g)^{y a_j R_j}, & \text{if } b = 1 \\ e(g,g)^{\theta_j}, & \text{if } b = 0 \end{cases},$$

where all θ_j are uniformly and independently chosen random elements of $\mathbb{Z}_p$.

Given an adversary $\mathtt{Adv}_1$ that has advantage ϵ in the DABE game, we can construct $\mathtt{Adv}_2$ as follows: In the phases **Setup**, **Phase 1**, and **Phase 2**, $\mathtt{Adv}_2$ forwards all messages he receives from $\mathtt{Adv}_1$ to the challenger and all messages from the challenger to $\mathtt{Adv}_1$. In the **Challenge** phase, $\mathtt{Adv}_2$ receives two messages M_0 and M_1 from $\mathtt{Adv}_1$ and the challenge C (which is either $e(g,g)^{y a_j R_j}$ or $e(g,g)^{\theta_j}$) from the challenger. He flips a coin β and sends $C' := M_\beta \cdot C$ to $\mathtt{Adv}_1$. When $\mathtt{Adv}_1$ outputs a guess β', $\mathtt{Adv}_2$ outputs as its guess 1 if $\beta' = \beta$, or 0 if $\beta' \neq \beta$. If $C = e(g,g)^{y a_j R_j}$, then $\mathtt{Adv}_2$'s challenge is a well-formed DABE ciphertext and $\mathtt{Adv}_1$ has advantage ϵ of guessing the correct $\beta' = \beta$. If $C = e(g,g)^{\theta_j}$, the challenge is independent of the messages M_0 and M_1, so the advantage of $\mathtt{Adv}_2$ is 0. Thus, we have

$$\begin{aligned} \Pr[\mathtt{Adv}_2 \text{ succeeds}] &= \Pr[C = e(g,g)^{y a_j R_j}] \Pr[\beta' = \beta \,|\, C = e(g,g)^{y a_j R_j}] + \\ &\quad \Pr[C = e(g,g)^{\theta_j}] \Pr[\beta' \neq \beta \,|\, C = e(g,g)^{\theta_j}] \\ &\leq \frac{1}{2}(\frac{1}{2} + \epsilon) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1+\epsilon}{2} \end{aligned}$$

and the overall advantage of $\mathtt{Adv}_2$ is $\frac{\epsilon}{2}$, as required. The existence of any successful $\mathtt{Adv}_1$ implies the existence of an adversary $\mathtt{Adv}_2$ who succeeds with non-negligible advantage as well.

Our next step is to show that no $\mathtt{Adv}_2$ can distinguish between $e(g,g)^{y a_j R_j}$ and $e(g,g)^{\theta_j}$ in polynomial time. A combination of both results implies that no $\mathtt{Adv}_1$ can have non-negligible advantage, either.

To show this, we use the generic group model from [2], with the extensions for bilinear groups with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ developed in [14], which we simplify slightly for our case where $\mathbb{G}_1 = \mathbb{G}_2$. For groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p, the simulator chooses an arbitrary generator $\tilde{g} \in \mathbb{G}$ and uses random maps $\xi, \xi_T : \mathbb{Z}_p \to \{0,1\}^{\lceil \log p \rceil}$ that encode any $\tilde{g}^x$ or $e(\tilde{g},\tilde{g})^x$ as a random string. The maps ξ and ξ_T must be invertible, so the simulator can map representations back to elements of $\mathbb{G}$ and $\mathbb{G}_T$. The simulator gives the adversary two oracles that compute the group operations of $\mathbb{G}$ and $\mathbb{G}_T$ and another oracle that computes the pairing e. All oracles take as input string representations of group elements. The adversary can only perform operations on group elements by interacting with the oracles. For example, given two string representations $A := \xi_T(a)$ and $B := \xi_T(b)$ of elements of group $\mathbb{G}_T$, the adversary can query the group oracle for the result of the group operation $A \cdot B$ in $\mathbb{G}_T$. The simulator will map A and B back to the respective elements of $\mathbb{G}_T$ using ξ_T^{-1}, then execute the group operation and map the result to a string using ξ_T. From the view of the adversary, the simulator can simply return $\xi(a) \cdot \xi(b) = \xi(a+b)$ or $\xi_T(a) \cdot \xi_T(b) = \xi_T(a+b)$ for multiplication and $\xi(a)/\xi(b) = \xi(a-b)$ or $\xi_T(a)/\xi_T(b) = \xi_T(a-b)$ for division. Note that no oracle will accept input from different encodings (for example, one cannot compute $\xi(a) \cdot \xi_T(b)$). The oracle for e can be implemented easily: Given two encodings $A = \xi(a)$ and $B = \xi(b)$ for some $a, b \in \mathbb{Z}_p$, the encoding of the pairing of A and B is $\xi_T(ab)$. We assume that the adversary only makes oracle queries on strings previously obtained from the simulator.

The simulator plays the DABE game as follows:

Setup. The simulator chooses $\mathbb{G}$, $\mathbb{G}_T$, e, $\tilde{g}$ and random exponents $y, \tilde{p} \in \mathbb{Z}_p$, as well as two encoding functions ξ, ξ_T and oracles for the group operations in $\mathbb{G}$, $\mathbb{G}_T$, and the pairing as described above. The public parameters are $g := \xi(1)$, $P := \xi(\tilde{p})$, and $Y := \xi_T(y)$.

Phase 1. When the adversary calls *CreateUser* for some u, the simulator chooses a random $\mathrm{mk}_u \in \mathbb{Z}_p$ and returns $\mathrm{PK}_u := \xi(\mathrm{mk}_u)$ and $\mathrm{SK}_u := \xi(y + \tilde{p} \cdot \mathrm{mk}_u)$. Whenever the simulator gets a request involving an attribute $\mathcal{A}$ that the adversary has not used before, he chooses a new, unique random value $\mathrm{mk}_{\mathcal{A}}$, which simulates the term $H_{\mathrm{SK}_a}(\mathcal{A})$ of an attribute $\mathcal{A}$ with attribute authority $a = a_{\mathcal{A}}$. (The association between values $\mathrm{mk}_{\mathcal{A}}$ and attributes $\mathcal{A}$ is stored for future queries). For every public attribute key request for an attribute $\mathcal{A}$, the simulator returns $\mathrm{PK}'_{\mathcal{A}} := \xi(\mathrm{mk}_{\mathcal{A}})$ and $\mathrm{PK}''_{\mathcal{A}} := \xi_T(y\,\mathrm{mk}_{\mathcal{A}})$. If queried for a secret attribute key, the simulator returns $\mathrm{SK}_{\mathcal{A},u} = \xi(\mathrm{mk}_u\,\mathrm{mk}_{\mathcal{A}})$.

Challenge. When the adversary asks for a challenge, the simulator flips a coin b. Then he chooses a random $R_j \in \mathbb{Z}_p$ for each conjunction $\mathbb{A}_j$ and computes $a_j = \sum_{\mathcal{A} \in S_j} \mathrm{mk}_{\mathcal{A}}$. If $b = 0$, he sets θ_j to a random value from $\mathbb{Z}_p$, otherwise $\theta_j := y a_j R_j$. Finally he computes the ciphertext components of CT_j as

$$\left\langle\, E_j := \xi_T(\theta_j),\ E'_j := \xi(\tilde{p} R_j),\ E''_j := \xi(a_j R_j) \,\right\rangle \ ,$$

which he returns as ciphertext.

Phase 2. The simulator behaves as in Phase 1. However, the simulator refuses any secret attribute key that would give the respective user a set of attributes satisfying $\mathbb{A}$.

All values that the adversary knows are either encodings of random values of $\mathbb{Z}_p$ (namely $1, \tilde{p}, y, \mathrm{mk}_u, \mathrm{mk}_{\mathcal{A}}$ and θ), combinations of these values given to him by the simulator (for example $\mathrm{SK}_{\mathcal{A},u} = \xi(\mathrm{mk}_u\,\mathrm{mk}_{\mathcal{A}})$), or results of oracle queries on combinations of these values. We keep track of the algebraic expressions used to query the oracles; all queries can thus be written as rational functions. We assume that different terms always result in different string representations. This assumption can only be false if due to the choice of the random encodings two different terms "accidentally" result in the same string representation. Similar to the proof in [1] it can be shown that the probability of this event is $O(q^2/p)$ where q is the number of oracle queries that the adversary makes. In the following we will condition that no such event occurs.

Now, under this assumption consider how the adversary's views differ between the case where the θ_j are random ($b = 0$) and the case where $\theta_j = ya_jR_j$ ($b = 1$). We claim that the views are identically distributed for both cases and therefore any adversary has no advantage to distinguish between them in the generic group model. To proof this claim, assume the opposite. Since the adversary can only test for equality of string representations he receives (and all representations of group elements are random), the only possibility for the views to differ is that there exist two different terms that result in the same answer in the view where $\theta_j = ya_jR_j$ ($b = 1$), for at least one j, and in different answers in the view corresponding to $b = 0$. Call two such terms ν_1 and ν_2 and fix one relevant j. Since θ_j only occurs as $E_j := \xi_T(\theta_j)$ and elements of ξ_T cannot be paired, the adversary can only construct queries where θ_j appears as an additive term. Thus, ν_1 and ν_2 can be written as

$$\begin{aligned} \nu_1 &= \gamma_1\theta_j + \nu_1' \\ \nu_2 &= \gamma_2\theta_j + \nu_2' \ , \end{aligned}$$

for some ν_1' and ν_2' that do not contain θ_j. Since by assumption $\theta_j = ya_jR_j$ results in $\nu_1 = \nu_2$, we have $\gamma_1 ya_jR_j + \nu_1' = \gamma_2 ya_jR_j + \nu_2'$. Rearranging the equation yields

$$\nu_1' - \nu_2' = (\gamma_2 - \gamma_1)ya_jR_j \ .$$

Thus, the adversary can construct an oracle query for a term γya_jR_j (which we can, without loss of generality, add to the queries of the adversary).

It remains to be shown that, without having a sufficient set of attributes satisfying $\mathbb{A}$, the adversary *cannot* construct a query of the form $\xi_T(\gamma ya_jR_j)$ for any γ and j from the information that he has. This contradicts the assumption that the views in the modified game are not identically distributed.

After Phase 2, the adversary has received the following information from the simulator:

- The tuple PK.
- PK_u and SK_u for an arbitrary number of users.
- $\mathrm{PK}'_{\mathcal{A}}$ and $\mathrm{PK}''_{\mathcal{A}}$ for an arbitrary number of attributes.
- $\mathrm{SK}_{\mathcal{A},u}$ for an arbitrary number of attributes and users, with the restriction that for no u, he has a sufficient set of secret attributes keys that satisfies $\mathbb{A}$.
- E_j, E_j', and E_j'' of the challenge ciphertext.

Furthermore, he possibly obtained encodings of arbitrary combinations of these values through queries to the five oracles that implement the group operations and the pairing. Since all R_j and y are random, the only way to construct

Table 2. Results of pairings

Source	Term	Pairing with SK	Pairing with E'
$\mathrm{PK}\ {}'$	$\mathrm{mk}\ {}'$	$\mathrm{mk}\ {}' y + \tilde{p}\,\mathrm{mk}\ \ \mathrm{mk}\ {}'$	$\mathrm{mk}\ \ \tilde{p}R$
$\mathrm{SK}\ {}'$	$y + \tilde{p}\,\mathrm{mk}\ {}'$	$y^2 + y\tilde{p}(\mathrm{mk}\ {}' + mk\ \ ) + \tilde{p}^2\,\mathrm{mk}\ \ \mathrm{mk}\ {}'$	$y\tilde{p}R\ \ + \tilde{p}^2\,\mathrm{mk}\ \ R$
$\prod_{\mathcal{A}\in S_j} \mathrm{PK}'_{\mathcal{A}}$	a	$ya\ \ + \tilde{p}\,\mathrm{mk}\ \ a$	$a\ \tilde{p}R$
E'	$\tilde{p}R$	$y\tilde{p}R\ \ + \tilde{p}^2\,\mathrm{mk}\ \ R$	$\tilde{p}^2R^2$
E''	$a\ R$	$ya\ R\ \ + \tilde{p}\,\mathrm{mk}\ \ a\ R$	$a\ \tilde{p}R^2$

$\xi_T(\gamma y a_j R_j)$ is to pair two values from $\mathbb{G}$ by querying the pairing oracle, so that each of the components is contained in any of the terms.

First we show how the adversary can find representations of terms that contain a_j. Aside from E_j and E''_j, a_j can only be constructed by querying the multiplication oracle for encodings of the terms containing $\mathrm{mk}_{\mathcal{A}}$ for all $\mathcal{A} \in S_j$ and some j with $1 \leq j \leq n$. These values occur only in $\mathrm{PK}'_{\mathcal{A}}$, $PK''_{\mathcal{A}}$, and $\mathrm{SK}_{\mathcal{A},u}$. Since $PK''_{\mathcal{A}} \in \mathbb{G}_T$, it cannot be used as input of the pairing. Multiplying representations of $\mathrm{PK}'_{\mathcal{A}}$ for some $\mathcal{A}$ and $\mathrm{SK}_{\mathcal{A},u}$ for some u and $\mathcal{A}$ yields terms with exponents of the form

$$\sum_u \left(\gamma_u\,\mathrm{mk}_u \sum_{\mathcal{A}} \gamma_{\mathcal{A},u}\,\mathrm{mk}_{\mathcal{A}} \right) + \gamma' \sum_{\mathcal{A}} \mathrm{mk}_{\mathcal{A}} \ \ ,$$

for some γ_u, $\gamma_{\mathcal{A},u}$ and γ'. Since the adversary does not have all secret attribute keys corresponding to any one user u to satisfy any conjunction, no sum $\sum_{\mathcal{A}} \gamma_{\mathcal{A},u}\,\mathrm{mk}_{\mathcal{A}}$ will yield any a_j. Furthermore, the simulator chooses all $\mathrm{mk}_{\mathcal{A}}$ randomly, so any oracle query involving any sum over $\sum_{\mathcal{A}} \mathrm{mk}\,\mathcal{A}$ with a set of attributes that does not precisely correspond to the attributes of the challenge $\mathbb{A}$ gives no information about a_j. The only way that the sum $\gamma' \sum_{\mathcal{A}} \mathrm{mk}_{\mathcal{A}}$ evaluates to a_j for some j is as a product of corresponding public attribute keys, which is obtained by querying the multiplication oracle with all representations of $\mathrm{PK}'_{\mathcal{A}}, \mathcal{A} \in S_j$, yielding $\xi(a_j)$. It follows that to construct a term containing a_j, the adversary has no other option than to use either E_j, E''_j, or $\prod_{\mathcal{A}\in S_j} \mathrm{PK}_{\mathcal{A}}$ for any j. Other terms containing $\mathrm{mk}_{\mathcal{A}}$ are not useful for him.

Next we consider how to obtain terms containing y and R_j. All R_j and y are random, so the only way to construct a relevant pairing is to pair two representations of terms from $\mathbb{G}$ by querying the pairing oracle, such that both y and one R_j are contained in one of the terms. The only values in $\mathbb{G}$ that contain y are SK_u. We examine all possible results from pairing some $\gamma\,\mathrm{SK}_u$ with some other value. As shown above, we need not consider terms where the adversary has some $\mathrm{mk}_{\mathcal{A}}$, but not all to create a value a_j.

The first three columns of Table 2 list all remaining combinations. It can be seen that the only result that contains all y, a_j and R_j is the pairing of some SK_u and some E''_j which results in

$$\xi_T(\tilde{p}R_j\,\mathrm{mk}_u\,a_j + y a_j R_j) \ \ .$$

In order to obtain the required term $\xi_T(\gamma y a_j R_j)$, the adversary will have to eliminate the first term, $\tilde{p} R_j \,\mathrm{mk}_u\, a_j$. To construct this, he needs to pair a term containing $\tilde{p}$ with another term. Thus we need to examine all possible results from pairing SK_u or E'_j (the only terms depending on $\tilde{p}$) with another value. Once again, Table 2 on the previous page lists all possible combinations not containing terms involving results of the hash oracle. (Including terms given by the oracles one gets terms of the above form that will not help, either.) We can conclude from the case analysis that no term of the form $\xi_T(\tilde{p} R_j \,\mathrm{mk}_u\, a_j)$ can be constructed.

Thus, the term $\xi_T(y a_j R_j)$ cannot be constructed by the adversary, which contradicts the assumption that the views in the modified game are not identically distributed. Thus, any $\mathtt{Adv}_2$ will have negligible success to win the game. In turn, a successful $\mathtt{Adv}_1$ cannot exist either, which proves the theorem. □

Improved Partial Key Exposure Attacks on RSA by Guessing a Few Bits of One of the Prime Factors

Santanu Sarkar and Subhamoy Maitra

Indian Statistical Institute, 203 B T Road, Kolkata 700 108, India
{santanu_r,subho}@isical.ac.in

Abstract. Consider RSA with $N = pq$, $q < p < 2q$, public encryption exponent e and private decryption exponent d. We study cryptanalysis of RSA when certain amount of the Most Significant Bits (MSBs) or Least Significant Bits (LSBs) of d is known. This problem has been well studied in literature as evident from the works of Boneh et. al. in Asiacrypt 1998, Blömer et. al. in Crypto 2003 and Ernst et. al. in Eurocrypt 2005. In this paper, we achieve significantly improved results by modifying the techniques presented by Ernst et. al. Our novel idea is to guess a few MSBs of the secret prime p (may be achieved by exhaustive search over those bits in certain cases) that substantially reduces the requirement of MSBs of d for the key exposure attack.

Keywords: Cryptanalysis, Factorization, Lattice, LLL Algorithm, RSA, Side Channel Attacks, Weak Keys.

1 Introduction

RSA [19] is one of the most popular cryptosystems in the history of cryptology. Here, we use the standard notations in RSA as follows:

- primes p, q, with $q < p < 2q$;
- $N = pq$, $\phi(N) = (p-1)(q-1)$;
- e, d are such that $ed = 1 + k\phi(N)$, $k \geq 1$;
- N, e are publicly available and message M is encrypted as $C = M^e \bmod N$;
- the secret key d is required to decrypt the cipher as $M = C^d \bmod N$.

Though RSA is quite safe till date if applied with proper cryptographic practices, the literature related to its cryptanalysis is quite rich. RSA is found to be weak when the prime factors of any one of $p \pm 1$, $q \pm 1$ is small [18,25]. In [11], it has been pointed out that short public exponents may cause weakness if the same message is broadcast to many parties. One very important result regarding RSA weak keys has been presented in [24], where it has been shown that N can be factored from the knowledge of N, e if $d < \frac{1}{3}N^{\frac{1}{4}}$. Though it has been shown [21] that the idea of [24] cannot be substantially extended further

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 37–51, 2009.

than the bound of d as $O(N^{\frac{1}{4}})$, many papers [3,9,23,26] used the idea of Continued Fraction (CF) expression to get different kinds of weak keys in RSA (one may follow the material from [20, Chapter 5] for basics of CF expression and related results). The seminal idea of [8] using lattice based techniques has also been exploited in great detail [5,1] to find weak keys of RSA when $d < N^{0.292}$. An outstanding survey on the attacks on RSA before the year 2000 is available in [4]. For very recent results on RSA, one may refer to [2,13,14] and the references therein.

One important model of cryptanalysis is the side channel attack such as fault attacks, timing attacks, power analysis etc. [6,15,16], by which an adversary may obtain some bits of the private key d. In [6], it has been studied how many bits of d need to be known to mount an attack on RSA. The constraint in the work of [6] was the upper bound on e which is $\sqrt{N}$. The study attracted interest and the idea of [6] has been improved in [2] where the bound of e was increased upto $N^{0.725}$. Then the work of [10] improved the result for full size public exponent e. We present further improvement over the work of [10] noting that if one guesses a few MSBs of p, then the requirement on the number of bits in d gets substantially reduced.

As an example (see Example 1 later) with practical parameters, for a specific 1024 bit N and 309 bit d, the idea of [10] requires 112 many MSBs of d to be exposed, whereas, our idea requires only 80 MSBs of d with a guess of 21 bits of MSBs in p. First of all, the total requirement of bits to be known in our case is $80 + 21 = 101$, which is 11 bits less than the 112 many bits to be known in [10]. More importantly, one needs to know the bits of d by side channel attacks and a reduction of $112 - 80 = 32$ bits makes the chance of this kind of attack more realistic. Further, with higher lattice dimension we get even more interesting results where as less as 53 many MSBs of d are required with the knowledge of 21 many MSBs of p.

One may note that given the constraint $q < p < 2q$, a few bits of p, q can be known in polynomial time (e.g., around 7 bits for 1024 bit N and 9 bits for 2048 bit N following the work of [22]). This will indeed reduce the search effort further for guessing a few MSBs of p.

As we use different notations in this paper compared to [10], let us list the results of [10] with our notations here. Let d be of bitsize $\delta \log_2 N$. Given $(\delta - \gamma)\log_2 N$ many MSBs of d, the product N can be factored in probabilistic polynomial time [10] (we ignore the term ϵ as given in [10]) if

1. $\gamma \leq \frac{5}{6} - \frac{1}{3}\sqrt{1 + 6\delta}$, or
2. $\gamma \leq \frac{1}{3}\lambda + \frac{1}{2} - \frac{1}{3}\sqrt{4\lambda^2 + 6\lambda}$, where $\lambda = \max\{\gamma, \delta - \frac{1}{2}\}$

There are also some results in [10], where cryptanalysis of RSA is studied when some LSBs of d are known.

In this paper we use similar kind of analysis as in [10] and explain different cases relevant to the attacks. The theoretical results are presented in Theorems 2, 3 and 4. The advantages of our work over [10] are as follows.

1. Given that a few MSBs of p can be guessed, the requirement of MSBs of d in our attack is less than that of [10] (where no guess on p is made).
2. The total amount of bits, to be known considering the MSBs of both p, d in our case, is less than the number of MSBs to be known for d as reported in [10].
3. In Theorem 4, we have also studied the cryptanalysis of RSA when some MSBs of p along with the LSBs of d are known and our results is better than that of [10].

Our theoretical results are supported by experimental evidences. We have implemented the programs in SAGE 2.10.1 over Linux Ubuntu 7.04 on a computer with Dual CORE Intel(R) Pentium(R) D CPU 2.80 GHz, 1 GB RAM and 2 MB Cache.

While comparing our experimental results with that of [10], we implement the idea of [10] on our own platform. As all the parameters for the experiments in [10] may not be the same with our implementations, the results may vary a little. We point out the exact experimental values presented in [10] as and when required.

1.1 Preliminaries

Let us present some basics on lattice reduction techniques. Consider the linearly independent vectors $u_1, \ldots, u_w \in Z^n$, when $w \leq n$. A lattice, spanned by $< u_1, \ldots, u_w >$, is the set of all linear combinations of $u_1, \ldots, u_w$, i.e., w is the dimension of the lattice. A lattice is called full rank when $w = n$. Let L be a lattice spanned by linearly independent vectors $u_1, \ldots, u_w$, where $u_1, \ldots, u_w \in Z^n$. By $u_1^*, \ldots, u_w^*$, we denote the vectors obtained by applying the Gram-Schmidt process to the vectors $u_1, \ldots, u_w$. It is known that given a basis $u_1, \ldots, u_w$ of a lattice L, LLL algorithm [17] can find a new basis $b_1, \ldots, b_w$ of L with the following properties.

- $\| b_i^* \|^2 \leq 2 \| b_{i+1}^* \|^2$, for $1 \leq i < w$.
- For all i, if $b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$ then $|\mu_{i,j}| \leq \frac{1}{2}$ for all j.
- $\| b_1 \| \leq 2^{\frac{w}{2}} \det(L)^{\frac{1}{w}}$, $\| b_2 \| \leq 2^{\frac{w}{2}} \det(L)^{\frac{1}{w-1}}$.

By $b_1^*, \ldots, b_w^*$, we mean the vectors obtained by applying the Gram-Schmidt process to the vectors $b_1, \ldots, b_w$.

The determinant of L is defined as $\det(L) = \prod_{i=1}^{w} ||u_i^*||$, where $||.||$ denotes the Euclidean norm on vectors. Given a polynomial $g(x, y) = \sum a_{i,j} x^i y^j$, we define the Euclidean norm as $\| g(x, y) \| = \sqrt{\sum_{i,j} a_{i,j}^2}$ and infinity norm as $\| g(x, y) \|_\infty = \max_{i,j} |a_{i,j}|$.

In [8], techniques have been discussed to find small integer roots of polynomials in a single variable $\bmod\ n$, and of polynomials in two variables over the integers. The idea of [8] extends to more than two variables also, but the method becomes probabilistic. The following theorem is also relevant to the idea of [8].

Theorem 1. *[12] Let $g(x,y,z)$ be a polynomial which is a sum of ω many monomials. Suppose $g(x_0,y_0,z_0) = 0 \bmod n$, where $|x_0| < X$, $|y_0| < Y$ and $|z_0| < Z$. If $\| g(xX,yY,zZ) \| < \frac{n}{\sqrt{\omega}}$, then $g(x_0,y_0,z_0) = 0$ holds over integers.*

Thus, the condition $2^{\frac{w}{2}} \det(L)^{\frac{1}{w-1}} < \frac{n}{\sqrt{\omega}}$ implies that if polynomials b_1, b_2 (corresponding to the two shortest reduced basis vectors) have roots over $0 \bmod n$, then those roots hold over integers also. The solutions corresponding to each unknown is achieved by calculating the resultant of two polynomials (if they are algebraically independent) and then finding the solution of the resultant.

2 MSBs of *d* and *p* Known

In this section we consider that certain amount of MSBs of both d, p will be available. We will study to methods following the ideas of [10].

2.1 Method I

Let us start with the following result.

Theorem 2. *Let $d \leq N^{\delta}$ and consider that d_0, p_0 are exposed such that $|d - d_0| < N^{\gamma}$ and $|p - p_0| < N^{\beta}$. Then one can factor N (in probabilistic polynomial time) when*

$$\gamma \leq 1 - \frac{\beta + 2\sqrt{\beta(\beta + 3\delta)}}{3}.$$

Proof. Let $q_0 = \lfloor \frac{N}{p_0} \rfloor$. We have $ed - 1 = k\phi(N) = k(N - (p + q - 1))$. Now writing $d = d_0 + d_1$, the above equation can be written as $e(d_0 + d_1) - 1 = k(N - p_0 - q_0 - (p + q - p_0 - q_0 - 1))$. This can be rewritten as $ed_1 - (N - p_0 - q_0)k + k(p + q - p_0 - q_0 - 1) + ed_0 - 1 = 0$. Let us consider the corresponding polynomial $f_{MSB1} = ex - (N - p_0 - q_0)y + yz + R$, where $R = ed_0 - 1$, and d_1 is renamed as x, k is renamed as y and $p + q - p_0 - q_0 - 1$ is renamed as z. Hence, we have to find the solution $(x_0, y_0, z_0) = (d_1, k, p + q - p_0 - q_0 - 1)$ of the polynomial $f_{MSB1} = ex - (N - p_0 - q_0)y + yz + R$.

Let $X = N^{\gamma}, Y = N^{\delta}, Z = N^{\beta}$, and one can check that they are the upper bounds of x_0, y_0, z_0. Note that k (renamed as y) is less than $d = N^{\delta}$. Also, the bound Z should be $|p + q - p_0 - q_0|$, which is actually less than $2N^{\beta}$ (however, we ignore the constant term in the proof as in [10]).

Let us fix the lattice parameters m, t. Define $W = ||f_{MSB1}(xX, yY, zZ)||_{\infty}$ and $n = (XY)^m Z^{m+t} W$. In order to work with a polynomial having the constant term 1, we define

$$f \equiv R^{-1} f_{MSB1}(x, y, z) \bmod n \equiv 1 + ax + by + cyz.$$

(During the experiments, as long as gcd of R, n is not 1, we keep on increasing n by 1.) Then we use the shifts

$$\begin{aligned} g_{ijk} &= x^i y^j z^k f(x,y,z) X^{m-i} Y^{m-j} Z^{m+t-k}, \\ &\quad \mathit{for}\ i = 0,\ldots,m;\ j = 0,\ldots,m-i;\ k = 0,\ldots,j; \end{aligned}$$

$$\begin{aligned} h_{ijk} &= x^i y^j z^k f(x,y,z) X^{m-i} Y^{m-j} Z^{m+t-k}, \\ &\quad \mathit{for}\ i = 0,\ldots,m;\ j = 0,\ldots,m-i;\ k = j+1,\ldots,j+t; \end{aligned}$$

$$\begin{aligned} g'_{ijk} &= n x^i y^j z^k, \\ &\quad \mathit{for}\ i = 0,\ldots,m+1;\ j = m+1-i;\ k = 0,\ldots,j; \end{aligned}$$

$$\begin{aligned} h'_{ijk} &= n x^i y^j z^k, \\ &\quad \mathit{for}\ i = 0,\ldots,m+1;\ j = m+1-i;\ k = j+1,\ldots,j+t. \end{aligned}$$

Now we build a lattice L with the basis elements coming from the coefficient vectors of $g_{ijk}(xX, yY, zZ), h_{ijk}(xX, yY, zZ), g'_{ijk}(xX, yY, zZ)$ and $h'_{ijk}(xX, yY, zZ)$ following the idea of [10]. The vectors are ordered in such a manner that the matrix corresponding to the lattice L becomes triangular, and the diagonal entries of g and h are equal to $(XY)^m Z^{m+t}$. Then we follow the similar computation as in [10, Appendix A], taking $t = \tau m$. If

$$X^{1+3\tau} Y^{2+3\tau} Z^{1+3\tau+3\tau^2} \le W^{1+3\tau}, \tag{1}$$

we get polynomials f_1 and f_2 (the first two elements after lattice reduction using LLL algorithm) that satisfy the Howgrave-Graham bound as described in Theorem 1.

Now we construct two resultants G_1, G_2 taking two different pairs from f_{MSB1}, f_1, f_2 (in our experiments, mostly, G_1 is constructed using f_{MSB1}, f_1 and G_2 is constructed using f_{MSB1}, f_2). Then we construct the resultant of G_1, G_2 to get G. The integer root of G provide z_0, which in turn gives the primes. We assume that the resultant computations for multivariate polynomials constructed in our approach yield non-zero polynomials. This is successful in most of the cases in our experiment. However, as this step involves some probability of success, we consider the algorithm as probabilistic. As each of lattice reduction, resultant computation and root finding is polynomial time algorithm in $\log_2 N$, the product N can be factored in probabilistic polynomial time given the constraints in this theorem.

Here $X = N^\gamma, Y = N^\delta, Z = N^\beta$, and

$$W = \max(eX, (N - p_0 - q_0)Y, YZ, R) \ge (N - p_0 - q_0)Y \approx NY = N^{1+\delta}.$$

So the inequality 1 holds if,

$$\begin{aligned} X^{1+3\tau} Y^{2+3\tau} Z^{1+3\tau+3\tau^2} &\le (NY)^{1+3\tau} \;\Leftrightarrow \\ N^{\gamma(1+3\tau)} N^{\delta(2+3\tau)} N^{\beta(1+3\tau+3\tau^2)} &\le N^{(1+\delta)(1+3\tau)} \;\Leftrightarrow \\ 3\beta\tau^2 + (3\beta + 3\gamma - 3)\tau + (\gamma + \delta + \beta - 1) &\le 0. \end{aligned}$$

Putting the optimal value of τ, which is $\tau = \frac{1-\beta-\gamma}{2\beta}$ in the above inequality we get the required condition $\gamma \leq \frac{(3-\beta)-\sqrt{4\beta^2+12\beta\delta}}{3}$. □

One may note that putting $\beta = \frac{1}{2}$ in Theorem 2, we get the same bound $\gamma \leq \frac{5}{6} - \frac{1}{3}\sqrt{1+6\delta}$ as in [10, Theorem 1]. As we have the knowledge of a few MSBs of p, the value of β decreases below $\frac{1}{2}$ in our case, increasing the value of γ. As $\delta - \gamma$ proportion of bits of d needs to be known for the attack, we require less number of MSBs of d to be exposed than [10].

We present some numerical values first. Consider 1024 bits N, i.e., $\log_2 N = 1024$ and $\delta = 0.35$, i.e., $\log_2 d = 359$. Thus, the upper bound of γ using the formula $\gamma \leq \frac{5}{6} - \frac{1}{3}\sqrt{1+6\delta}$ of [10] comes to be 0.24644. Then the requirement of MSBs of d to be known is $(0.35 - 0.24644) \times 1024 = 106$ bits. If we consider that 0.039 proportion of MSBs of p (i.e., 0.0195 proportion of $\log_2 N$) is known, then $\beta = 0.5 - 0.0195 = 0.4805$. In this case 20 many MSBs of p is required to be guessed. Using our Theorem 2, the value of γ becomes 0.26813. Thus the requirement of MSBs of d to be known is $(0.35 - 0.26813) \times 1024 = 84$ bits.

One should note that the total requirement of bits to be known in our case is $84 + 20 = 104$, which is less than the requirement of 106 bits in [10]. The number of MSBs of d to be exposed in [10] is $(\delta - \gamma_1)\log_2 N$ (we denote γ by γ_1 here). In our case, the requirement of MSBs in p is $(0.5 - \beta)\log_2 N$ and that of d is $(\delta - \gamma_2)\log_2 N$ (we denote γ by γ_2 here), and adding them we get the total requirement of MSBs (considering both p, d) is $(0.5 - \beta + \delta - \gamma_2)\log_2 N$. One may check that $(\delta - \gamma_1)\log_2 N$ of [10] is greater than $(0.5 - \beta + \delta - \gamma_2)\log_2 N$ when $\beta < \frac{1}{2}$. This theoretically justifies the advantage of our technique.

Based on Theorem 2, we get the following probabilistic polynomial time algorithm.

As we will work with low lattice dimensions, the actual requirement of MSBs to be known will be higher in experimental results than the numerical values arrived from the theoretical results. Let us first present an example with all the relevant data that highlights our improvement.

Example 1. We consider 1024 bits N, where p, q are as follows:
125076192352751041131507009460095319151891488205387463013857272133794535733443372033786891784694556227753494467523090187993837113578541321880095737053207 99, and
110791215693704761804913407298464219271673668591116468423029324683331660038391674471106817478734147986480207404489676435380576442892517619070138864997993 83.

The public encryption exponent e and the private decryption exponent $d(> N^{0.3})$ are as follows:
41114195314827033022131522152498201993652976103174529855585727679733063464769115345985695600033379618093485626368069580331701437171399103541158583303509793517930633496883835424622296561497709443871759791207393279618329492446932621470954494041615618545237490828036465397182668742616838575576909861473509095701, and
911260046070098225464230311775052873569746472764337803805303583934395253129269343722635765941.

Algorithm 1.

Inputs: N, e; MSBs of d, p, i.e., d_0, p_0; Parameters γ, β, δ.

Steps:

```
0. If $\gamma \not\lesssim 1 - \frac{\beta+2\sqrt{\beta(\beta+3\delta)}}{3}$ then exit with failure.
1. Construct polynomial $f_{MSB1} = ex - (N - p_0 - q_0)y + yz + R$ where
        $q_0 = \frac{N}{_0}$, and $R = ed_0 - 1$.
2. Initialize $X = N^{\gamma}, Y = N^{\delta}, Z = N^{\beta}$.
3. Fix the lattice parameters $m, t$.
4. Calculate $W = ||f_{MSB1}(xX, yY, zZ)||_{\infty}$ and $n = (XY) \quad Z \quad ^{+} W$.
5. Construct $f \equiv R^{-1} f_{MSB1}(x, y, z) \bmod n \equiv 1 + ax + by + cyz$.
6. Construct the lattice $L$ from $f$, i.e., with the coefficients of the shift
        polynomials $g_i \quad (xX, yY, zZ)$, $h_i \quad (xX, yY, zZ)$, $g'_i \quad (xX, yY, zZ)$
        and $h'_i \quad (xX, yY, zZ)$, where $g, h, g', h'$ are constructed from $f$.
7. Reduce $L$ using LLL algorithm to get the first two elements $f_1, f_2$.
8. Calculate the resultant $G_1$ using $f_{MSB1}, f_1$
        and the resultant $G_2$ using $f_{MSB1}, f_2$.
9. If both $G_1, G_2$ are nonzero
        then calculate the resultant $G$ of $G_1, G_2$;
   else
        exit with failure.
10. If $G$ is nonzero
        then solve $G$ to get the integer root $z_0 = (p + q - p_0 - q_0 - 1)$;
    else
        exit with failure.
```

First we work with the case $m = t = 1$, i.e., getting a lattice with dimension $w = 16$ which corresponds to a 16×16 matrix (one may refer to [10, Section 4.1.1, Page 378] for the exact matrix). Factoring N requires the knowledge of 112 many MSBs of d using the method of [10], whereas, our technique requires 80 many MSBs of d and 21 many MSBs of p. Both the techniques require around 1.5 seconds on our platform. Following the idea of [22], around 7 MSBs of p may be known in polynomial time and hence we need 2^{21-7} many guesses for p, which requires less than 7 hours in our experimental set-up. The existing works on partial key exposure attacks will not work with the knowledge of only 80 bits of MSBs that we achieve here.

Considering a higher lattice dimension, $m = t = 2$, i.e., $w = 40$, factoring N requires knowledge of 110 many MSBs of d using the idea of [10]. This requires 53.03 seconds. According to experimental results in [10, Figure 5], this should require around 93 MSBs of d. In our case, we require only 53 MSBs of d and 21 MSBs of p to factor N that requires 46.25 seconds; thus the total requirement is $53 + 21 = 74$ many bits. Considering that 7 many MSBs of p may be known using the idea of [22], the overall attack will take a day in a cluster of 9 machines.

One may also consider guessing MSBs of $p + q$ rather than p as the polynomial f_{MSB1} deals with $p + q$ rather than p and q. Experimental results of [22] show that around 12 many MSBs of $p + q$ can be estimated correctly for the 1024-bit N, whereas the estimation gives around 7 many MSBs for p. Consider that b_1

Table 1. Our results for 1024 bits N with lattice dimension $m = 1, t = 1$, i.e., $w = 16$

308-bit d and # MSBs of d revealed in our case is 80 bits										
# MSBs of d [10]	112	112	107	111	122	114	115	114	113	113
# MSBs of p (our)	21	22	26	27	33	20	23	27	24	17

359-bit d and # MSBs of d revealed in our case is 150 bits										
# MSBs of d [10]	213	213	224	221	210	213	213	209	214	209
# MSBs of p (our)	55	58	64	63	56	58	58	60	57	64

many MSBs of p are known (p is estimated by p') and we estimate q by $q' = \lfloor \frac{N}{p'} \rfloor$. Further, let us assume that the estimation $p' + q'$ has b_2 many MSBs identical with the exact value $p + q$. Then experimentally we observed that $b_2 > b_1$ in general and for $b_1 = 7$, we get $b_2 = 12$ on an average. This shows that the effect of guessing the MSBs of p or $p + q$ are same.

In Table 1, we consider different 1024 bits N and present the results of 10 runs of Algorithm 1 for two cases, one when $d > N^{0.3}$ (308-bit d) and the other when $d > N^{0.35}$ (359-bit d). Let MSB_d, MSB_p be the number of MSBs exposed in d, p respectively and b_d, b_N be the number of bits in d, N respectively. For the experiments, we have taken $X = 2^{b_d - MSB_d - \tau} + 3$, $Y = 2^{b_d - \tau} + 3$ and $Z = 2^{\frac{b_N}{2} - 1 - MSB_p - \tau} + 3$, where τ is assigned to either 0 or 1. As already discussed, $m = 1, t = 1$, i.e., $w = 16$.

First, we consider that only 80 MSBs of d will be leaked and studied the requirement of the MSBs of p for the attack. In each case, the algorithm of [10] has also been executed and the requirement of the minimum number of MSBs for d is presented. Next, we consider that 150 MSBs of d will be exposed for our attack. The results of Table 1 clearly identifies the improvement through our approach over the idea of [10].

2.2 Method II

We start this section with the following theorem.

Theorem 3. *Let $d \leq N^{\delta}$ and consider that d_0, p_0 are exposed such that $|d - d_0| < N^{\gamma}$ and $|p - p_0| < N^{\beta}$. Then one can factor N (in probabilistic polynomial time) when*

$$\gamma \leq 1 + \frac{1}{3}\lambda - \beta - \frac{2}{3}\sqrt{\lambda}\sqrt{\lambda + 3\beta},$$

where $\lambda = max\{\gamma, \delta - \frac{1}{2}\}$.

Proof. Note that the attacker can compute $k_0 = \lfloor \frac{ed_0 - 1}{N} \rfloor$. Let $k_1 = k - k_0$, the unknown part of k. It can be shown similar to [2] that $|k_1| < \frac{e}{\phi(N)}(N^{\gamma} + 3N^{\delta - \frac{1}{2}})$. So we get $|k_1| < 4N^{\lambda}$, where $\lambda = \max\{\gamma, \delta - \frac{1}{2}\}$.

Now, $ed - 1 = k(N + 1 - p - q) \Leftrightarrow e(d_0 + d_1) - 1 = (k_0 + k_1)(N - (p + q - 1)) \Leftrightarrow$ $e(d_0 + d_1) - 1 = (k_0 + k_1)(N - p_0 - q_0 - (p + q - p_0 - q_0 - 1)) \Leftrightarrow ed_1 - (N - p_0 - q_0)$

$k_1 + k_1(p+q-p_0-q_0-1) + k_0(p+q-p_0-q_0-1) + ed_0 - 1 - (N-p_0-q_0)k_0 = 0$. Hence we have to find the solution of the polynomial

$$f_{MSB2}(x,y,z) = ex - (N-p_0-q_0)y + yz + k_0 z + R,$$

where $R = ed_0 - 1 - (N-p_0-q_0)k_0$. That is, the root of $f_{MSB2}(x,y,z)$ is $(x_0,y_0,z_0) = (d_1, k_1, p+q-p_0-q_0-1)$.

Let $X = N^{\gamma}, Y = N^{\lambda}, Z = N^{\beta}$, and one can check that they are the upper bounds of x_0, y_0, z_0 neglecting the small constant multipliers.

Let us fix the lattice parameters m, t. Define $W = ||f_{MSB2}(xX, yY, zZ)||_{\infty}$ and $n = X^m Y^{m+t} Z^m W$. In order to work with a polynomial with constant term 1, we define

$$f \equiv R^{-1} f_{MSB2}(x,y,z) \bmod n \equiv 1 + ax + by + cyz + dz.$$

(During the experiments, as long as gcd of R, n is not 1, we keep on increasing n by 1.) Then we use the shifts

$$\begin{aligned} g_{ijk} &= x^i y^j z^k f(x,y,z) X^{m-i} Y^{m+t-j} Z^{m-k}, \\ &\quad for\ i = 0,\ldots,m;\ j = 0,\ldots,m-i;\ k = 0,\ldots,m-i; \end{aligned}$$

$$\begin{aligned} h_{ijk} &= x^i y^j z^k f(x,y,z) X^{m-i} Y^{m+t-j} Z^{m-k}, \\ &\quad for\ i = 0,\ldots,m;\ j = m-i+1,\ldots,m-i+t;\ k = 0,\ldots,m-i; \end{aligned}$$

$$\begin{aligned} g'_{ijk} &= n x^i y^j z^k, \\ &\quad for\ i = 0,\ldots,m+1;\ j = 0,\ldots,m+t+1-i;\ k = m+1-i; \end{aligned}$$

$$\begin{aligned} h'_{ijk} &= n x^i y^j z^k, \\ &\quad for\ i = 0,\ldots,m;\ j = m+t+1-i;\ k = 0,\ldots,m-i. \end{aligned}$$

Now we build a lattice L with the basis elements coming from the coefficient vectors of $g_{ijk}(xX,yY,zZ), h_{ijk}(xX,yY,zZ), g'_{ijk}(xX,yY,zZ)$ and $h'_{ijk}(xX,yY,zZ)$ following the idea of [10]. The vectors are ordered in such a manner that the matrix corresponding to the lattice L becomes triangular, and the diagonal entries of g and h are equal to $X^m Y^{m+t} Z^m$. Now we follow the similar computation as in [10, Appendix B], taking $t = \tau m$. If

$$X^{2+3\tau} Y^{3+6\tau+3\tau^2} Z^{3+3\tau} \leq W^{2+3\tau}, \tag{2}$$

we get polynomials f_1 and f_2 (the first two elements after lattice reduction using LLL algorithm) that satisfy the Howgrave-Graham bound as described in Theorem 1.

Now we construct two resultants G_1, G_2 taking two different pairs from f_{MSB2}, f_1, f_2 (in our experiments, mostly, G_1 is constructed using f_{MSB2}, f_1 and G_2 is constructed using f_{MSB2}, f_2). Then we construct the resultant of G_1, G_2 to get G. The integer root of G provide z_0, which in turn gives the prime.

We assume that the resultant computations for multivariate polynomials constructed in our approach yield non-zero polynomials. This is successful in most of the cases in our experiment. However, as this step involves some probability of success, we consider the algorithm as probabilistic. As each of lattice reduction, resultant computation and root finding is polynomial time algorithm in $\log_2 N$, the product N can be factored in probabilistic polynomial time given the constraints in this theorem.

Here $X = N^{\gamma}, Y = N^{\lambda}, Z = N^{\beta}$, and

$$W = \max(eX, (N - p_0 - q_0)Y, YZ, k_0 Z, R) \geq (N - p_0 - q_0)Y \approx NY = N^{1+\lambda}.$$

So the Inequality 2 holds if,

$$\begin{aligned} X^{2+3\tau} Y^{3+6\tau+3\tau^2} Z^{3+3\tau} &\leq (NY)^{2+3\tau} \ \Leftrightarrow \\ N^{\gamma(2+3\tau)} N^{\lambda(3+6\tau+3\tau^2)} N^{\beta(3+3\tau)} &\leq N^{(1+\lambda)(2+3\tau)} \ \Leftrightarrow \\ 3\lambda\tau^2 + (3\beta + 3\gamma + 3\lambda - 3)\tau + (2\gamma + \lambda + 3\beta - 2) \leq 0 & \end{aligned}$$

Putting the optimal value of τ, which is $\tau = \frac{1-\beta-\gamma-\lambda}{2\lambda}$, in the above inequality we get the required condition $\gamma \leq \frac{(6+2\lambda-6\beta)-\sqrt{16\lambda^2+48\lambda\beta}}{6}$. □

Putting $\beta = \frac{1}{2}$ in Theorem 3, we get the same bound as in [10, Theorem 1]. As we have the knowledge of a few MSBs of p, the value of β decreases below $\frac{1}{2}$ in our case, increasing the value of γ. As $\delta - \gamma$ proportion of bits of d needs to be known for the attack, we require less number of MSBs of d to be exposed than [10]. Similar to Algorithm 1 corresponding to Theorem 2, one can devise a probabilistic polynomial time algorithm following Theorem 3.

Let us now present an example with all the relevant data that highlights our improvement.

Example 2. We consider 1024 bits N, where p, q are as follows:
129009549990053752073859201814163564189023684680391501151338376702098744712580162829362111710263879758520746505779736380616669758756082522934769465036431531 and
100018509329865935646436400634421440180345180969932799051114353462459764015419519476055271010012194150583838878020173194022682316782601191836891187015992 91.

The public encryption exponent e and the private decryption exponent $d(> N^{0.635})$ are as follows:
26464279449637052908320010402643210645183306440142727819011766921275747995184991062700504366357036237348582610659452376574441390684860427257433960292828065723745795366302145165594304294557845010241961636348596529237538193077131072546681188380145244844079755319955227511927745024777291417353383785591531787203 and
7161023303467486069671927956706449459095092348532240745792204228848640890584976007853666974474085220376561849594212667546760685105870728672799323285469369900587950978784699041414105582850665589707.

First we work with the case $m = t = 1$, i.e., lattice dimension $w = 20$. Factoring N requires the knowledge of 572 many MSBs of d using the method of [10], whereas, our technique requires 517 many MSBs of d and 31 many MSBs of p. Both the techniques require around 7.5 seconds on our platform. Following the idea of [22], around 7 MSBs of p may be known in polynomial time and hence we need 2^{31-7} many guesses for p, which requires around a day in a cluster of 2^{10} machines. The existing works on partial key exposure attacks will not work with the knowledge of only 517 bits of MSBs that we achieve here. Further the total requirement of unknown bits in our case is $517 + 31 = 548$ which is less than 572.

With higher lattice dimension, $m = t = 2$, i.e., $w = 50$, factoring N requires 527 many MSBs of d using the idea of [10]. This takes 859.64 seconds. In our case, it is enough to know 494 MSBs of d with 31 MSBs of p. The time required is 887.22 seconds.

We now present the experimental details of 10 runs with 10 different 1024 bits N in Table 2. We consider that only 517 many MSBs of d will be leaked and then study the requirement of the MSBs of p for our attack. In each case, the algorithm of [10] has also been executed and the requirement of the minimum number of MSBs for d is presented. The results of Table 2 clearly identifies the improvement through our approach over the idea of [10].

Table 2. Our results for 1024 bits N with lattice dimension $m = 1, t = 1$, i.e., $w = 20$

651-bit d and # MSBs of d revealed in our case is 517 bits										
# MSBs of d [10]	572	573	572	573	573	571	570	569	578	575
# MSBs of p (our)	31	34	35	35	35	32	38	33	33	35

2.3 Comparison of Methods I and II

Let us first concentrate on Theorem 3. We get $\gamma \leq 1 + \frac{1}{3}\lambda - \beta - \frac{2}{3}\sqrt{\lambda}\sqrt{\lambda + 3\beta}$, where $\lambda = max\{\gamma, \delta - \frac{1}{2}\}$.

Now $\lambda = \gamma$ implies that $\gamma \leq \frac{3(1-\beta)^2}{4}$. This bound of γ is valid when $\delta - \frac{1}{2} \leq \gamma$, i.e., when $\delta \leq \frac{1}{2} + \frac{3(1-\beta)^2}{4}$.

If $\lambda = \delta - \frac{1}{2}$, we get that $\gamma \leq \frac{5}{6} - \beta + \frac{\delta}{3} - \frac{1}{3}\sqrt{4\delta^2 - 4\delta + 1 + 12\beta\delta - 6\beta}$. We consider this bound for $\delta > \frac{1}{2} + \frac{3(1-\beta)^2}{4}$.

We need $(\delta - \gamma)\log_2 N$ many MSBs of d to factor N and thus when the upper bound of γ is larger, one gets the better result. Thus from Theorem 3, we get,

1. $\gamma \leq \frac{3(1-\beta)^2}{4}$, when $\delta \leq \frac{1}{2} + \frac{3(1-\beta)^2}{4}$;
2. $\gamma \leq \frac{5}{6} - \beta + \frac{\delta}{3} - \frac{1}{3}\sqrt{4\delta^2 - 4\delta + 1 + 12\beta\delta - 6\beta}$, when $\delta > \frac{1}{2} + \frac{3(1-\beta)^2}{4}$.

Now we compare the item 1 above with the bound of Theorem 2. In Theorem 2, we have $\gamma \leq 1 - \frac{\beta + 2\sqrt{\beta(\beta + 3\delta)}}{3}$.

Note that $\frac{3(1-\beta)^2}{4} \leq 1 - \frac{\beta + 2\sqrt{\beta(\beta+3\delta)}}{3}$ iff $\delta \leq \frac{1}{12\beta}(\frac{81}{16}\beta^4 - \frac{63}{4}\beta^3 + \frac{39}{8}\beta^2 + \frac{21}{4}\beta + \frac{9}{16})$. One can check $\frac{1}{12\beta}(\frac{81}{16}\beta^4 - \frac{63}{4}\beta^3 + \frac{39}{8}\beta^2 + \frac{21}{4}\beta + \frac{9}{16}) \leq \frac{1}{2} + \frac{3(1-\beta)^2}{4}$, for $\beta > 0.07$. In our analysis, we generally consider β is very close to 0.5, i.e., $\beta > 0.07$. Thus for the required range of values for β, we have, $\frac{1}{12\beta}(\frac{81}{16}\beta^4 - \frac{63}{4}\beta^3 + \frac{39}{8}\beta^2 + \frac{21}{4}\beta + \frac{9}{16}) \leq \frac{1}{2} + \frac{3(1-\beta)^2}{4}$.

Hence, we can conclude that Method I (corresponding to Theorem 2) is more effective when $\delta \leq \frac{1}{12\beta}(\frac{81}{16}\beta^4 - \frac{63}{4}\beta^3 + \frac{39}{8}\beta^2 + \frac{21}{4}\beta + \frac{9}{16})$, but for higher values of δ, Method II (corresponding to Theorem 3) will perform better.

3 LSBs of d and MSBs of p Known

In [10, Theorem 3], the cryptanalysis of RSA has been studied when some LSBs of d are exposed. We here extend the idea with the additional idea that a few MSBs of p are also known. This gives the following theorem. We present the proof briefly as the technique is similar to our Theorem 2.

Theorem 4. *Let $d < N^\delta$. Given $(\delta - \gamma)\log_2 N$ many LSBs of d and p_0 when $|p - p_0| < N^\beta$, N can be factored in probabilistic polynomial time when*

$$\gamma \leq 1 - \frac{\beta + 2\sqrt{\beta(\beta + 3\delta)}}{3}.$$

Proof. Consider that d_0 is the integer corresponding to the exposed LSBs of d. Thus, $d_0 \equiv d \bmod M$ for some M, i.e., $d = d_0 + d_1 M$, for some d_1. Now we have $ed - 1 = k(N - (p + q - 1))$, which can be written as $e(d_0 + d_1 M) - 1 = k(N - p_0 - q_0 - (p + q - p_0 - q_0 - 1)) \Leftrightarrow eMd_1 - (N - p_0 - q_0)k + k(p + q - p_0 - q_0 - 1) + ed_0 - 1 = 0$. Hence we have to find the solution of the polynomial

$$f_{LSB}(x, y, z) = eMx - (N - p_0 - q_0)y + yz + R,$$

where $R = ed_0 - 1$. So, the root of $f_{LSB}(x, y, z)$ is $(x_0, y_0, z_0) = (d_1, k, p + q - p_0 - q_0 - 1)$. This polynomial is same as the polynomial f_{MSB1} in the proof of Theorem 2. Thus, using similar analysis as in the proof of Theorem 2, we get the constraint as

$$X^{1+3\tau} Y^{2+3\tau} Z^{1+3\tau+3\tau^2} \leq W^{1+3\tau}.$$

Putting $X = N^\gamma, Y = N^\delta, Z = N^\beta$ we get $\gamma \leq 1 - \frac{\beta + 2\sqrt{\beta(\beta+3\delta)}}{3}$. □

Putting $\beta = \frac{1}{2}$ in Theorem 4, we get the same bound as in [10, Theorem 3]. As we have the knowledge of a few MSBs of p, the value of β decreases below $\frac{1}{2}$ in our case, increasing the value of γ. As $\delta - \gamma$ proportion of bits of d needs to be known for the attack, we require less number of LSBs of d to be exposed than [10]. Similar to Algorithm 1 corresponding to Theorem 2, one can devise a probabilistic polynomial time algorithm following Theorem 4.

Let us now present an example with all the relevant data that highlights our improvement.

Table 3. Our results for 1024 bits N with lattice dimension $m = 1, t = 1$, i.e., $w = 16$

308-bit d and # LSBs of d revealed in our case is 80 bits										
# LSBs of d [10]	115	107	105	108	109	109	114	116	112	108
# MSBs of p (our)	23	24	23	29	24	27	30	27	20	19

Example 3. We consider 1024 bits N, where p, q are as follows:
120345555452049651309296431229078115451502115011463732197427366O
4036604551051432401698923375314223219352776116668992562953977601
494812370217390511745064609 and
1170162232428076043275963242092394902992044041699922765182745491
1687794587069471939459107891700953238765852825589195765523177221
061363437357581056385345193.

The public encryption exponent e and the private decryption exponent $d(> N^{0.30})$ are as follows:
9262840848832818099725923231290910682284377479861057935159238392
2152908007127148216664565531845550317794995167278441598392908149
4300715331067535008047871523708599866902351068839273181735190226
3333864097908955752096238221073594906199364950641439860998004693
1029715538636463760752793958294478936586780899434369 and
5009727027589508051673544277436160282160739874039432019366401679
69825484681181534595620036481.

First we work with the case $m = t = 1$, i.e., lattice dimension $w = 16$. Factoring N requires the knowledge of 115 many LSBs of d using the method of [10], whereas, our technique requires 80 many LSBs of d and 23 many MSBs of p. Both the techniques requires little less than 1.5 seconds on our platform. Following the idea of [22], around 7 MSBs of p may be known in polynomial time and hence we need 2^{23-7} many guesses for p, which requires a day in our experimental set-up.

When we work with higher lattice dimension $m = t = 2$, i.e., $w = 40$, factoring N requires 112 LSBs of d using the idea of [10]. It takes 46.39 seconds. In our case, we need 48 LSBs of d with 25 MSBs of p (requires 38.21 seconds) or 62 LSBs of d with 23 MSBs of p (requires 39.41 seconds).

We now present the experimental details of 10 runs in Table 3 considering 10 different 1024 bits N. We consider that only 80 many LSBs of d will be leaked and then study the requirement of the LSBs of p for the attack. In each case, the algorithm of [10] has also been executed and the requirement of the minimum number of LSBs for [10] is presented. The results of Table 3 clearly identifies the improvement through our approach over the idea of [10].

4 Conclusion

In this paper we have studied cryptanalysis of RSA when either certain amount of MSBs or certain amount of LSBs of d are exposed. Our additional idea is to guess a few MSBs of the secret prime p. With this additional information, we find that our technique is more efficient than that of [10] (where no guess on the bits of p is attempted) in terms of the amount of bits of d to be exposed. Our technique is also better if one considers total number of bits to be known from d, p together than that of d only in [10]. Our theoretical results are implemented and we present experimental evidences of 1024 bits N, that can be factored with the exposure of considerably less amount of bits in d than [10] with a guess of a few MSBs in p that can be searched exhaustively (say around 20 to 30 bits).

Acknowledgments. The authors like to thank the anonymous reviewers for detailed comments that improved the technical as well as editorial quality of this paper. The first author likes to acknowledge the Council of Scientific and Industrial Research (CSIR), India for supporting his research fellowship.

References

1. Blömer, J., May, A.: Low secret exponent RSA revisited. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 4–19. Springer, Heidelberg (2001)
2. Blömer, J., May, A.: New partial key exposure attacks on RSA. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 27–43. Springer, Heidelberg (2003)
3. Blömer, J., May, A.: A generalized wiener attack on RSA. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 1–13. Springer, Heidelberg (2004)
4. Boneh, D.: Twenty Years of Attacks on the RSA Cryptosystem. Notices of the AMS 46(2), 203–213 (1999)
5. Boneh, D., Durfee, G.: Cryptanalysis of RSA with private key d less than $N^{0.292}$. IEEE Trans. on Information Theory 46(4), 1339–1349 (2000)
6. Boneh, D., Durfee, G., Frankel, Y.: Exposing an RSA Private Key Given a Small Fraction of its Bits. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 25–34. Springer, Heidelberg (1998)
7. Boneh, D., DeMillo, R., Lipton, R.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
8. Coppersmith, D.: Small solutions to polynomial equations and low exponent vulnerabilities. Journal of Cryptology 10(4), 223–260 (1997)
9. Duejella, A.: Continued fractions and RSA with small secret exponent. Tatra Mt. Math. Publ. 29, 101–112 (2004)
10. Ernst, M., Jochemsz, E., May, A., de Weger, B.: Partial key exposure attacks on RSA up to full size exponents. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 371–386. Springer, Heidelberg (2005)
11. Hastad, J.: On using RSA with low exponent in public key network. In: Advances in Cryplogy-CRYPTO 1985 Proceedings. LNCS, pp. 403–408. Springer, Heidelberg (1985)
12. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997)
13. Jochemsz, E.: Cryptanalysis of RSA variants using small roots of polynomials. Ph. D. thesis, Technische Universiteit Eindhoven (2007)
14. Jochemsz, E., May, A.: A Polynomial Time Attack on RSA with Private CRT-Exponents Smaller Than $N^{0.073}$. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 395–411. Springer, Heidelberg (2007)

15. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
16. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
17. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261, 513–534 (1982)
18. Pollard, J.M.: Theorems on factorization and primality testing. Proc. of Combridge Philos. Soc. 76, 521–528 (1974)
19. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. Communications of ACM 21(2), 158–164 (1978)
20. Stinson, D.R.: Cryptography – Theory and Practice, 2nd edn. Chapman & Hall/CRC, Boca Raton (2002)
21. Steinfeld, R., Contini, S., Pieprzyk, J., Wang, H.: Converse results to the Wiener attack on RSA. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 184–198. Springer, Heidelberg (2005)
22. Sun, H.-M., Wu, M.-E., Chen, Y.-H.: Estimating the prime-factors of an RSA modulus and an extension of the wiener attack. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 116–128. Springer, Heidelberg (2007)
23. Verheul, E.R., van Tilborg, H.C.A.: Cryptanalysis of 'less short' RSA secret exponents. Applicable Algebra in Engineering, Communication and Computing 8, 425–435 (1997)
24. Wiener, M.: Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory 36(3), 553–558 (1990)
25. Williams, H.C.: A $p+1$ method of factoring. Mathematics of Computation 39(159), 225–234 (1982)
26. de Weger, B.: Cryptanalysis of RSA with small prime difference. Applicable Algebra in Engineering, Communication and Computing 13(1), 17–28 (2002)

Simple Algorithms for Computing a Sequence of 2-Isogenies

Reo Yoshida[1] and Katsuyuki Takashima[1,2]

[1] Graduate School of Informatics, Kyoto University,
36-1 Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501 Japan
yoshida@ai.soc.i.kyoto-u.ac.jp
[2] Mitsubishi Electric, 5-1-1 Ofuna, Kamakura, Kanagawa, 247-8501 Japan
Takashima.Katsuyuki@aj.MitsubishiElectric.co.jp

Abstract. Recently, some cryptographic primitives have been described that are based on the supposed hardness of finding an isogeny between two (supersingular) elliptic curves. As a part of such a primitive, Charles *et al.* proposed an algorithm for computing sequences of 2-isogenies. However, their method involves several redundant computations. We construct simple algorithms without such redundancy, based on very compact descriptions of the 2-isogenies. For that, we use some observations on 2-torsion points.

1 Introduction

Computing a sequence of isogenies of elliptic curves is a new cryptographic basic operation in some applications. For example, a cryptographic hash function from expander graphs, proposed in [CGL07], consists of computing an isogeny sequence. Moreover, there exists an attempt [RS06] to construct a new type of public key cryptosystems using such operations. The security of these applications is based on the supposed hardness of finding an isogeny between two elliptic curves. To solve such a problem with a quantum computer seems hard [RS06]. Therefore, the above applications are considered as candidates for post-quantum cryptosystems.

Previously, isogenies between elliptic curves were used in a crucial way for calculating the cardinality of an elliptic curve over a finite field (see [CF06], for example). However, in the above, computation of a sequence of isogenies is used in cryptographic operations, e.g., hashing and encryption/decryption, not just for some offline operations, e.g., domain (or system) parameter generation. Therefore, we need to improve the efficiency of this computation to implement new cryptographic applications.

Charles, Goren and Lauter [CGL07] proposed an algorithm for computing a sequence of 2-isogenies. Their method is based on Vélu's formulas, which computes a 2-isogeny explicitly for that purpose. By using *supersingular* elliptic curves over $\mathbb{F}_{p^2}$, all computations can stay in $\mathbb{F}_{p^2}$, the quadratic extension of a prime finite field (see Section 3.4). It makes the sequence computation a reasonable cryptographic operation.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 52–65, 2009.

We show two simple methods based on concise expressions using non-trivial observations in Section 5. The first method is based on the new recurrence relation (18) between a coefficient and a 2-torsion point of elliptic curves. The second method only computes 2-torsion points of elliptic curves on the sequence. Both methods have costs of one multiplication and square root computation of a field for each 2-isogeny. Moreover, we append another simple method that was communicated to the authors from an anonymous referee.

In Section 2, we review facts related to elliptic curves. In Section 3, we prepare notations and facts for 2-isogeny computation. In Section 4, we give the algorithm of [CGL07]. In Section 5, we propose algorithms for a sequence of 2-isogenies. In Section 6, we describe a related method told by an anonymous referee. In Section 7, we compare the costs of the proposed methods with that of the existing one.

2 2-Isogenies between Elliptic Curves

We summarize facts about elliptic curves in this section. For details, see [S86], for example.

2.1 Elliptic Curves

Let p be a prime greater than 3 and $\mathbb{F}_p$ be the finite field with p elements. Let $\overline{\mathbb{F}}_p$ be its algebraic closure. An elliptic curve E over $\overline{\mathbb{F}}_p$ is given by the Weierstrass normal form

$$E : y^2 = x^3 + Ax + B \tag{1}$$

for A and $B \in \overline{\mathbb{F}}_p$ where the discriminant of the RHS of (1) is non-zero. We denote the point at infinity on E by $\mathcal{O}_E$. Elliptic curves are endowed with a unique algebraic group structure, with $\mathcal{O}_E$ as neutral element. The j-invariant of E is $j(A, B) = 1728\frac{4A^3}{4A^3+27B^2}$. Conversely, for $j \neq 0, 1728 \in \overline{\mathbb{F}}_p$, set $A = A(j) = \frac{3j}{1728-j}$, $B = B(j) = \frac{2j}{1728-j}$. Then, the obtained E in (1) has j-invariant j. Two elliptic curves over $\overline{\mathbb{F}}_p$ are isomorphic if and only if they have the same j-invariant. For a positive integer n, the set of n-torsion points of E is $E[n] = \{P \in E(\overline{\mathbb{F}}_p) \,|\, nP = \mathcal{O}_E\}$.

Given two elliptic curves E and E' over $\overline{\mathbb{F}}_p$, a homomorphism $\phi : E \to E'$ is a morphism of algebraic curves that sends $\mathcal{O}_E$ to $\mathcal{O}_{E'}$. A non-zero homomorphism is called an isogeny, and a separable isogeny with the cardinality ℓ of the kernel is called ℓ-isogeny.

An elliptic curve E over $\overline{\mathbb{F}}_p$ is called supersingular if there are no points of order p, i.e., $E[p] = \{\mathcal{O}_E\}$. The j-invariants of supersingular elliptic curves lie in $\mathbb{F}_{p^2}$ (see [S86, Chap. V, Th. 3.1]).

2.2 Vélu's Formulas

We compute the 2-isogeny by using Vélu's formulas. Vélu gave in [V71] the explicit formulas of the isogeny $\phi : E \to E'$ and the equation of the form (1) of E' when E is given by (1) and $C = \ker\phi$ is explicitly given. It is for general degree ℓ. We consider only the degree $\ell = 2$ case in this paper. Let $C = \langle(\alpha, 0)\rangle$ be a subgroup of order 2 on an elliptic curve E. Then there exists a unique isogeny $\phi_\alpha : E \to E'$ s.t. $C = \ker\phi_\alpha$, and we denote E' by E/C. Note that for a *general* (i.e., ordinary) elliptic curve, α may be defined over a quadratic or cubic extension field only. However, in our computations all the 2-torsion points will always be defined over $\mathbb{F}_{p^2}$, see Section 3.4.

For a subgroup $C = \langle(\alpha, 0)\rangle \subset E[2]$, the elliptic curve E/C is given by the equation

$$y^2 = x^3 - (4A + 15\alpha^2)x + (8B - 14\alpha^3). \tag{2}$$

Therefore, E/C is also defined over $\mathbb{F}_{p^2}$ when E is supersingular. Moreover, the isogeny ϕ_α: $E \to E/C$ is given by

$$\phi_\alpha : (x, y) \mapsto \left(x + \frac{(3\alpha^2 + A)}{x - \alpha}, y - \frac{(3\alpha^2 + A)y}{(x - \alpha)^2}\right), \tag{3}$$

$\phi_\alpha(\mathcal{O}_E) = \mathcal{O}_{E/C}$ and $\phi_\alpha((\alpha, 0)) = \mathcal{O}_{E/C}$. Clearly, ϕ_α is also defined over $\mathbb{F}_{p^2}$ for supersingular E.

3 A Walk on a Pizer Graph

We consider a graph consisting of 2-isogenies between supersingular elliptic curves. The graph has an expanding property (expander graph), and is called a Pizer graph [P90, CGL07].

3.1 Expander Graph

Let $G = (\mathcal{V}, \mathcal{E})$ be a graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. We will deal with undirected graphs, and say a graph is k-regular if each vertex has k edges coming out of it. A graph G is an expander graph with expansion constant $c > 0$ if, for any subset $\mathcal{U} \subset \mathcal{V}$ s.t. $|\mathcal{U}| \le \frac{|\mathcal{V}|}{2}$, then $|\Gamma(\mathcal{U})| \ge c\,|\mathcal{U}|$ where $\Gamma(\mathcal{U})$ is the boundary of $\mathcal{U}$ (which is all neighbors of $\mathcal{U}$ minus all elements of $\mathcal{U}$). Any expander graph is connected. A random walk on an expander graph has rapidly mixing property. After $O(\log(|\mathcal{V}|))$ steps, the last point of the random walk approximates the uniform distribution on $\mathcal{V}$.

Such a property is useful for cryptography. Therefore, there exist several cryptographic constructions using an expander graph ([CGL07, RS06, G00] etc.). For details of expander graphs, see [G01, HLW06].

3.2 Pizer Graph with $\ell = 2$

The Pizer graph $G = (\mathcal{V}, \mathcal{E})$ with $\ell = 2$ consists of (isomorphism classes of) supersingular elliptic curves over $\overline{\mathbb{F}}_p$ and their 2-isogenies, i.e.,

$$\mathcal{V} = \left\{\text{supersingular elliptic curve } E/\overline{\mathbb{F}}_p\right\} / \cong,$$
$$\mathcal{E} = \{([E], [E']) \mid \text{there is a 2-isogeny } \phi : E \to E'\} \subset \mathcal{V}^{(2)}$$

where $\mathcal{V}^{(2)}$ denotes the set of unordered pairs of elements of $\mathcal{V}$. This is well-defined due to the existence of a dual isogeny $\widehat{\phi}$. Hereafter, we use an abused notation $E \in \mathcal{V}$ for simple presentation.

Then, the graph G is an undirected 3-regular graph. G is known to have a rapidly mixing property. In particular, this is called a Ramanujan graph, a special type of expander graph. For details, see [P90, CGL07], for example.

3.3 A Walk without Backtracking

We consider computing a walk

$$E_0 \xrightarrow{\phi_0} E_1 \xrightarrow{\phi_1} \cdots \xrightarrow{\phi_{n-2}} E_{n-1} \xrightarrow{\phi_{n-1}} E_n \tag{4}$$

in G without backtracking, i.e., $\phi_i \neq \widehat{\phi_{i+1}}$ for $i = 0, \ldots, n-2$ and $E_i \in \mathcal{V}$. The supersingular elliptic curve

$$E_i : y^2 = f_i(x) := x^3 + A_i x + B_i \tag{5}$$

is defined over $\mathbb{F}_{p^2}$.

As the graph G is 3-regular and the walk we consider has no backtracking, we have 2 possibilities to proceed to the next vertex in $\mathcal{V}$ at $i \geq 1$. When $i = 0$, we choose 2 edges from E_0 at the beginning. Therefore, we associate a walk data $\omega = b_0 b_1 \cdots\ b_{n-1} \in \mathcal{W} = \{0,1\}^n$ with a walk (4). Then, the correspondence is bijective (see [CGL07]) as follows:

$$\mathcal{W} = \{0,1\}^n \longleftrightarrow \left\{ \begin{matrix} \text{a sequence (4) of 2-isogenies } \phi_i \text{ starting from} \\ E_0 \text{ without backtracking} \end{matrix} \right\}$$

where (4) starts from one of the edges chosen as above (see (7)). Our goal is to compute the j-invariant $j_n = j(E_n)$ from $j_0 = j(E_0)$ and a walk data $\omega \in \mathcal{W} = \{0,1\}^n$ that determines which edge is selected.

3.4 Why All Computations Stay in $\mathbb{F}_{p^2}$

First, we note that $j = 0$ (resp. $j = 1728$) is a supersingular j-invariant if and only if $p \equiv 2 \mod 3$ (resp. $p \equiv 3 \mod 4$). If we start the walk from $j_0 = 0$ (resp. $j_0 = 1728$) for such a prime p, then we use a Weierstrass model E_0 over the *prime* field $\mathbb{F}_p$, e.g., $E_0 : Y^2 = X^3 + 1$ (resp. $E_0 : Y^2 = X^3 + X$). Otherwise, if we start it from a supersingular $j_0 \neq 0, 1728$, then we use *any* Weierstrass model $E_0/\mathbb{F}_{p^2}$ with j_0. We will show that all computations remain in $\mathbb{F}_{p^2}$.

Suppose that we have an elliptic curve $E/\mathbb{F}_{p^2}$ with $\sharp E(\mathbb{F}_{p^2}) = p^2 \pm 2p + 1$. A theorem by Schoof [Sc87, Lemma 4.8] asserts that the group structure of a curve with that many points is of the form $(\mathbb{Z}/(p \pm 1)\mathbb{Z})^2$. In particular, all 2-torsion points of E are defined over $\mathbb{F}_{p^2}$, or in other words, the right hand side of a Weierstrass equation (1) for E factors over $\mathbb{F}_{p^2}$. By Vélu's formulas, this means that the three 2-isogenies leaving E are defined over $\mathbb{F}_{p^2}$, as well are the destination curves. By Tate's theorem [T66], these destination curves again have $p^2 \pm 2p + 1$ points, and by applying Schoof's theorem again, they have full 2-torsion over $\mathbb{F}_{p^2}$. Then, the above reasoning can be applied repeatedly.

This already shows that, if the starting curve $E_0/\mathbb{F}_{p^2}$ has $\sharp E_0(\mathbb{F}_{p^2}) = p^2 \pm 2p+1$ points, then all computations will stay in $\mathbb{F}_{p^2}$.

For the starting model E_0 with $j_0 = 0, 1728$, then $\sharp E_0(\mathbb{F}_p) = p + 1$, and $\sharp E_0(\mathbb{F}_{p^2}) = (p + 1)^2$, e.g., see [S86, Exercise 5.10 (b)]. Therefore, it remains to show that if we start from $j_0 \neq 0, 1728$, then any model $E_0/\mathbb{F}_{p^2}$ with j_0 will have such a number of points. We follow the argument in the proof of [AT02, Prop. 2.2]. Let π be the p^2-th power Frobenius on E_0. Because multiplication by p on E_0 is purely inseparable of degree $p^2 = \deg \pi$, it factors as $\psi \cdot \pi$ for some automorphism ψ of E_0 (see [S86, Chap. II, Cor. 2.12]). The condition that $j \neq 0, 1728$ implies $\psi = \pm 1$, hence $\pi = \pm p \in \mathrm{End}(E_0)$ and $E_0(\mathbb{F}_{p^2}) = E_0[1 - \pi](\overline{\mathbb{F}}_p) = E_0[p \pm 1](\overline{\mathbb{F}}_p) \cong (\mathbb{Z}/(p \pm 1)\mathbb{Z})^2$, in particular, $\sharp E_0(\mathbb{F}_{p^2}) = p^2 \pm 2p + 1$.

This proves our statement.

3.5 Notations for 2-Torsion Points

We fix notations of the x-coordinates of three 2-torsion points on E_i corresponding to a walk data $\omega = b_0 b_1 \cdots b_{n-1} \in \{0, 1\}^n$ as in Table 1.

In this paper, we call the map ϕ_α given by (3) the isogeny associated with α. Then, for $i \geq 0$, α_i is used as being associated with the i-th isogeny i.e.,

$$\phi_i = \phi_{\alpha_i} : E_i \to E_{i+1} = E_i/\langle(\alpha_i, 0)\rangle.$$

The 2-torsion point on E_i which induces the dual isogeny of ϕ_{i-1} on E_i is called a backtracking point, and we denote the backtracking point as $(\beta_i, 0)$ in the following.

Finally, the other 2-torsion point remains. We denote this as $(\gamma_i, 0)$.

Table 1. Notation of the x-coordinates of the 2-torsions of E_i

the point associated with the i-th isogeny ϕ_i	α_i
the i-th backtracking point	β_i
the other point on E_i	γ_i

3.6 Selector Function

We must choose the next step during a walk in G. For that purpose, we assign 1 bit $b \in \{0, 1\}$ to the 2 non-backtracking edges emanating from a vertex in $\mathcal{V}$.

First, we recall that each edge is associated with $\alpha \in \mathbb{F}_{p^2}$ (see (3)). Then we can use an order in $\mathbb{F}_{p^2}$. For example, fixing a generator τ s.t. $\mathbb{F}_{p^2} = \mathbb{F}_p[\tau] = \mathbb{F}_p + \mathbb{F}_p\tau \cong (\mathbb{F}_p)^2$, we use a natural lexicographic order of $(\mathbb{F}_p)^2$. For $\lambda_0, \lambda_1 \in \mathbb{F}_{p^2}$ and $b \in \{0, 1\}$, we define the selector function as follows:

$$\mathtt{select}(\lambda_0, \lambda_1, b) = \begin{cases} \mathtt{min}(\lambda_0, \lambda_1) \text{ if } & b = 0, \\ \mathtt{max}(\lambda_0, \lambda_1) \text{ if } & b = 1 \end{cases} \tag{6}$$

according to the above order in $\mathbb{F}_{p^2}$. As explained in Section 3.3, we take an exceptional selection $\texttt{select}_0$ at the starting vertex E_0. Since we can take any two 2-torsion points on E_0 to start a walk, we choose them in the following way. For three roots $\lambda_0, \lambda_1, \lambda_2$ of $f_0(x) = x^3 + A_0x + B_0 = 0$,

$$\texttt{select}_0(A_0, B_0, b) = (\alpha_0, \beta_0) = \begin{cases} (\texttt{mid}(\lambda_0, \lambda_1, \lambda_2), \texttt{max}(\lambda_0, \lambda_1, \lambda_2)) \text{ if } & b = 0 \\ (\texttt{max}(\lambda_0, \lambda_1, \lambda_2), \texttt{mid}(\lambda_0, \lambda_1, \lambda_2)) \text{ if } & b = 1 \end{cases} \quad (7)$$

according to the above order.

4 Charles *et al.*'s Algorithm

Charles *et al.* [CGL07] proposed an algorithm for computing a sequence (4) of 2-isogenies. It was based on Vélu's formulas (2) and (3), and the following lemma which describes the $(i+1)$-th backtracking point.

Lemma 1. *Let $(\alpha_i, 0)$, $(\beta_i, 0)$, $(\gamma_i, 0)$ be three 2-torsion points on an elliptic curve $E_i : y^2 = x^3 + A_ix + B_i$ over $\mathbb{F}_{p^2}$. Then*

$$(\beta_{i+1}, 0) = \phi_i((\beta_i, 0)) = \phi_i((\gamma_i, 0)) \quad (8)$$

where $\phi_i = \phi_{\alpha_i}$ is given by (3). That is, the $(i+1)$-th backtracking point can be calculated by using α_i, A_i, and β_i (or γ_i).

4.1 Description of Algorithm

The algorithm computes a 2-isogeny repeatedly, precisely saying, it computes $(\alpha_i, \beta_i, A_i, B_i)$ w.r.t. E_i in (4) repeatedly (Algorithm 1). That is, it consists of the iteration of Algorithm 2, i.e.,

$$\begin{aligned} & j_0 \to (A_0, B_0, \alpha_0, \beta_0) \to (A_1, B_1, \alpha_1, \beta_1) \to \cdots \\ & \to (A_{n-1}, B_{n-1}, \alpha_{n-1}, \beta_{n-1}) \to (A_n, B_n) \to j_n. \end{aligned} \quad (9)$$

The outline of Algorithm 2 is as follows:

1. Using Vélu's formulas (2), calculate the equation of the next elliptic curve E_{i+1}, i.e. (A_{i+1}, B_{i+1}).
2. Using Lemma 1 and the formula (3), calculate the backtracking point $(\beta_{i+1}, 0)$ on E_{i+1}.
3. Calculate the remaining two 2-torsion points by solving a quadratic equation.
4. According to the select bit b_{i+1}, choose α_{i+1}, which determines the next isogeny ϕ_{i+1}, from the two roots.

Algorithms 1 and 2 describe a natural one which computes the sequence (4) using Vélu's formulas (2) and (3). Algorithm 1 iterates a subroutine $\texttt{CGLIsog}$ (Algorithm 2). Additionally, $\texttt{select}_0$ and coversion functions between (A, B) and j-invariant (see Section 2.1) are used.

Algorithm 1. `CGLIsogSeq` : Computation of a sequence of 2-isogenies

```
Input : $j_0 = j(E_0)$, walk data $\omega = b_0 b_1 \dots b_{n-1}$.
Output : $j_n = j(E_n)$.
1: $(A_0, B_0) \leftarrow (A(j_0), B(j_0))$,  $(\alpha_0, \beta_0) \leftarrow \texttt{select}_0(A_0, B_0, b_0)$.
2: for $i \leftarrow 0$ to $n-2$ do
3:     $(A_{i+1}, B_{i+1}, \alpha_{i+1}, \beta_{i+1}) \leftarrow \texttt{CGLIsog}(A_i, B_i, \alpha_i, \beta_i, b_{i+1})$.
4: end for
5: $\xi \leftarrow \alpha_{n-1}^2$,  $A_n \leftarrow -(4A_{n-1} + 15\xi)$, $B_n \leftarrow 8B_{n-1} - 14\alpha_{n-1}\xi$,  $j_n \leftarrow j(A_n, B_n)$.
6: return $j_n$.
```

Algorithm 2. `CGLIsog` : Computation of a 2-isogeny

```
Input : $A_i$, $B_i$, $\alpha_i$, $\beta_i$, select bit $b_{i+1}$.
Output : $A_{i+1}$, $B_{i+1}$, $\alpha_{i+1}$, $\beta_{i+1}$.
1: $\xi \leftarrow \alpha_i^2$.
2: $A_{i+1} \leftarrow -(4A_i + 15\xi)$, $B_{i+1} \leftarrow 8B_i - 14\alpha_i\xi$.        /* by Vélu's formula (2) */
3: $\zeta \leftarrow 3\xi + A_i$, $\beta_{i+1} \leftarrow \beta_i + \frac{\zeta}{\beta_i - \alpha_i}$.              /* by Lemma 1 */
4: $u' \leftarrow -\frac{\beta_{i+1}}{2}$, $v \leftarrow A_{i+1} + \beta_{i+1}^2$, $\zeta' \leftarrow (u')^2 - v$, $\eta' \leftarrow (\zeta')^{\frac{1}{2}}$.
5: $\lambda_0 \leftarrow u' + \eta'$, $\lambda_1 \leftarrow u' - \eta'$.
6: $\alpha_{i+1} \leftarrow \texttt{select}(\lambda_0, \lambda_1, b_{i+1})$.
7: return $A_{i+1}$, $B_{i+1}$, $\alpha_{i+1}$, $\beta_{i+1}$.
```

Details of Algorithm 2 are as follows: At steps 1 and 2, we calculate (A_{i+1}, B_{i+1}) from A_i, B_i, α_i using the formula (2). At step 3, we calculate the x-coordinate of the backtracking point on E_{i+1} using (8) and (3). At step 4 and 5, we calculate the x-coordinates of the 2-torsion points other than the backtracking point by solving

$$g_{i+1}(x) = x^2 + ux + v \tag{10}$$

where $u = u_{i+1} = \beta_{i+1}$, $v = v_{i+1} = A_{i+1} + \beta_{i+1}^2$. Here, $g_{i+1}(x)$ is a quadratic factor of $f_{i+1}(x) = x^3 + A_{i+1}x + B_{i+1} = (x - \beta_{i+1})g_{i+1}(x)$. At step 6, we calculate α_{i+1} according to the select bit b_{i+1} using the selector function `select` defined by (6). Then, return $(A_{i+1}, B_{i+1}, \alpha_{i+1}, \beta_{i+1})$.

Remark 1. As $B_n = -8\alpha_{n-1}A_{n-1} - 22\alpha_{n-1}^3$, and all the other B_i where $i = 0, \dots, n-1$ are not needed for `CGLIsog` (Algorithm 2), we can immediately see that the above algorithms can be simplified to compute the following chain

$$j_0 \rightarrow (A_0, \alpha_0, \beta_0) \rightarrow (A_1, \alpha_1, \beta_1) \rightarrow \cdots \rightarrow (A_{n-1}, \alpha_{n-1}, \beta_{n-1}) \rightarrow (A_n, B_n) \rightarrow j_n.$$

We will see that it can be simplified further in Section 5.3 (see the chain (11)) using a non-trivial observation (Proposition 1).

4.2 Cost Estimate of Algorithm 2

Charles *et al.* gave a rough cost estimate of their algorithm. A more careful analysis shows that it in fact involves 8 field multiplications and 1 square root computation (see Algorithm 2). Our algorithms in Section 5, need only 1 multiplication plus 1 square root for each 2-isogeny.

5 Proposed Methods

We will describe two versions of our proposed methods based on Proposition 1 and Theorem 1, which give simple formulas for 2-torsion points on E_i in (4).

The first method (Algorithms 3 and 4) in Section 5.3 is a simplification of the method of [CGL07] (Algorithms 1 and 2). That is, we show that we need only (A_i, α_i) w.r.t. E_i for computing the last E_n in the sequence (4). We compute the following chain instead of the chain (9).

$$j_0 \to (A_0, \alpha_0) \to (A_1, \alpha_1) \to \cdots \quad \to (A_{n-1}, \alpha_{n-1}) \to (A_n, B_n) \to j_n. \tag{11}$$

The second method (Algorithms 5 and 6) in Section 5.4 is a variant of the above method. It computes only the x-coordinates of three 2-torsion points $(\alpha_i, \beta_i, \gamma_i)$ on E_i repeatedly, without calculating (A_i, B_i). These are enough for computing j_n in (9). We compute the following chain instead of the chain (9).

$$j_0 \to (\alpha_0, \beta_0, \gamma_0) \to (\alpha_1, \beta_1, \gamma_1) \to \cdots \to (\alpha_{n-1}, \beta_{n-1}, \gamma_{n-1}) \to (A_n, B_n) \to j_n.$$

We will show Proposition 1 and Theorem 1 to establish these methods in Section 5.2. They give simple expressions of $(\alpha_{i+1}, \beta_{i+1}, \gamma_{i+1})$ using $(\alpha_i, \beta_i, \gamma_i)$, the previous x-coordinates of the 2-torsions. They are simplifications of Vélu's formulas for the sequence (4) of 2-isogenies. In particular, Proposition 1 which shows that $\beta_{i+1} = -2\alpha_i$ is the key to our construction. In Section 5.1, we show preliminary lemmas (Lemmas 2 and 3) for the propositions.

5.1 Basic Lemmas

In the following Lemmas, let $\alpha_i, \beta_i, \gamma_i$ be the x-coordinates of the three 2-torsion points on $E_i : y^2 = f_i(x) = x^3 + A_i x + B_i$ described in Table 1. The relations of the roots and the coefficients of $f_i(x)$ are as follows:

$$\begin{aligned} \sigma_1(\alpha_i, \beta_i, \gamma_i) &:= \alpha_i + \beta_i + \gamma_i = 0, \\ \sigma_2(\alpha_i, \beta_i, \gamma_i) &:= \alpha_i\beta_i + \beta_i\gamma_i + \gamma_i\alpha_i = A_i, \\ \sigma_3(\alpha_i, \beta_i, \gamma_i) &:= \alpha_i\beta_i\gamma_i = -B_i \end{aligned} \tag{12}$$

where σ_t is the t-th elementary symmetric polynomial for $t = 1, 2, 3$. The following Lemma 2 gives the key equation of Proposition 1 and Theorem 1.

Lemma 2

$$(\beta_i - \alpha_i)(\gamma_i - \alpha_i) = 3\alpha_i^2 + A_i. \tag{13}$$

Proof From (12), $-\alpha_i\beta_i - \beta_i^2 = \alpha_i^2 + A_i$ and $\gamma_i = -\alpha_i - \beta_i$ hold. Then, the LHS of (13) is as follows:

$$(\beta_i - \alpha_i)(\gamma_i - \alpha_i) = (\beta_i - \alpha_i)(-2\alpha_i - \beta_i) = 2\alpha_i^2 - \alpha_i\beta_i - \beta_i^2 = 3\alpha_i^2 + A_i. \quad \square$$

Lemma 3. *The x-coordinates α_{i+1} and γ_{i+1} of non-backtracking points on E_{i+1} are given using β_{i+1} and A_{i+1} as follows:*

$$\alpha_{i+1}, \gamma_{i+1} = -\frac{\beta_{i+1}}{2} \pm \left(-\frac{3\beta_{i+1}^2}{4} - A_{i+1} \right)^{\frac{1}{2}}. \tag{14}$$

Proof. Let δ be α_{i+1} or γ_{i+1}. In the following proof, we use β, A, B to denote $\beta_{i+1}, A_{i+1}, B_{i+1}$, respectively, for readability.

Then, from the equation (12) on E_{i+1}, $\beta^2\delta + \beta\delta^2 - B = 0$. Adding $-A\beta$ to both sides of the equation, and using $-A\beta - B = \beta^3$ yield the equation $\beta^3 + \beta^2\delta + \beta\delta^2 = -A\beta$. Therefore,

$$\beta(\beta^2 + \beta\delta + \delta^2 + A) = 0.$$

If $\beta \neq 0$, then $\beta^2 + \beta\delta + \delta^2 + A = 0$, and we obtain (14). This equation (14) also holds if $\beta = 0$ because E_{i+1} is given by the equation $y^2 = x(x^2 + A)$ in that case. □

5.2 Simple Formulas for 2-Torsion Points

Lemma 1 shows that the $(i+1)$-th backtracking point $(\beta_{i+1}, 0)$ can be calculated by using the i-th backtracking point. It is a key observation for the algorithm in [CGL07].

We will show that the point can be calculated much more easily in Proposition 1. The proposition is a key to our algorithms (Algorithms 3, 4 and Algorithms 5, 6). That is, the $(i+1)$-th backtracking point is given by just α_i.

Proposition 1. *The x-coordinate β_{i+1} of the backtracking point on E_{i+1} is given by*

$$\beta_{i+1} = -2\alpha_i. \tag{15}$$

Proof. From Lemma 1 and Lemma 2,

$$\beta_{i+1} = \beta_i + \frac{(3\alpha_i^2 + A_i)}{\beta_i - \alpha_i} = \beta_i + \gamma_i - \alpha_i = -2\alpha_i.$$ □

Theorem 1. *The x-coordinates α_{i+1} and γ_{i+1} of non-backtracking points on E_{i+1} are given, using α_i, β_i, γ_i, as follows:*

$$\alpha_{i+1}, \gamma_{i+1} = \alpha_i \pm 2\left[(\beta_i - \alpha_i)(\gamma_i - \alpha_i)\right]^{\frac{1}{2}} = \alpha_i \pm 2\left(3\alpha_i^2 + A_i\right)^{\frac{1}{2}}. \tag{16}$$

Proof. From Vélu's formula (2), we obtain $A_{i+1} = -(4A_i + 15\alpha_i^2)$. Using $\beta_{i+1} = -2\alpha_i$ in Lemma 3, the equation (14) is as follows:

$$\begin{aligned} \alpha_{i+1}, \gamma_{i+1} &= -\frac{-2\alpha_i}{2} \pm \left\{ -\frac{3(-2\alpha_i)^2}{4} - \left[-(4A_i + 15\alpha_i^2)\right] \right\}^{\frac{1}{2}} \\ &= \alpha_i \pm 2\left(3\alpha_i^2 + A_i\right)^{\frac{1}{2}}. \end{aligned}$$

Then, from Lemma 2, we obtain formula (16). □

Corollary 1. *$\zeta' = 4\zeta$ in Algorithm 2.*

Proof. From (16),

$$(\alpha_{i+1} - \gamma_{i+1})^2 = 16\left(3\alpha_i^2 + A_i\right). \tag{17}$$

The LHS of (17) is the discriminant of the quadratic polynomial $g_{i+1}(x)$ given by (10), and it is $4\zeta'$. The RHS of (17) is 16ζ. Therefore, (17) leads to $\zeta' = 4\zeta$ as the characteristic p of the finite field is not 2. □

5.3 Proposed Method 1

From Vélu's formula (2) and Theorem 1, (A_i, α_i) are determined by the following recurrence formulas.

$$A_{i+1} = -(4A_i + 15\alpha_i^2), \quad \alpha_{i+1} = \alpha_i \pm 2\left(3\alpha_i^2 + A_i\right)^{\frac{1}{2}} \tag{18}$$

where the plus-minus sign remains ambiguous in the second formula. Precisely, the second formula is presented as follows:

$$\alpha_{i+1} \leftarrow \texttt{select}(\alpha_i + 2\left(3\alpha_i^2 + A_i\right)^{\frac{1}{2}}, \alpha_i - 2\left(3\alpha_i^2 + A_i\right)^{\frac{1}{2}}, b_{i+1}).$$

We make Algorithms 3 and 4 using formulas (18).

Algorithm 3. `OurIsogSeq1` : Computation of a sequence of 2-isogenies

```
Input : $j_0 = j(E_0)$, walk data $\omega = b_0 b_1 \ldots b_{n-1}$.
Output : $j_n = j(E_n)$.
$(A_0, B_0) \leftarrow (A(j_0), B(j_0))$, $(\alpha_0, \beta_0) \leftarrow \texttt{select}_0(A_0, B_0, b_0)$.
for $i \leftarrow 0$ to $n-2$ do
    $(A_{i+1}, \alpha_{i+1}) \leftarrow \texttt{OurIsog1}(A_i, \alpha_i, b_{i+1})$.
end for
$\xi \leftarrow \alpha_{n-1}^2$, $A_n \leftarrow -(4A_{n-1} + 15\xi)$, $B_n \leftarrow -\alpha_{n-1}(8A_{n-1} + 22\xi)$, $j_n \leftarrow j(A_n, B_n)$.
return $j_n$.
```

Algorithm 4. `OurIsog1` : Computation of a 2-isogeny

```
Input : $A_i$, $\alpha_i$, select bit $b_{i+1}$.
Output : $A_{i+1}$, $\alpha_{i+1}$.
$\xi \leftarrow \alpha_i^2$, $\zeta \leftarrow 3\xi + A_i$, $\eta \leftarrow \zeta^{\frac{1}{2}}$.
$A_{i+1} \leftarrow -(4A_i + 15\xi)$.                  /* by Vélu's formula (2) */
$\lambda_0 \leftarrow \alpha_i + 2\eta$, $\lambda_1 \leftarrow \alpha_i - 2\eta$.   /* by Theorem 1 */
$\alpha_{i+1} \leftarrow \texttt{select}(\lambda_0, \lambda_1, b_{i+1})$.
return $A_{i+1}, \alpha_{i+1}$.
```

In particular, we can omit steps 3 and 4 in Algorithm 2 by using $\beta_{i+1} = -2\alpha_i$ and $\zeta' = 4\zeta$. The result is Algorithm 4. We can omit the calculation of B_i in Algorithm 4 as we already observed in Remark 1. Here, "ζ in Algorithm 4" is equivalent to "$\zeta = \zeta'/4$ in Algorithm 2".

Algorithm 3 iterates a subroutine `OurIsog1` (Algorithm 4).

Details of Algorithm 4 are as follows. At step 1, intermediate variables are calculated. At step 2, we calculate A_{i+1} based on Vélu's formula (2). At step 3, we calculate the x-coordinates of the 2-torsion points other than the backtracking point using already the obtained η. At step 4, we calculate α_{i+1} according the select bit b_{i+1} using the selector function `select` defined by (6). Finally, return (A_{i+1}, α_{i+1}).

5.4 Proposed Method 2

From Proposition 1 and Theorem 1, $(\alpha_i, \beta_i, \gamma_i)$ are determined by the recurrence formulas (15) and (16). Precisely, the second formula is presented as follows:

$$\alpha_{i+1} \leftarrow \texttt{select}(\alpha_i + 2\eta, \alpha_i - 2\eta, b_{i+1}), \quad \gamma_{i+1} \leftarrow \texttt{select}(\alpha_i + 2\eta, \alpha_i - 2\eta, \overline{b}_{i+1})$$

where $\eta = [(\beta_i - \alpha_i)(\gamma_i - \alpha_i)]^{\frac{1}{2}}$. We make Algorithms 5 and 6 by using these formulas. Algorithm 5 iterates a subroutine `OurIsog2` (Algorithm 6).

The proposed algorithm computes the sequence of x-coordinates of 2-torsion points on E_i only, without calculating (A_i, B_i).

The difference between Algorithms 4 and 6 lies in the way of calculating ζ. $\zeta = 3\alpha_i^2 + A_i$ in Algorithm 4 can be expressed as $\zeta = (\beta_i - \alpha_i)(\gamma_i - \alpha_i)$ using Lemma 2. We use the latter expression in Algorithm 6.

Details of Algorithm 6 are as follows. At step 1, intermediate variables are calculated. At step 2, we calculate β_{i+1} using (15). At step 3, we calculate the x-coordinates of the 2-torsion points other than the backtracking point using the

Algorithm 5. `OurIsogSeq2` : Computation of a sequence of 2-isogenies

```
Input : $j_0 = j(E_0)$, walk data $\omega = b_0 b_1 \dots b_{n-1}$.
Output : $j_n = j(E_n)$.
1: $(A_0, B_0) \leftarrow (A(j_0), B(j_0))$, $(\alpha_0, \beta_0) \leftarrow \texttt{select}_0(A_0, B_0, b_0)$.
2: for $i \leftarrow 0$ to $n-2$ do
3:     $(\alpha_{i+1}, \beta_{i+1}, \gamma_{i+1}) \leftarrow \texttt{OurIsog2}(\alpha_i, \beta_i, \gamma_i, b_{i+1})$.
4: end for
5: $\xi \leftarrow \alpha_{n-1}^2$, $A_n \leftarrow -[4\sigma_2(\alpha_{n-1}, \beta_{n-1}, \gamma_{n-1}) + 15\xi]$,
   $B_n \leftarrow -8\sigma_3(\alpha_{n-1}, \beta_{n-1}, \gamma_{n-1}) - 14\alpha_{n-1}\xi$, $j_n \leftarrow j(A_n, B_n)$.
6: return $j_n$.
```

Algorithm 6. `OurIsog2` : Computation of a 2-isogeny

```
Input : $\alpha_i$, $\beta_i$, $\gamma_i$, select bit $b_{i+1}$.
Output : $\alpha_{i+1}$, $\beta_{i+1}$, $\gamma_{i+1}$.
1: $\zeta \leftarrow (\beta_i - \alpha_i)(\gamma_i - \alpha_i)$, $\eta \leftarrow \zeta^{\frac{1}{2}}$.
2: $\beta_{i+1} \leftarrow -2\alpha_i$.                          /* by Proposition 1 */
3: $\lambda_0 \leftarrow \alpha_i + 2\eta$, $\lambda_1 \leftarrow \alpha_i - 2\eta$.          /* by Theorem 1 */
4: $\alpha_{i+1} \leftarrow \texttt{select}(\lambda_0, \lambda_1, b_{i+1})$, $\gamma_{i+1} \leftarrow \texttt{select}(\lambda_0, \lambda_1, \overline{b}_{i+1})$.
5: return $\alpha_{i+1}$, $\beta_{i+1}$, $\gamma_{i+1}$.
```

already obtained η. At step 4, we calculate α_{i+1} and γ_{i+1} according the select bit b_{i+1} using `select`. Finally, return $(\alpha_{i+1}, \beta_{i+1}, \gamma_{i+1})$.

5.5 Costs of Proposed Methods

The total cost of Algorithms 4 and 6, respectively, can be estimated easily. There are one field multiplication, and one square root computation.

Remark 2. In the proposed method 1, if the coefficient A_{i+1} of x in (5) for E_{i+1} is zero (i.e., $j(E_{i+1}) = 0$), then, Lemma 3 and Proposition 1 shows that α_{i+1} is given by $\alpha_i + \alpha_i\,(-3)^{\frac{1}{2}}$ or $\alpha_i - \alpha_i\,(-3)^{\frac{1}{2}}$. Then at the vertex E_{i+1}, further improved computation can be done.

6 Another Method Using the Modular Polynomial

We are told by an anonymous referee that there is another efficient method by using the modular polynomial $\Phi_2(x, y) = x^3 + y^3 - x^2y^2 + 1488(x^2y + xy^2) - 162000(x^2 + y^2) + 40773375xy + 8748000000(x + y) - 157464000000000$. We describe it briefly and compare it with our methods in Section 7.

Two (supersingular) elliptic curves $E_1/\mathbb{F}_{p^2}$ and $E_2/\mathbb{F}_{p^2}$ are related by a 2-isogeny if and only if $\Phi_2(j(E_1), j(E_2)) = 0$. So what one should do is: consider the polynomial $\Phi_2(x, j_i) = 0$ where j_i is the i-th j-invariant in a walk on the Pizer graph, which has three roots, one being j_{i-1}. Then factor the remaining quadratic and select the right j_{i+1} according to the select bit. Because supersingular j-invariants are in $\mathbb{F}_{p^2}$, all operations are done in $\mathbb{F}_{p^2}$.

We estimate the cost of the above method as 3 multiplications, 1 square root computation and 5 multiplications with constants (which are coefficients in $\Phi_2(x, y)$).

7 Comparison

Table 2 shows the costs of Algorithm 2, the method in Section 6, Algorithm 4 `OurIsog1`, and Algorithm 6 `OurIsog2`. In Table 2, "Mult.","Sq. root", and "Const. Mult." columns indicate the number of field multiplications, square roots, field multiplications with a constant, respectively. Here, 1 field inversion counts as 5 field multiplications as in [CGL07].

Each row of Table 2 shows the cost estimates of the algorithms given in this paper. Our proposed methods are a little more efficient than others, however, square root computation is dominant in all algorithms in Table 2. Its precise cost depends on the characteristic p, and its rough estimate is $O(\log p)$ multiplications in $\mathbb{F}_{p^2}$.

8 Example

Let $p = 1048583$. The polynomial $h(T) := T^2 + 653340\,T + 920973$ is irreducible over $\mathbb{F}_p$. Thus, we use the representation $\mathbb{F}_{p^2} \cong \mathbb{F}_p[\tau]$ where $h(\tau) = 0$ here.

Table 2. Costs for each 2-isogeny

Field operation	Mult.	Sq. root	Const. Mult.
`CGLIsog`	8	1	–
Method using Φ_2	3	1	5
`OurIsog1`	1	1	–
`OurIsog2`	1	1	–

Table 3. Example Input/Output of Algorithm 6

i	α_i	β_i	γ_i
0	775166	381302	940698
1	704826	546834	845506
2	$833407\ \tau + 922311$	687514	$215176\ \tau + 487341$

We will give examples of computing the sequence with $n = 3$. We want to calculate j_3 from $j_0 = 54000$ and the walk data $\omega = 001$. Then, we start with $E_0 : y^2 = x^3 + A_0x + B_0$ over $\mathbb{F}_{p^2}$ where $A_0 = 1013916, B_0 = 675944$. Table 3 lists the example input/output data of Algorithm 6, which gives the sequence of α_i, β_i, γ_i iteratively.

While running Algorithm 5, we don't know the values A_i and B_i where $i = 1, 2$. However, we can obtain finally $j(E_3) = j(A_3, B_3) = 286275\ \tau + 443480$ as in step 7 in Algorithm 5.

9 Conclusion

We have proposed compact algorithms for computing a sequence of 2-isogenies. They are based on observations on 2-torsion points. The algorithms are also interesting from the viewpoint of security evaluations.

Acknowledgments. We would like to thank an anonymous referee for his (or her) valuable comments and telling us the contents in Section 6 and permission to include it. Moreover, we would like to thank Tatsuaki Okamoto and Yoshifumi Manabe for their helpful comments and valuable discussions.

References

[AT02] Auer, R., Top, J.: Legendre elliptic curves over finite fields. J. Number Theory 95, 303–312 (2002)

[CGL07] Charles, D.X., Goren, E.Z., Lauter, K.E.: Cryptographic hash functions from expander graphs. To appear in Journal of Cryptology, electronically (2007), `http://www.springerlink.com/`

[CF06] Cohen, H., Frey, G., et al.: Handbook of Elliptic and Hyperelliptic Curve Cryptography. Chapman and Hall, Boca Raton (2006)

[HLW06] Hoory, S., Linial, N., Wigderson, A.: Expander graphs and their applications. Bull. AMS 43(4), 439–561 (2006)

[G00] Goldreich, O.: Candidate one-way functions based on expander graphs. Elect. Colloq. on Computational Complexity (ECCC) 7(090) (2000)

[G01] Goldreich, O.: Randomized Methods in Computation - Lecture Notes (2001), `http://www.wisdom.weizmann.ac.il/~oded/rnd.html`

[P90] Pizer, A.K.: Ramanujan graphs and Hecke operators. Bull. AMS 23(1), 127–137 (1990)

[RS06] Rostovtsev, A., Stolbunov, A.: Public-key cryptosystem based on isogenies (preprint, 2006), `http://eprint.iacr.org`

[Sc87] Schoof, R.: Nonsingular plane cubic curves over finite fields. J. Comb. Th., Series A 46, 183–211 (1990)

[S86] Silverman, J.H.: The Arithmetic of Elliptic Curves, GTM 106. Springer, Heidelberg (1986)

[T66] Tate, J.: Endomorphisms of Abelian varieties over finite fields. Inv. Math. 2, 134–144 (1966)

[V71] Vélu, J.: Isogénies entre courbes elliptiques. Comptes-Rendus de l'Académie des Sciences 273, 238–241 (1971)

Survival in the Wild: Robust Group Key Agreement in Wide-Area Networks

Jihye Kim and Gene Tsudik

University of California, Irvine
{jihyek,gts}@ics.uci.edu

Abstract. Group key agreement (GKA) allows a set of *players* to establish a shared secret and thus bootstrap secure group communication. GKA is very useful in many types of peer group scenarios and applications. Since all GKA protocols involve multiple rounds, robustness to player failures is important and desirable. A *robust* group key agreement (RGKA) protocol runs to completion even if some players fail during protocol execution.

Previous work yielded constant-round RGKA protocols suitable for the LAN setting, assuming players are homogeneous, failure probability is uniform and player failures are independent. However, in a more general wide-area network (WAN) environment, heterogeneous hardware/software and communication facilities can cause wide variations in failure probability among players. Moreover, congestion and communication equipment failures can result in correlated failures among subsets of GKA players.

In this paper, we construct the first RGKA protocol that supports players with different failure probabilities, spread across any LAN/WAN combination, while also allowing for correlated failures among subgroups of players. The proposed protocol is efficient (2 rounds) and provably secure. We evaluate its robustness and performance both analytically and via simulations.

1 Introduction

The last decade has witnessed a sharp spike in the popularity of collaborative applications, such as multi-media conferencing, distributed simulations, multi-user games and replicated servers. Such application often operate across the insecure and unstable "wilderness" of the global Internet. To be effective, collaborative applications need robust and secure communication. However, basic security services (such as confidentiality, integrity and authentication) require key management as the foundation.

A number of group key management techniques have been proposed. They generally fall into three categories: 1) centralized, 2) distributed and 3) contributory.

Centralized group key management involves a single entity that generates and distributes keys to group members via a pair-wise secure channel established with each group member. This is generally inappropriate for secure peer

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 66–83, 2009.

group communication, since a central key server must be, continuously available and present in every possible subset of a group in order to support continued operation in the event of arbitrary network partitions. Continuous availability can be addressed by using fault-tolerance and replication techniques. Unfortunately, the omni-presence issue is difficult to solve in a scalable and efficient manner.[1]

Distributed group key management is more suitable to peer group communication over unreliable networks. It involves dynamically selecting a group member that acts as a key distribution server. Although robust, this approach has a notable drawback in that it requires a key server to maintain long-term pairwise secure channels with all current group members in order to distribute group keys. Some schemes take advantage of data structures to minimize the number of encryption operations and messages generated whenever the group key changes. When a new key server is selected all these data structures need to be constructed.

In contrast, *contributory* group key agreement requires every group member to contribute an equal share to the common group secret, computed as a function of all members' contributions. This is particularly appropriate for dynamic peer groups since it avoids the problems with the single point(s) of trust and failure. Moreover, some contributory methods do not require establishing pairwise secret channels among group members. Also, unlike most group key distribution protocols, they offer strong key management security properties such as key independence and perfect forward secrecy (PFS) [11]. More detailed discussion can be found in [10].

In the rest of this paper we focus on *contributory* group key agreement and refer to it as GKA from here on.

1.1 Prior Work on Robust GKA

Most early work in GKA focused on security and efficiency. A number of basic protocols were proposed , notably: STR [14,10], BD [4], GDH [15] and TGDH [9]. All of these protocols are provably secure with respect to passive adversaries (eavesdroppers). Each protocol is efficient in its own way, while none is efficient in all respects (e.g., number of messages, rounds and cryptographic operations). To protect against active adversaries, *authenticated* versions of the above protocols were later constructed, e.g., [8,2,3].

All current GKA protocols involve multiple communication rounds. Since no one-round GKA has ever been proposed, the issue of robustness applies to all current GKA protocols; in fact, none of them is inherently robust. In this context, robustness means the ability to complete the protocol despite player and/or communication failures during protocol execution.

The robustness issue has been identified several years ago. In 2001, Amir, et al. [1] proposed the first robust GKA (RGKA) technique based on a (non-robust)

[1] However, that the centralized approach works well in one-to-many multicast scenarios since a trusted third party (TTP) placed at, or very near, the source of communication, can support continued operation within an arbitrary partition as long as it includes the source.

group key agreement protocol (called GDH) by Steiner, et al. [15], and a view-based group communication system (GCS) which provides the abstraction of consistent group membership. Since the GCS can detect crashes among players during the execution of GKA, the protocol can react accordingly. However, its round complexity is $O(n)$ and it requires $O(n^2)$ broadcasts where n is the number of players.

Subsequently, Cachin and Strobl (CS) proposed a very different constant-round RGKA technique operating over asynchronous networks [5]. It tolerates both player and link failures. The exact communication and infrastructure assumptions of the CS protocol depend on the choice of the consensus sub-protocol which the CS protocol invokes. However, assuming a reliable broadcast channel, the CS protocol takes 2 rounds, each player broadcasts $O(n)$-sized messages and performs $O(n)$ public key operations.[2]

More recently, Jarecki, et al. [7] proposed a 2-round RGKA protocol (called JKT) which operates over a reliable broadcast channel, and tolerates up to $O(T)$ player failures using $O(T)$-sized messages, for any $T < n$. It achieves a natural trade-off between message size and desired level of fault-tolerance. However, JKT assumes that: (1) every player has the same fault probability, and (2) all fault probabilities are random and independent.

1.2 Starting Point

The JKT protocol is well-suited for a local area network (LAN) environment. This is because the assumption of independent and random faults is valid in a typical LAN, where a group of players communicate directly via broadcast and are not directly bothered by failures of other players. Furthermore, JKT is geared for a homogeneous environment where each player runs on the same hardware/software platform. These two assumptions limit its scope. Specifically, JKT is a poor match for settings where players with heterogeneous hardware/software are spread across a wide-area network (WAN).

By allowing each player to piggyback its individual fault probability onto its first broadcast message, JKT can be used in a heterogeneous environment by trivially replacing the fault probability of every player by the highest fault probability. Such a protocol would be safe in terms of robustness, but it would cause larger messages, incurring higher costs than necessary for a specified level of robustness. Moreover, if there is a player with a very high fault probability, the protocol will always produce a maximum-sized messages for full robustness.

Also, a router failure in the WAN (e.g., due to a misconfiguration or congestion) increases the probability of network partitioning [12], which in turn increases the failure probability of the GKA protocol. Specifically, router failure results in the communication failure of all players which use that router as

[2] Assuming reliable broadcast, the CS protocol works as follows: First every player broadcasts its public encryption key. Then every player picks its contribution to the shared key, encrypts it under each broadcasted public key, and broadcasts a message containing the resulting n ciphertexts. The shared key is computed by each player as the sum of all broadcasted contributions.

a gateway. Assuming the router's fault probabilities are given (e.g., from historical statistics), one naive solution might be to determine a player's overall fault probability by combining player failure and router failure probabilities and then computing the suitable message size according to the result. However, it is unclear how to combine these two different types of probabilities: individual player's faults are independent, while player faults stemming from router crashes are correlated. Moreover, performance is not only determined by fault probabilities, but also by the order of the players. For example, there is a higher chance of partition for the same-sized message if players are randomly ordered, as in [7]. This prompts the question of how to order players and how to treat two different failures in order to obtain good performance.

1.3 Scope

We consider applications where a peer group of players is distributed over any combination of LANs and WANs (including the Internet). The group is a long-term entity and a group-wide secret key is needed to bootstrap secure communication. We assume that router failure probability can be computed from historical statistics. Examples of the kind of groups we focus on are as follows:

- **Collaborative workspace system.** A collaborative workspace is a distributed environment, wherein participants in geographically spread locations can access each other's data and interact as if they were at a single location. A group of people, e.g., covering America, Europe, and Asia, can cooperate to brainstorm, take a note, and develop source code. Softwares supporting group collaboration include: CoOffice (working in MS Office programs), SubEthaEdit (a real-time editor designed for MAC OS X), Gobby (available on Windows and Unix-like platforms), and UNA (a development environment for software engineers). Many such settings are real-time in nature and better GKA performance improves the overall QoS of group communications.
- The Border Gateway Protocol (BGP) is the core routing protocol of the Internet and a group of Internet BGP routers operates in one autonomous domain. From a security perspective, they need to function as a single entity and hence must agree on a common key. Extraneous tasks (such as key management) are often the bottleneck (and pure overhead) for high-speed routers. Saving bandwidth and rounds is a real concern.
- Consider a group of entities (routers or servers) in extreme environments, such as deep-space, that lack continuous network connectivity.[3] In such a setting, re-starting a GKA protocol, because a single participant failed, results in inordinately expensive costs.

[3] Note that IETF/IRTF for several years already have a working group called Delay-Tolerant Networking Research Group to explore issues (including security) in the context of very-long-distance (and hence high-delay) networking.

– Security policies usually dictate that group keys must be refreshed periodically.[4] Thus, a GKA protocol needs to be re-run (perhaps often) and improving GKA performance is essential.

1.4 Contributions

First, we investigate how to efficiently use prior work in a setting with heterogeneous players and construct a protocol that supports more flexible control parameters. We localize the message size parameter for each player and allow a player to compute its message size adaptively depending on reliability level of its neighbors.

Second, we address the challenge of combining two different types of fault probabilities (individual player and sub-group of players) by treating each type at a different layer. Basically, we plug two types of control parameters into our protocol: one (computed from a player fault probability) increases player connectivity within a clustered subgroup, and the other (computed from a router fault probability) increases connectivity among subgroups.

Third, since player ordering affects performance in a heterogeneous setting, we determine – through step-by-step simulations – which ordering is preferable in order to maximize efficiency. Simulation results show that random player ordering in the same subgroup outperforms non-random order, e.g., topological order or fault probability increasing order. In addition, keeping the topological order of a player among subgroups outperforms random order of players beyond its subgroup range.

Finally, we construct the first RGKA protocol that supports players with different failure probabilities, spread across any LAN/WAN combination, while also allowing for correlated failures among subgroups of players. The proposed protocol is efficient (2 rounds) and provably secure. We evaluate its robustness and performance both analytically and via simulations.

1.5 Organization

The rest of this paper is organized as follows: Section 2 presents our terminology, notation, communication and adversarial models, as well as necessary security definitions and cryptographic assumptions. Next, Section 3 describes the new RGKA protocol. Section 4 evaluates its performance and section 5 concludes the paper.

[4] Keys can be compromised in many ways, including by brute-force deciphering or so-called "lunch-hour attacks", a difficult-to-detect form of espionage where key information is obtained by physical means from within. Once a key is compromised, all of the information transmitted over the communication link is vulnerable until the key is refreshed. For systems with very low (or zero) key-refresh rates, a compromised key provides an eavesdropper full access to information encrypted with that key.

2 Preliminaries

2.1 Terminology and Notation

We now summarize our notation and terminology.

- **Players.** Participating set of n players are denoted: P_1, ..., P_n where the ordering is determined by the protocol itself.
- **Sub-group.** A subset of players who can communicate directly, i.e., those on the same LAN.
- **(Border) Router.** A device that forwards data between sub-groups. It might be a player. Router failure causes the entire sub-group to fail (become disconnected) from the perspective of other players.
- **Failures/Faults.** Any player and any router can crash. A player crash results from hardware/software failure. A router crash results from network misconfiguration or traffic congestion and causes the communication failure of all players that use that router as a gateway to the rest of the world.
- **Multicast Communication.** We assume that all communication takes place over *reliable* and authenticated multicast channels [13,6] where all non-faulty players have the same view of the broadcasted message (which can be null if the sender is faulty). We assume weak synchrony, i.e., players have synchronized clocks and execute the protocol in synchronized rounds. Messages from non-faulty players must arrive within some fixed time window, which we assume is large enough to accommodate clock skews and reasonable communication delays.
- **Adversary.** We assume an honest-but-curious outside adversary which can also impose arbitrary stop faults on the (otherwise honest) players. (We note, however, that using standard zero-knowledge proofs our protocols can easily be strengthened to tolerate malicious insiders at the price of a small constant increase in communication and computation.) Also, although the adversary can make each player stop at any time during protocol execution, such player failure can not violate the contract imposed by the reliable multicast assumption. The goal of the adversary is to learn the group key(s).

2.2 System Model

Figure 1 illustrates our system model. Our security model is a standard model for GKA protocols executed over authenticated links[5]. Since players in our setting do not use long-term secrets, we define GKA security (following [4,8,7]), as semantic security of the session key created in a single instance of the GKA protocol executed among honest parties. Specifically, the adversary can not distinguish between the key and a random value (with probability negligibly over 1/2). The formal definition is as follows:

[5] Note that there are standard and inexpensive "compilation" techniques which convert any GKA protocol into an *authenticated* GKA protocol [8].

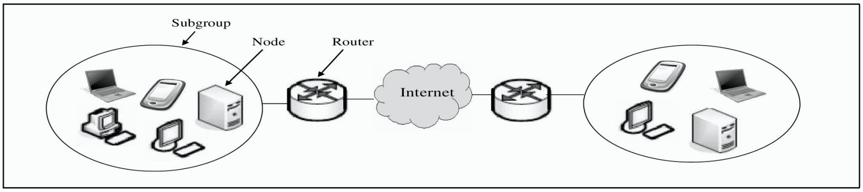


Fig. 1. System Model

Definition 1. (**GKA Security**) *Consider an adversarial algorithm $\mathcal{A}$ which observes an execution of the GKA protocol between n honest players, and, depending on bit b, is given the session key computed by this protocol (if $b = 1$) or a random value chosen from the same domain as the session key (if $b = 0$). The adversary $\mathcal{A}$ outputs a single bit b'. We define adversary's advantage:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{GKA}} = |\Pr[b' = b] - 1/2|$$

where the probability goes over the random execution of the protocol, the adversary $\mathcal{A}$, and the random choice of bit b.

We call a GKA protocol secure *if, for all adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{GKA}}$ is negligible.*

2.3 Cryptographic Setting

We now describe our cryptographic assumptions. This section is included for the sake of completeness and can be skipped without any loss of continuity.

Let $\mathbb{G}$ be a cyclic group of prime order q, and let g be its generator. We assume that both Decision Diffie-Hellman (DDH) and Square-DDH problems are hard in $\mathbb{G}$. For example, $\mathbb{G}$ could be a subgroup of order q in the group of modular residues $\mathbb{Z}_p^*$ s.t. $p - 1$ divides q, $|p| = 1024$ and $|q| = 160$, or it can be a group of points on an elliptic curve with order q for $|q| = 160$.

Definition 2. *The DDH problem is hard in $\mathbb{G}$, if, for every algorithm A, we have: $|\Pr[x, y \leftarrow \mathbb{Z}_q : A(g, g^x, g^y, g^{xy}) = 1] - \Pr[x, y, z \leftarrow \mathbb{Z}_q : A(g, g^x, g^y, g^z) = 1]| \leq \epsilon$ and ϵ is negligible.*

Definition 3. *The Square-DDH problem is hard in $\mathbb{G}$ if for every A we have: $|\Pr[x \leftarrow \mathbb{Z}_q : A(g, g^x, g^{x^2}) = 1] - \Pr[x, z \leftarrow \mathbb{Z}_q : A(g, g^x, g^z) = 1]| \leq \epsilon$ and ϵ is negligible.*

3 Robust Group Key Agreement Protocol in a WAN

In this section, we show the construction of a WAN-oriented RGKA protocol. As mentioned earlier, our work builds on [7]. We thus begin by describing the JKT protocol in more detail.

3.1 Overview of JKT Protocol

JKT is basically a robust version of the 2-round GKA protocol by Burmester and Desmedt (BD) [4]. (We describe the BD protocol in appendix A.) BD succeeds only if the second-round message values we call *gadgets* form a circular path through the graph of all "live" players. The idea behind adding robustness is simple: In the second round, players send out *additional* gadgets, such that, even if some players fail in the broadcast stage, messages broadcasted by the live players can still compute a key. In the following, we briefly describe a T-robust GKA protocol and then a fully robust GKA protocol which repeats the T-robust protocol until it succeeds.

T-robust GKA Protocol. As in [7], this protocol operates in two rounds assuming n players are ordered *cyclically* and *randomly*: P_1,...,P_n. In practice, the order can be determined by the hash of the first-round message: each player P_i broadcasts a public version $z_i = g^{t_i}$ of its (secret) contribution t_i to the group key. (We assume, for simplicity and without loss of generality, that all players survive the first round of the protocol.) In the second round, each P_i broadcasts gadget values: $X_{[k,i,i']} = (z_k/z_i)^{t_i}$ for $|k-i| \leq T$. (Note that gadgets $X_{[k,i,j]}$ for $|k-i| \leq T$ and $|j-i| \leq T$ can be also constructed since $X_{[k,i,j]} = X_{[k,i,i']}/X_{[j,i,i']}$.)

For the better delivery of the protocol description, following [7], we redefine the gadget value using a graph terminology and explain how gadgets are used to agree upon a key. A gadget $X_{[k,i,j]}$ corresponds to the path of length two connecting players P_k, P_i, and P_j. Two gadgets are **connectable** if there exists a overlapping path. For example, for every i, gadgets $X_{[a_{i-1},a_i,a_{i+1}]}$ and $X_{[a_i,a_{i+1},a_{i+2}]}$ are connectable because the path of P_{a_i} and $P_{a_{i+1}}$ overlaps. If we virtually split a node into two nodes as shown in the *node-doubling* technique [7], we add dash $'$ to denote the index of the other node. For example, node i is split into node i and node i'. For example, for every i, gadgets $X_{[a_{i-1},a_i,a_{i'}]}$ and $X_{[a_{i+1},a_i,a_{i'}]}$ are connectable because the path of P_{a_i} and $P_{a_{i'}}$ overlaps.

Each player ends up computing the same group key if the sequence of gadgets sent by all live players forms a circular path through the graph of all live players. If all live players form a *path*, a cycle can be also constructed by visiting every player twice as described in [7]. Let $P_{a_1}, ..., P_{a_m}$ denote the players who survive after the second broadcast round and form a circular path. Each P_{a_i} computes session key as:

$$\mathsf{sk}_{a_i} = (z_{a_{i-1}})^{m \cdot t_{a_i}} \cdot X_{a_i}^{m-1} \cdot X_{a_{i+1}}^{m-2} \cdot \cdots \cdot X_{a_{i-2}} = \mathsf{sk} = g^{t_{a_1}t_{a_2}+t_{a_2}t_{a_3}+\ldots+t_{a_m}t_{a_1}}$$

where $X_{a_i} = X_{[a_{i-1},a_i,a_{i'}]}/X_{[a_{i+1},a_i,a_{i'}]}$. The actual protocol – as viewed by a single player – is shown in Figure 2.

Fully Robust GKA with *Homogeneous* and *Random* Faults. A fully robust (but *not* constant-round) GKA protocol simply repeats the T-robust protocol above, with some parameter T, which we fix from the player fault probability and the expected number of rounds, until the T-robust protocol succeeds. Repeating the protocol increases the number of rounds and the protocol

[**Round 1**]:

1.1 Each P_i picks a random $t_i \in Z_q$ and broadcasts $z_i = g^{t_i}$.

[**Round 2**]:

2.1 Let $\mathsf{ActiveList}$ be the list of indices of all players who complete Round 1.

2.2 Each P_i broadcast gadgets: $X_{[k,i,i']} = (z_i/z_k)^{t_i}$ for T nearest neighbors to the right and T nearest neighbors to the left, among players $k \in \mathsf{ActiveList}$. Define $X_{[i,i',k]}$ as $(X_{[k,i,i']})^{-1}$.

[**Key Computation**]:

3.1 Let $\mathsf{ActiveList}$ be the list of indices of all players who complete Round 2.

3.2 Every P_i sorts $\mathsf{ActiveList}$ in the same order and connects each pair of connectable gadgets. The session key can be computed only if P_i can construct a cycle either from a true Hamiltonian cycle or from a Hamiltonian path taken twice; wlog, we assume that the path is formed as $\{P_{a_1}, ..., P_{a_m}\}$, where for some i, j we have $a_i = a_j$ and $m \le n$.

3.3 Each P_{a_i} computes $\mathsf{sk}_{a_i} = (z_{a_{i-1}})^{m \cdot t_{a_i}} \cdot X_{a_i}^{m-1} \cdot X_{a_{i+1}}^{m-2} \cdot \cdots \cdot X_{a_{i-2}}$ where $X_{a_i} = X_{[a_{i-1}, a_i, a_{i'}]} \cdot X_{[a_i, a_{i'}, a_{i+1}]}$.

(Note that $\mathsf{sk}_{a_i} = g^{t_{a_1} t_{a_2} + t_{a_2} t_{a_3} + \ldots + t_{a_m} t_{a_1}}$.)

Fig. 2. The robust GKA Protocol with homogeneous players in a LAN

communication complexity, i.e., given protocol failure probability f and the message size per player T, EXP(R)= $1 + 1/(1-f)$, EXP(MS)= $1 + 2T/(1-f)$, respectively[6]. Assuming that player faults happen *independently* on the ν rate, the protocol failure probability is upper-bounded by: $f \le n^2/2 * \nu^{2T}$.

Therefore, given n, ν, and f, we can compute a minimal gadget size – T – with which the protocol fails with probability at most f. As a result, the protocol will have at most $1 + 1/(1-f)$ rounds. This is described in Algorithm 1.

The JKT protocol can upper-bound protocol failure probability f and compute optimal message size T using approximation techniques, assuming homogeneous players. However, it is not clear how to compute f and T in a heterogenous player model. Moreover, while individual player faults are independent, subgroup faults are correlated.

Even if we could compute optimal message size, it would work only for a particular order of participating players. In other words, for each message size, the order of players changes the performance of the protocol. Recall that the JKT protocol computes gadgets for T nearest neighbors hoping that at least one of them survives all protocol steps. If a given player is surrounded by other players with high fault probabilities and gadgets connects only those players, the said player will very likely end up disconnected.

In the following two sections, we explore how to order players and how to compute the message size heuristically.

[6] Note that the protocol restarts only the second round since the messages from the first round are safely reusable.

Algorithm 1. Optimal T Selection in random fault model

```
Input: $(n, \nu, f)$
Output: $T$
for ($T' \leftarrow 1$ to $n/2$) do
    $f' \leftarrow n^2/2 * \nu^{2T}$
    if $f' < f$ then
        break
$MinMS \leftarrow 1 + 2T'/(1 - f')$
for ($T' \leftarrow T' + 1$ to $n/2$) do
    $f' \leftarrow n^2/2 * \nu^{2T}$
    $MS \leftarrow 1 + 2T'/(1 - f')'$
    if $MinMS > MS$ then
        $MinMS \leftarrow MS$
return $T$
```

3.2 Random or Non-random Order?

Heterogeneous Players on a *LAN*. Basically, there are two extreme cases: ordering players by their fault probabilities and ordering them randomly.[7] For the same number of gadgets T, to see which way of ordering provides better performance, we simulate the protocol for each case, compute the expected number of rounds and the expected message size, and then compare them.

We use a simple scenario with two subgroups of 25 players with a low and a high fault probability, respectively. In this scenario step, we assume that every player is on the same LAN and thus do not consider router failures. We denotes by μ router failure probability (or subgroup fault probability).

Table 1. Scenario: two subgroups of players with different but *independent* fault probabilities, Results: expected number of rounds and expected message size with three different T values on two different player orders

Group ID	A	B
n	25	25
ν	0.01	0.3
μ	0	0

Ordering	Topological Order			Random Order		
T	2	3	4	2	3	4
EXP(R)	2.913	2.052	2.005	2.266	2.010	2.000
EXP(MS)	8.652	7.312	9.040	6.064	7.060	9.000

The summary of this scenario and the results are summarized in Table 1. We simulate the scenario with three different values of T (2, 3, and 4) in topological and random orders, respectively. For every T, random order outperforms topological order. We believe that this is because random order uniformly distributes players with low fault rate and increases the probability for every player to have non-faulty players among its T nearest neighbors. Thus, random order can be a practical solution for players with different, but independent, fault probabilities.

[7] Of course, other ordering criteria are possible, e.g., by bit error rates of player interfaces. However, there is considerably less intuitive justification for considering these criteria.

Table 2. Scenario: two subgroups of players are given with different but *correlated* fault probabilities, Results: expected number of rounds and expected message size with three different T values, with two player orders

Group ID	A	B
n	25	25
ν	0	0
μ	0.01	0.3

Ordering	Topological Order			Random Order		
T	2	3	4	2	3	4
EXP(R)	2.000	2.000	2.000	2.427	2.427	2.007
EXP(MS)	5.000	7.000	9.000	6.708	9.562	9.056

Heterogeneous Players in a *WAN*. To see how correlated failures affect performance, we simulate another scenario with two subgroups of 25 players. Each subgroup connects to the WAN via a router. We assume that one router has low, and the other – high, failure probability. Since we now focus on correlated failures, we also assume that an individual player never fails, but only its communication can fail due to the failure of the router connecting its subgroup to the WAN.

The summary of this scenario and the results are summarized in Table 1. We simulate the scenario with three different values of T (2, 3, and 4) in topological and random orders, respectively. For every T, random order outperforms topological order. We believe that this is because random order uniformly distributes players with low fault rate and increases the probability for every player to have non-faulty players among its T nearest neighbors. Thus, random order can be a practical solution for players with different, but independent, fault probabilities.

Heterogeneous Players in a LAN/WAN. As shown above, random player order improves performance with independent player faults, while topological order achieves better performance with correlated subgroup faults. The natural next step is to consider the case of *both* independent and correlated faults, e.g., in a mixed LAN/WAN setting. To this end, we simulate a scenario with 4 subgroups, each composed of 15 players. Each player has an independent fault probability (among players) and each subgroup also has an independent fault probability (among subgroups). A player thus fails either alone or as part of its subgroup failure. In the latter case, the disconnected subgroup (containing a given player) might still complete the protocol; however, from the perspective of outside players, all players in the failed subgroup are gone.

This scenario and simulation results are described in Table 3. They show that topological order has fewer expected rounds and lower expected message

Table 3. Scenario: 4 subgroups of 15 players each with different fault probabilities, Results: expected number of rounds and message size with 3 T values, with two different player orders

Group ID	A	B	C	D
n	15	15	15	15
ν	0.2	0.01	0.1	0.3
μ	0.01	0.1	0.3	0.1

Ordering	Topological Order			Random Order		
T	2	3	4	2	3	4
EXP(R)	3.371	2.165	2.045	3.701	2.404	2.165
EXP(MS)	10.484	7.99	9.360	11.804	9.424	10.32

size. We conclude that, to obtain better performance, players should be ordered topologically by subgroups and randomly within a subgroup.

3.3 Player-Specific Message Size

In this section, we describe how to compute optimal message sizes for different players. We assume that players are grouped into subgroups topologically and randomly within subgroups. We determine message size in terms of both player-to-player and subgroup-to-subgroup communication, respectively.

Player-to-Player. Our approach is to localize T depending on the reliability of neighboring players. Specifically, a player has a larger T with less reliable neighbors, and a lower T with more reliable neighbors. If player fault probabilities are evenly distributed, we can obtain the best performance by simply letting each player compute the same number of gadgets T using algorithm 1. Whereas, more realistically, if player fault probabilities are unevenly spaced, to make every point equally reliable, each player has to adaptively compute the number of gadgets depending on the robustness of its neighbors. (Recall that a player is disconnected if all T nearest neighbors fail.)

We introduce three new variables:

- OT_i: optimal number of gadgets, applicable to both right- and left-side neighbors of P_i, *assuming* that each player has the same fault probability as P_i.
- RT_i and LT_i: localized numbers of gadgets applicable to P_i's right and left side neighbors, respectively.

We estimate P_i's robustness from its OT_i value. For example, a player with $OT = 1$ is most robust, while a player with $OT = n/2$ is least robust (where n is the number of players in a subgroup. In our algorithm, whenever a player computes a gadget which makes a connection to one of its neighbors, the robustness level of the player increases in inverse proportion to its neighbor's OT. P_i computes RT_i and LT_i according to its right and left neighbor's OT values (OT_{i-1}, OT_{i+1}), respectively, until its robustness level reaches a specified level, which is 1 in our algorithm. A more precise description is shown in Algorithm 2.

Note that a gadget that a player computes for one neighbor can not be used if the neighbor does not compute a reciprocal gadget. (Recall that gadgets are connectable if there exists a overlapping path.) In fact, in Algorithm 2, for a given pair of players, either both compute a gadget for each other or neither does.

Proposition 1. *In Algorithm 2, if a player computes a gadget for a neighbor, then the neighbor computes a reciprocal gadget.*

Proof. Since RT and LT are symmetric, we focus only on RT. Assume that P_i computes $RT_i = k$. Since P_{i+k} is added as one of its right-side neighbors, the summation of *reliability* from P_{i+1} to P_{i+k-1} is less than one, i.e., , $\sum_{j=i+1}^{i+k-1} \frac{1}{OT_j} < 1$. Therefore, P_{i+k} computes $LT_{i+k} \geq k$ including P_i as one of its left neighbors, since $\sum_{j=i+1}^{i+k-1} \frac{1}{OT_j} < 1$.

Algorithm 2. *T* Localization on player-to-player basis

```
Input: $(n, \nu_1, \cdots, \nu_n, f)$
Requirement 1: $P_1, \cdots, P_n$ are randomly ordered within subgroups and topologically across subgroups.
Requirement 2: $n$ is the number of players and indices cycle modulo $n$, i.e. $P_{n+1} = P_1$
Requirement 3: $P_i$'s fault probability is $\nu_i$
Ensure: for each $P_i$, compute $RT_i$ and $LT_i$
for ($i \leftarrow 1$ to $n$) do
    Compute optimal $OT_i$ using Algorithm 1 on input $(n, \nu_i, f)$
for ($i \leftarrow 1$ to $n$) do
    $reliability \leftarrow 0$
    $RT_i \leftarrow 0$
    for ($j \leftarrow i+1$; $reliability \geq 1$ or $j - i \geq n/2$; $j \leftarrow j+1$) do
        $RT_i \leftarrow RT_i + 1$    (Add $P_j$ to the list of $P_i$'s right neighbors)
        $reliability \leftarrow reliability + 1/OT_j$
    $reliability \leftarrow 0$
    $LT_i \leftarrow 0$
    for ($j \leftarrow i-1$; $reliability \geq 1$ or $i - j \geq n/2$; $j \leftarrow j-1$) do
        $LT_i \leftarrow LT_i + 1$    (Add $P_j$ to the list of $P_i$'s left neighbors)
        $reliability \leftarrow reliability + 1/OT_j$
```

Subgroup-to-Subgroup. In the proposed algorithm, we logically treat each subgroup as a kind of a *super-player*. Specifically, in each subgroup, a player with the lowest fault probability becomes a representative and sends out extra gadgets for other representatives. A representative player becomes faulty if either the player itself or its subgroup fails. Thus, given player failure rate ν and subgroup failure rate μ, the failure probability [8] of a representative player is: $\nu + \mu - \nu\mu$. The algorithm is described in Algorithm 3.

Proposition 2. *In Algorithm 3, if a representative player computes a gadget value for a representative neighbor then the representative neighbor also computes a gadget value for the representative player.*

Proof. Identical to the proof of Proposition 1.

3.4 RGKA in a LAN/WAN Setting

Based on Algorithms 2 and 3, we propose a *W-RGKA* protocol for heterogeneous players. *W-RGKA* allows each player to adaptively compute its message size depending on the reliability level of its neighbors. *W-RGKA* automatically defaults to the JKT protocol [7] in a setting with homogeneous players. Note that the

[8] At the player-to-player level, a player also fails from either its own or its subgroup fault. However, since players are topologically ordered and gadgets connect only nearest neighbors, the subgroup fault is not considered in the player-to-player level robustness.

Algorithm 3. T Localization in subgroup-to-subgroup level

```
Input: $(m, \nu_1, \cdots, \nu_m, \mu_1, \cdots, \mu_m, f)$
Requirement 1: $P_1, \cdots, P_m$ are a set of representative players
Requirement 2: $m$ is the number of subgroups and indices cycle mod $m$, i.e. $P_{m+1} = P_1$
Requirement 3: $P_i$'s fault probability and subgroup fault probability are $\nu_i$ and $\mu_i$, respectively
Ensure: for each $P_i$, compute $RT_i$ and $LT_i$
for ($i \leftarrow 1$ to $m$) do
    Compute optimal $OT_i$ using Algorithm 1 on input $(m, (\mu_i + \nu_i - \mu_i \cdot \nu_i), f)$
for ($i \leftarrow 1$ to $m$) do
    $reliability \leftarrow 0$
    $RT_i \leftarrow 0$
    for ($j \leftarrow i + 1$; $reliability \geq 1$ or $j - i \geq m/2$; $j \leftarrow j + 1$) do
        $RT_i \leftarrow RT_i + 1$    (Add $P_j$ to the list of $P_i$'s right neighbors)
        $reliability \leftarrow reliability + 1/OT_j$
    $reliability \leftarrow 0$
    $LT_i \leftarrow 0$
    for ($j \leftarrow i - 1$; $reliability \geq 1$ or $i - j \geq m/2$; $j \leftarrow j - 1$) do
        $LT_i \leftarrow LT_i + 1$    (Add $P_j$ to the list of $P_i$'s left neighbors)
        $reliability \leftarrow reliability + 1/OT_j$
```

homogeneous player setting is a special case of Algorithm 2, where every player computes the same $OT = RT = LT$. In other words, we succeed in extending the JKT protocol without losing its optimality in a homogeneous setting.

We also relax the way the key is computed such that *any* circular path that connects all live players can be used for key computation. The resulting graph that gadgets draw in two levels is more complex than the one (called T-th power of a circle) shown in the JKT protocol which builds either a Hamiltonian cycle or a Hamiltonian path on all live players. To enable stronger robustness, we relax the way of finding a circular path, so that the key is associated not necessarily with a Hamiltonian cycle or a Hamiltonian path, but any circular path where a player can be visited more than once. If there is no partition, there is a always a circular path. The resulting *W-RGKA* protocol is shown in Figure 3.

Theorem 1. *Assuming that the DDH problem and Square-DDH problem are hard, protocol* W-RGKA *is a secure Group Key Agreement.*

The security of the *W-RGKA* protocol which broadcasts RT and LT sized messages is implied by the security argument for the RGKA protocol in [7] which broadcasts maximum sized messages, thus revealing maximum amount of information. The only difference is that the resulting key in *W-RGKA* might contain each contribution of the form t_{a_i}, $t_{a_{i+1}}$ more than once. However, the resulting key equation is still linearly independent from the equations generated from gadgets. Thus, the key value is independent from gadgets values and the adversary cannot learn anything about the key from the messages it observes. For details, refer to Section 6.2 in [7].

[**Round 1**]:

1.1 Same as in Figure 2 except that each P_i also broadcasts its fault probability ν_i.

[**Round 2**]: same as in Figure 2 except:

2.2 Each player P_i computes RT_i and LT_i using Algorithm 2 and broadcast gadgets $X_{[k,i,i']} = (z_i/z_k)^{t_i}$ for RT_i nearest neighbors to the right and LT_i nearest neighbors to the left among players $k \in \mathsf{ActiveList}$.

***2.3** Each representative P_i computes RT_i and LT_i using Algorithm 3 and broadcast gadgets $X_{[k,i,i']} = (z_i/z_k)^{t_i}$ for RT_i nearest neighbors to the right and LT_i nearest neighbors to the left among representative players $k \in \mathsf{ActiveList}$.

[**Key Computation**]: Same as in Figure 2 except:

3.2 The session key can be computed if there exists a circular path where every player is visited at least more than once. Wlog, we assume that the path is formed as $\{P_{a_1}, \cdots, P_{a_m}\}$, where for some i, j we can have $a_i = a_j$.

* Executed only by representative players

Fig. 3. *W-RGKA* protocol with heterogeneous players in a LAN/WAN setting

4 Performance Evaluation

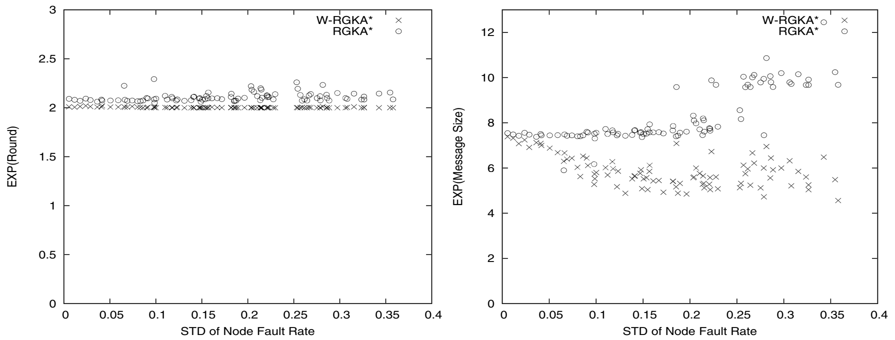
We first summarize the relevant aspects of protocol efficiency.

- **Round Complexity:** number of protocol rounds.
- **Communication Complexity:** (expected) total bit-length of all messages sent in the protocol.
- **Computational Complexity:** computation that must be performed by each player.

REMARK: In the specific GKA protocols we compare, computational complexity increases in proportion to communication complexity. Generally, one message unit incurs one exponentiation (which dominates computational cost). Thus we do not separate computational complexity from communication complexity in the following comparison.

We compare *W-RGKA* with the fully robust JKT protocol [7], as described in section 3.1. To make *W-RGKA* fully robust we repeat it until it succeeds; this is the same approach used to obtain a fully robust version of JKT in [7]. We denote our fully robust version as *W-RGKA** and the fully robust JKT version by *RGKA**. However, since *RGKA** works in a homogeneous setting, we simulate *RGKA** by taking the average of all player fault probabilities.

We analyze how player heterogeneity and correlated faults affect performance. We evaluate the protocols in a setting of 5 subgroups with 10 players for each. In the first simulation, we generate subgroup fault probabilities such that the average subgroup fault probability is around 0.1 (with a standard deviation less than 0.1) and generate 10 players for each group with random fault probability, such that the standard deviation varies between 0 and 0.4. Note that the

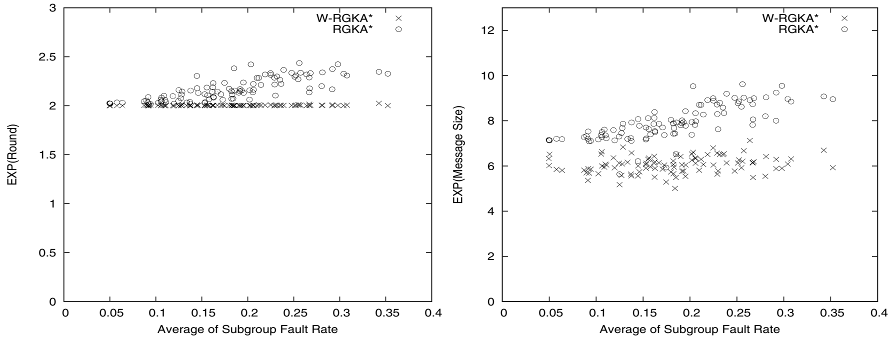


(a) expected number of rounds (b) expected message size

Fig. 4. *W-RGKA** vs. *RGKA** for different standard deviations of player fault rate distribution. Results are based on simulating over 100 runs.

standard deviation of player fault probability distribution indirectly shows the heterogeneity of the set of players. In the second simulation, we generate players randomly but with a small deviation (less than 0.1) and change subgroup fault probabilities such that the average ranges from 0 to 0.4.

Figure 4 shows the expected number of rounds (a) and expected message size (b) for each protocol with different standard deviations. Overall, *W-RGKA** outperforms *RGKA** in both round and communication complexity. The number of rounds in *W-RGKA** tops out at 2.01, compared with 2.2 in *RGKA**. This might seem insignificant, however, considering that the underlying non-robust BD protocol always takes 2 rounds, the difference becomes more substantial. We also observe that the communication cost of *W-RGKA** is far lower than that of *RGKA**, particularly, with higher standard deviation in player fault rates.



(a) expected number of rounds (b) expected message size

Fig. 5. *W-RGKA** vs. *RGKA** for different average subgroup fault rates. Results are based on simulating over 100 runs.

Figure 5 shows the expected number of rounds (a) and expected message size (b) for both protocols, taking into account subgroup fault rates. Once again, *W-RGKA** exhibits better performance on both counts. The number of rounds in *W-RGKA** still lies below 2.01. Whereas, for *RGKA**, the number of rounds increases proportionally to averaged correlated fault rates, and thus quickly shoots up to 2.5. Also, communication complexity of *RGKA** increases as the average of correlated fault rates grows. This is mainly because *RGKA** does not consider correlated faults in its design.

5 Conclusions

This paper started off with the state-of-the-art in robust GKA protocols. Having identified certain limitations of prior work, i.e., assumptions about independent failures and homogeneous players, we demonstrated a step-by-step construction of a new protocol *W-RGKA* suitable for a mixed LAN/WAN setting. While the proposed protocol inherits the attractive features of its predecessor (JKT), it also heuristically determines per-player optimal message sizes and handles heterogeneous fault probabilities as well as correlated failures. Simulations help determine the preferred player order for different scenarios.

One obvious item for future work is to conduct a more extensive set of experiments and simulations. Another issue is that the current protocol does not take into account inter-subgroup delay. It is natural to consider this variable (assuming it is known ahead of time) in determining the optimal subgroup order.

References

1. Amir, Y., Nita-Rotaru, C., Schultz, J., Stanton, J., Kim, Y., Tsudik, G.: Exploring robustness in group key agreement. In: ICDCS, pp. 399–408 (2001)
2. 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, p. 290. Springer, Heidelberg (2001)
3. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.: Provably authenticated group Diffie-Hellman key exchange. In: ACM CCS (November 2001)
4. Burmester, M., Desmedt, Y.G.: A secure and efficient conference key distribution system. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
5. Cachin, C., Strobl, R.: Asynchronous group key exchange with failures. In: PODC, pp. 357–366 (2004)
6. Floyd, S., Jacobson, V., Liu, C., McCanne, S., Zhang, L.: A reliable multicast framework for light-weight sessions and application level framing. IEEE/ACM ToN 5(6), 784–803 (1997)
7. Jarecki, S., Kim, J., Tsudik, G.: Robust group key agreement using short broadcasts. In: ACM CCS, pp. 411–420 (2007)
8. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. Journal of Cryptology 20(1), 85–113 (2007)

9. Kim, Y., Perrig, A., Tsudik, G.: Simple and fault-tolerant key agreement for dynamic collaborative groups. In: ACM Conference on Computer and Communications Security, pp. 235–244 (2000)
10. Kim, Y., Perrig, A., Tsudik, G.: Group key agreement efficient in communication. IEEE ToC 33(7) (2004)
11. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
12. Moser, L., Amir, Y., Melliar-Smith, P., Agarwal, D.: Extended virtual synchrony. In: ICDCS, pp. 56–65 (1994)
13. Paul, S., Sabnani, K., Lin, J., Bhattacharya, S.: Reliable multicast transport protocol (rmtp). IEEE JSAC 15(3), 407–421 (1997)
14. Steer, D.G., Strawczynski, L., Diffie, W., Wiener, M.: A secure audio teleconference system. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 520–528. Springer, Heidelberg (1990)
15. Steiner, M., Tsudik, G., Waidner, M.: Key agreement in dynamic peer groups. IEEE TPDS 11(8), 769–780 (2000)

A Burmester-Desmedt GKA

The BD GKA protocol proceeds in two rounds (see Figure 6): First each player P_i broadcasts a public counterpart $z_i = g^{t_i}$ of its contribution t_i to the key. In the second round each P_i broadcasts gadget $X_{[i-1,i,i+1]} = g^{t_i t_{i+1} - t_{i-1} t_i}$ (which it can compute as $X_{[i-1,i,i+1]} = (z_{i+1}/z_i)^{t_i}$). Given the set of gadget values $X_{[n,1,2]}, X_{[1,2,3]}, ..., X_{[n-1,n,1]}$, each player P_i can use its contribution t_i to locally compute the common session key $\mathsf{sk} = g^{t_1 t_2 + t_2 t_3 + \ldots + t_n t_1}$.

[**Round 1**]:

Each player P_i picks a random $t_i \in Z_q$ and broadcasts $z_i = g^{t_i}$.

[**Round 2**]:

Each P_i broadcasts its gadget value $X_{[i-1,i,i+1]} = (z_{i+1}/z_{i-1})^{t_i}$, where the indices are taken in a cycle.

[**Key Computation**]:

Each P_i computes the key as $\mathsf{sk}_i = (z_{i-1})^{nt_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2}$, where $X_i = X_{[i-1,i,i+1]}$.

(Note that for all i we have $\mathsf{sk}_i = g^{t_1 t_2 + t_2 t_3 + \ldots + t_n t_1}$.)

Fig. 6. Burmester-Desmedt's Group Key Agreement Protocol (BD GKA)

As we explain in section 3.1, a sequence of gadgets *forms a path through the graph* if each two consecutive gadgets in the sequence are connectable. By inspecting the formula for deriving the secret key in the BD GKA protocol we can observe that each player derives the same key because the set of gadgets broadcasted in the second round of the protocol forms a Hamiltonian cycle (i.e. a circular path) through the graph of all players.

Visual Secret Sharing Schemes with Cyclic Access Structure for Many Images

Miyuki Uno and Mikio Kano

Department of Computer and Information Sciences
Ibaraki University, Hitachi, Ibaraki, 316-8511 Japan
uno.miyuki@gmail.com, kano@mx.ibaraki.ac.jp
http://gorogoro.cis.ibaraki.ac.jp/

Abstract. We consider a visual secret sharing scheme with cyclic access structure for n secret images and n shares, where two consecutive shares decode one secret image. This secret sharing scheme can be constructed by using Droste's method. However the contrast of its scheme is $1/(2n)$. In this paper, it is shown that for every integer $n \geq 4$, there exists no construction of such a visual secret sharing scheme having a perfect black reconstruction and contrast at least $1/4$. Also for every even integer $n \geq 4$, a new construction of such a visual sharing scheme that satisfies a slightly weaker condition and has a contrast $1/4$ is given.

1 Introduction

A visual secret sharing scheme (VSS scheme), which is also called a visual cryptography scheme (VCS), was introduced by Naor and Shamir [9]. Since then, it have been studied in many papers including [1,2,3,6]. A VSS scheme is a special kind of secret sharing scheme in which the secret is an image comprised of black and white pixels and encoded into n shares, where each share is usually printed on a transparency. In k-out-of-n VSS scheme, the secret image can be obtained only by stacking k of the shares, but we cannot get any information about the secret image from fewer than k shares.

Droste [5] introduced the following generalized VSS scheme and gave its construction. Let $\mathcal{F}$ be a family of non-empty subsets of $\{1, 2, \ldots, n\}$, and $\{Image(X) \mid X \in \mathcal{F}\}$ be a set of $|\mathcal{F}|$ secret images, each of which corresponds to an element of $\mathcal{F}$. Then we can construct n shares $Share(1)$, $Share(2)$, ..., $Share(n)$ so that for any element $X \in \mathcal{F}$, a stack of the shares in $\{Share(i) \mid i \in X\}$ recovers the secret image $Image(X)$, and we cannot get any information about $Image(X)$ from a set $\{Share(i) \mid i \in Y\}$ for $X \not\subseteq Y \subset \{1, 2, \ldots, n\}$. The family $\mathcal{F}$ is called the access structure of the VSS scheme.

If we apply this construction to a VSS scheme with cyclic access structure given below, then each pixel is split into $2n$ subpixels and its contrast is $1/(2n)$. Thus this VSS scheme loses a lot of contrast in reconstructed images when n is large.

In this paper, we first prove that for every $n \geq 4$, there exists no construction of a VSS scheme with cyclic access structure that has a perfect black reconstruction and contrast greater than or equal to $1/4$. Next, for every even $n \geq 4$,

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 84–97, 2009.

we give a new construction of a VSS scheme with cyclic access structure that satisfies a slightly weaker condition and has a contrast $1/4$. For $n = 3$, we give a similar results with contrast $1/6$.

We now explain a VSS scheme with cyclic access structure and another such a VSS scheme satisfying a slightly weaker condition. They consist of n shares $Share(1)$, ..., $Share(n)$ and n secret images $Image(1)$, ..., $Image(n)$ and posses the following properties either (a),(b),(c) or (a),(b*),(c):

(a) for every $1 \leq i \leq n$, a stack of $Share(i)$ and $Share(i+1)$ reconstructs $Image(i)$, where $Share(n+1) = Share(1)$;
(b) for every $1 \leq k \leq n$, a set $\{Share(i) \mid 1 \leq i \leq n, i \neq k\}$ of $n-1$ shares gives us no information about $Image(k-1)$ and $Image(k)$;
(b*) for every $1 \leq k \leq n$, a set $\{Share(i) \mid 1 \leq i \leq n,\ i \neq k, k+1\}$ of $n-2$ shares gives us no information about $Image(k-1)$, $Image(k)$, $Image(k+1)$; and
(c) this VSS scheme is *perfect*, that is, it has a perfect black reconstruction. So every black pixel of a secret image is recovered into a pure black region in the reconstructed image.

The condition (b*) says that if two consecutive shares $Share(k)$ and $Share(k+1)$ are missing, then any information about three images $Image(k-1)$, $Image(k)$, $Image(k+1)$ cannot be obtained. It is obvious that a VSS scheme having the property (b) satisfies (b*), and so in this sense, we say that the condition (b*) is slightly weaker than (b). As we shall show, it is impossible to construct a VSS scheme with cyclic access structure satisfying (a), (b), (c) and having contrast at least $1/4$ for every $n \geq 4$. Keeping a high contrast $1/4$, we give a new construction of a VSS scheme with cyclic access structure satisfying (a), (b*), (c) for every even $n \geq 4$.

We now explain the contrast of a VSS scheme with perfect black reconstruction. Consider such a VSS scheme in which each pixel of secret images is split into m subpixels in a share. We say that such a perfect VSS scheme has a contrast δ if for every white pixel of secret images, at least δm subpixels of the corresponding pixel in the reconstructed images are white, and for a certain white pixel of a secret image, exactly δm subpixels of the corresponding pixel in the reconstructed images are white.

This paper is organized as follows: In Sect. 2, a construction of cyclic VSS scheme that satisfies (a), (b), (c) is given where $n = 3$. In Sect. 3, it is proved that for every $n \geq 4$, non-existence of the VSS scheme that satisfies (a),(b),(c) and has contrast greater than or equal to $1/4$. In Sect. 4, for every even $n \geq 4$, a construction of the VSS scheme satisfying (a), (b*), (c) and having contrast $1/4$ is proposed. In appendix A, it is proved that for $n = 3$ non-existence of the VSS scheme satisfying (a), (b), (c) and having contrast greater than $1/6$. In appendix B, an example of the VSS scheme for $n = 6$ and with contrast $1/4$ is shown.

Other results on VSS scheme with many secret images can be found in [6], [10] and etc.

2 Preliminaries and a VSS Scheme with Cyclic Access Structure for $n = 3$

We first introduce some notations and definitions used throughout this paper. Consider a VSS scheme with cyclic access structure consisting of n secret images $Image(1), \ldots, Image(n)$ and n shares $Share(1), \ldots, Share(n)$. All the secret images are comprised of black and white pixels. Each pixel of secret images is split into m subpixels in a share.

Hereafter we consider any fixed pixel x of secret images, and denote its color in $Image(i)$ by $img(i) = img_x(i)$, and by $S(i) = S_x(i)$ the set of subpixels of $Share(i)$ corresponding to the pixel x. Then the pixel x corresponds to the $m \times n$ subpixels $S(1) \cup S(2) \cup \cdots \cup S(n)$. These $m \times n$ subpixels can be expressed by a $m \times n$ $(0, 1)$-matrix $B = [b_{ij}]$, where $b_{ij} = 1$ if the i-th subpixel of $S(j)$ is black, otherwise $b_{ij} = 0$. Namely, the j-th column vector of B corresponds to $S(j)$, and we also use $S(j)$ to denote the j-th column vector of B. The matrix B is called a *basis matrix* of the VSS scheme, which is the transposed matrix of usually used basis matrix. For convenience, this matrix is used in this paper. A 2×2 $(0, 1)$-matrix

$$M(i) = \begin{bmatrix} m_{i1} & m_{i3} \\ m_{i2} & m_{i4} \end{bmatrix}$$

is randomly chosen from the two matrices of the following (1) if the pixel of a secret image is black, and otherwise it is randomly chosen from (2).

$$\left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right\}, \tag{1}$$

$$\left\{ \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \right\}. \tag{2}$$

A VSS scheme with cyclic access structure for three secret images and three shares is presented as Fig. 1.

We adopt the Droste's method. Each pixel is split into six subpixels, and the 6×3 $(0, 1)$-matrix $B = [b_{ij}]$ is defined as follows:

$$B = \begin{bmatrix} m_{11} & m_{13} & 1 \\ m_{12} & m_{14} & 1 \\ 1 & m_{21} & m_{23} \\ 1 & m_{22} & m_{24} \\ m_{33} & 1 & m_{31} \\ m_{34} & 1 & m_{32} \end{bmatrix} = [S(1), S(2), S(3)],$$

which contains $M(1)$, $M(2)$ and $M(3)$ as submatrices.

Consider any fixed pixel x of images. If $img(1)$ is black, then $M(1)$ is chosen from (1), and so perfect black region is reconstructed by $S(1)$ and $S(2)$, otherwise $M(1)$ is chosen from (2), and thus one common subpixels of $S(1)$ and $S(2)$ are white, and hence a white region is reconstructed. For other images, the colors are

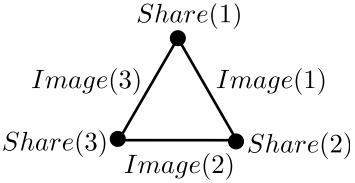


Fig. 1. The secret images and the shares correspond to the edges and the vertices, respectively

reconstructed in the same way by using $M(2)$ and $M(3)$. The security condition (b) is proved in [5].

We will prove in the appendix that it is impossible to construct a cyclic VSS scheme satisfying (a), (b), (c) and having contrast greater than 1/6.

3 Non-existence of the VSS with Contrast at Least 1/4

In this section we shall show that if $n \geq 4$, then the contrast of a VSS scheme with cyclic access structure for n images satisfying the conditions (a), (b), (c) is less than 1/4. Namely, we prove the following theorem.

Theorem 1. *Let $n \geq 4$ be an integer. Then there exists no construction of a VSS scheme with cyclic access structure for n secret images that satisfies (a), (b), (c) and has contrast greater than or equal to* 1/4.

Proof. Assume that there exists a construction of a VSS scheme with cyclic access structure for n images which satisfies (a), (b), (c) and whose contrast is greater than or equal to 1/4. We consider a fixed pixel x of images, and use the same notation as in the previous section. Namely, we write $img(i)$ for the color of x in $Image(i)$, and $S(i)$ for the set of subpixels of $Share(i)$ corresponding to x. Suppose that each pixel is split into m subpixels. Let $W_i,\ B_i \subseteq \{1, 2, \cdots, m\}$ denote the indices of white subpixels and black subpixels of $S(i)$, respectively, as follows (Fig. 2):

$$W_i = \{k \mid \text{the } k\text{-th subpixel of } S(i) \text{ is white}\},$$
$$B_i = \{k \mid \text{the } k\text{-th subpixel of } S(i) \text{ is black}\}.$$

Put $|W_i| = w_i$ and $|B_i| = b_i$. Then $m = w_i + b_i$ for every i.

Let λ denote the minimum number of $|W_i \cap W_{i+1}|$ such that $img(i)$ is white and $1 \leq i \leq n$. Since the contrast is greater than or equal to 1/4, we have $\lambda/m \geq 1/4$, and thus

$$m \leq 4\lambda. \tag{3}$$

First consider the case that n is even. Without loss of generality, we may assume that w_1 is maximum among all $w_1, w_3, \ldots, w_{n-1}$ with odd suffixes. Take a triple $(S_1(1), S_1(2), S_1(3))$ so that $|W_1 \cap W_3|$ is maximum among all triples $(S(1), S(2), S(3))$. Let $t = |W_1 \cap W_3|$ for the $(S_1(1), S_1(2), S_1(3))$ (Fig. 2 (i), (iv)).

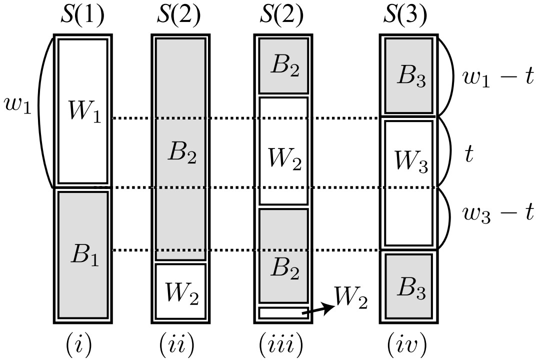


Fig. 2. The sets $S(1), S(2), S(3)$ of pixels

Assume $w_3 - t < \lambda$. If $img(2)$ is white, then $|W_2 \cap W_3| \geq \lambda$ and so $W_1 \cap W_2 \cap W_3 \neq \emptyset$. Since our VSS scheme is perfect, this implies that $S_1(1)$ and $S_1(2)$ must decode a white pixel. Namely, from $(S_1(1), S_1(3))$, we can obtain the information that $(img(1), img(2)) = (black, white)$ never occurs. This contradicts the security condition (b). Hence $w_3 - t \geq \lambda$. By the choice of w_1, we obtain

$$w_1 \geq w_3 \geq \lambda + t. \tag{4}$$

Consider a triple $(S_2(1), S_2(2), S_2(3))$ decoding $(img(1), img(2)) = (white, white)$ (Fig. 2 (i), (iii), (iv)). Then for these $S_2(i)$, it follows that $|W_1 \cap W_2| \geq \lambda$ and $|W_2 \cap W_3| \geq \lambda$, and so

$$|W_2| \geq 2\lambda - |W_1 \cap W_3| \geq 2\lambda - t \tag{5}$$

by the maximality of t. By considering a triple $(S_3(1), S_3(2), S_3(3))$ decoding $(img(1), img(2)) = (black, black)$ (Fig. 2 (i),(ii),(iv)), we have $W_2 \subseteq B_1 \cap B_3$ since the VSS scheme is perfect. Therefore it follows from Fig. 2 (iv), (4), (5) and the maximality of t that

$$\begin{aligned} m = |S_3(3)| &\geq |W_1 \cap B_3| + |B_1 \cap B_3| + |W_3| \\ &\geq |W_1 \cap B_3| + |W_2| + |W_3| \\ &\geq (w_1 - t) + (2\lambda - t) + (\lambda + t) \\ &\geq \lambda + 2\lambda - t + \lambda + t = 4\lambda. \end{aligned}$$

This inequality together with (3) implies $m = 4\lambda$, $|W_1 \cap B_3| = w_1 - t = \lambda$, $|B_1 \cap B_3| = |W_2| = 2\lambda - t$ and $|W_3| = \lambda + t$. Hence the following equality (6) and statement (7) hold.

$$w_1 = w_3 = \lambda + t, \quad w_2 = 2\lambda - t. \tag{6}$$

If $(img(1), img(2)) = (black, black)$ then

$$|W_1 \cap W_3| = t \quad \text{and} \quad B_1 \cap B_3 = W_2. \tag{7}$$

Notice that if the contrast is greater than 1/4, then $m > 4\lambda$ in (3), and so we derive a contradiction. Namely, hereafter we consider the case that the contrast is exactly 1/4.

By applying the same argument to $(S(3), S(4), S(5))$, we obtain

$$w_3 = w_5 = \lambda + t', \quad w_4 = 2\lambda - t', \tag{8}$$

where t' is the maximum value of $|W_3 \cap W_5|$. Hence it follows from (6) and (8) that $t = t'$ and

$$w_1 = w_3 = w_5 = \lambda + t, \quad w_2 = w_4 = 2\lambda - t.$$

By repeating the above argument for $(S(j), S(j+1), S(j+2))$, where $j = 5, \ldots, n-1$, we have

$$w_1 = w_3 = \cdots = w_{n-1} = \lambda + t, \tag{9}$$

$$w_2 \ = w_4 = \cdots = w_n = 2\lambda - t. \tag{10}$$

Let $s = |W_2 \cap W_4|$ be the maximum value among all triples ($S(2)$, $S(3)$, $S(4)$). Then by the same argument as above, we obtain

$$w_2 \ \ = w_4 = \cdots = w_n = \lambda + s, \tag{11}$$

$$w_1 = w_3 = \cdots = w_{n-1} = 2\lambda - s. \tag{12}$$

Moreover, it follows from (7) and the symmetry of t and s that if ($img(2)$, $img(3)$) = ($black$, $black$) then

$$|W_2 \cap W_4| = s \quad \text{and} \quad B_2 \cap B_4 = W_3. \tag{13}$$

Therefore it follows from (9), (10), (11) and (12) that for every integer $1 \leq i \leq n/2$,

$$\lambda = s + t, \quad w_{2i-1} = \lambda + t, \quad w_{2i} = \lambda + s.$$

By $m = 4\lambda = 4(s+t)$ and the symmetry of s and t, we may assume that $t \geq 1$. Consider a sequence ($S(1)$, $S(2)$, $S(3)$, $S(4)$) decoding ($img(1)$, $img(2)$, $img(3)$) = ($white$, $black$, $black$) (Fig. 3). If $|W_1 \cap W_3| \neq t$, then by (7) we can get the information from $S(1)$ and $S(3)$ without $S(2)$ that ($img(1)$, $img(2)$) = ($black$, $black$) does not occur. This contradicts the condition (b). Hence

$$|W_1 \cap W_3| = t. \tag{14}$$

Since $|W_1 \cap W_2| \geq \lambda$, $W_2 \cap W_3 = \emptyset$, (14) and $|W_1| = \lambda + t$, we have $|W_1 \cap W_2| = \lambda$ (Fig. 3 (ii)). By (13), we have $B_2 \cap B_4 = W_3$ and $|W_2 \cap W_4| = s$. Hence

$$|W_1 \cap W_4| = |W_1 \cap W_2 \cap W_4| \leq |W_2 \cap W_4| = s. \tag{15}$$

Next consider a sequence $(S(1), S(2), S(3), S(4))$ decoding ($img(1)$, $img(2)$, $img(3)$) = ($black$, $black$, $black$) (Fig. 4). Then $|W_1 \cap W_3| = t$ and $B_1 \cap B_3 = W_2$

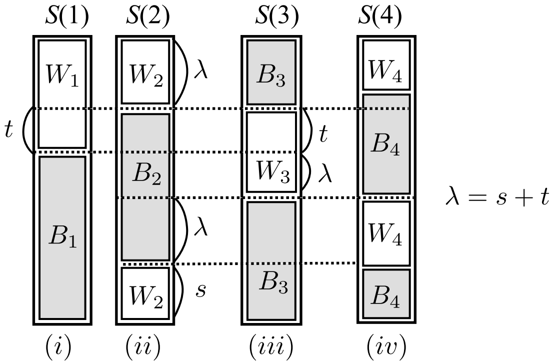


Fig. 3. $(S(1), S(2), S(3), S(4))$ for $(img(1),\ img(2),\ img(3))$=$(white,\ black,\ black)$

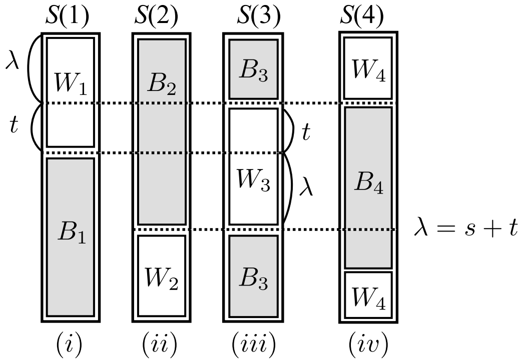


Fig. 4. $(S(1), S(2), S(3), S(4))$ for $(img(1), img(2), img(3)) = (black, black, black)$

by (7). Hence the structure of $(S(1), S(2), S(3))$ is determined as Fig. 4. Since $t \geq 1$ and $B_2 \cap B_4 = W_3$ by (13), we obtain

$$|W_1 \cap W_4| = \lambda = s + t > s. \tag{16}$$

Therefore by (15) and (16), we can get the information from $(S(1),\ S(4))$ that if $|W_1 \cap W_4| = \lambda$, then $(img(1),\ img(2),\ img(3)) = (white,\ black,\ black)$ does not occur. This contradicts the security condition. Hence the proof is complete in this case.

Suppose that n is odd. By the same argument as (9) and (10), we can show that the following holds.

$$\lambda + t = w_1 = w_3 = \ldots = w_n \tag{17}$$

$$= w_2 = w_4 = \ldots = w_{n-1} = 2\lambda - t. \tag{18}$$

By applying the same argument as above, we can derive a contradiction. Consequently the theorem is proved.

4 A New Construction of VSS Scheme with Cyclic Access Structure for Even $n \geq 4$

In this section, for every even integer $n \geq 4$, a new construction of VSS scheme with cyclic access structure for n images that satisfies (a), (b*) and (c) is given. It has contrast 1/4, and every pixel of the images is split into four subpixels in each share.

Let $n = 2r \geq 4$. Hereafter, for any fixed pixel x of images, we consider the colors $img(1), \cdots, img(n)$ and the sets $S(1), \cdots, S(n)$ of subpixels corresponding to x. For every $1 \leq i \leq r$, let $A(i)$ and $B(i)$ denote two column vectors consisting of four entries. For convenience, let $A(r+1) = A(1)$ and $B(r+1) = B(1)$. Then by these $A(i)$ and $B(i)$, $S(i)$'s are randomly determined in one of the following two ways (Fig. 5).

$$(S(1), S(2), \cdots, S(n)) = \begin{cases} (A(1), B(1), A(2), B(2), \cdots, A(r), B(r)) \text{ or,} \\ (B(r), A(1), B(1), A(2), \cdots, A(r)). \end{cases}$$

For every $1 \leq i \leq r$, the four row vectors of $[A(i)A(i+1)]$ consist of

$$[0\ 0],\ [0\ 1],\ [1\ 0],\ [1\ 1]. \tag{19}$$

Namely, $[A(i)A(i+1)]$ is obtained from the following matrix by a permutation on the four row vectors:

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

On the other hand, $B(i)$ is a column vector consisting of one 0 entry and three 1's entries, and is determined by the colors of two consecutive colors $(img(2i-1), img(2i))$ or $(img(2i), img(2i+1))$ according to the decision of $S(i)$'s.

Example. Assume that $n = 2r = 6$ and $(S(1), \cdots, S(6)) = (A(1), B(1), A(2), B(2), A(3), B(3))$. Then we first determine three column vectors $A(1), A(2), A(3)$ so that every $[A(i)A(i+1)]$ consisting of four row vectors of (19). For example, the following three column vectors satisfy this condition.

$$[A(1)A(2)A(3)] = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

Assume that $(img(1), img(2), \cdots, img(6))$ are (*white*, *black*, *black*, *black*, *black*, *white*). Then $B(1)$ is determined by a pair (*white*, *black*) of colors so that the second row vector $[0, 1]$ of $[A(1), A(2)]$ works in the reconstruction of $img(1)$ and $img(2)$. Namely,

$$B(1) = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad [A(1)B(1)A(2)] = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

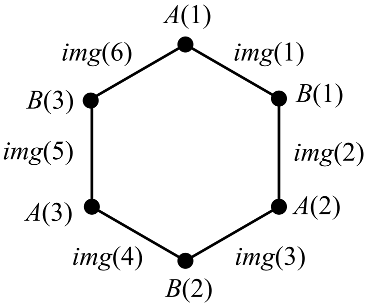


Fig. 5. The graph representing a VSS scheme with cyclic access structure for 6 images

Similarly, $B(2)$ and $B(3)$ are determined to reconstruct (*black*, *black*) and (*black*, *white*) by $[A(2), B(2), A(3)]$ and $[A(3), B(3), A(1)]$, respectively.
Hence

$$[A(1)B(1)A(2)B(2)A(3)B(3)] = \begin{bmatrix} 0\,1\,0\,1\,1\,0 \\ 0\,0\,1\,1\,0\,1 \\ 1\,1\,1\,0\,1\,1 \\ 1\,1\,0\,1\,0\,1 \end{bmatrix},$$

and thus the desired colors are reconstructed.

We now prove that a similar construction is always possible for every even integer $n \geq 4$. In order to do so, we need the next lemma.

Lemma 1. *Let $r \geq 2$ be an integer. Then a sequence $(X_1, X_2, \cdots, X_r)$ of r column vectors having the following properties can be constructed.*
(i) Each $X(i)$ consists of two 0 entries and two 1 entries.
(ii) For every $1 \leq i \leq r$, $[X(i)X(i+1)]$ consist of

$$[0\ 0],\ [0\ 1],\ [1\ 0],\ [1\ 1].$$

(iii) For any integer $1 \leq k \leq r$, we cannot guess $X(k)$ from the set $\{X(i) \mid 1 \leq i \leq r, i \neq k\}$ of $r-1$ vectors.

Proof. We first take $X(1)$ as

$$X(1) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

If $r = 2$, then $X(2)$ is determined as one of the following four vectors :

$$X(2) = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}, \text{ where } \{a, b\} = \{c, d\} = \{0, 1\}.$$

Assume $r \geq 3$. If $X(j)$, $j \geq 1$, is given, then $X(j+1)$ is obtained from $X(j)$ by independently and randomly replacing

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ by } \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Thus there exist four distinct $X(j+1)$. By this method, we can obtain $X(1)$, $X(2)$, $\cdots$, $X(r-1)$. The last vector $X(r)$ is randomly determined as follows depending on both $X(r-1)$ and $X(1)$. By symmetry, we may assume that $X(r-1)$ is one of the following vectors

$$X(r-1) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

Then determine

$$X(r) = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}, \begin{bmatrix} a \\ b \\ a \\ b \end{bmatrix} \text{ or } \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}, \text{ respectively,}$$

where $\{a,b\} = \{c,d\} = \{0,1\}$. Finally permute all the entries of all $X(i)$ simultaneously by any permutation on $\{1,2,3,4\}$.

We now prove the condition (iii). For any integer $1 \leq k \leq r$, consider the set $\{X(i) \mid 1 \leq i \leq r, i \neq k\}$. Without loss of generality, we may assume that

$$X(k-1) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \text{ and}$$

$$X(k+1) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

Then

$$X(k) = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}, \begin{bmatrix} a \\ b \\ a \\ b \end{bmatrix} \text{ or } \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}, \text{respectively,}$$

where $\{a,b\} = \{c,d\} = \{0,1\}$. Hence we cannot guess $X(k)$ from $\{X(i) \mid 1 \leq i \leq r, i \neq k\}$.

We are now ready to give a construction of the whole sequence. By Lemma 1, first take a random sequence $(A(1), A(2), \ldots, A(r))$. Then each $B(i)$ is chosen from the following four vectors so that $(A(i), B(i), A(i+1))$ reconstructs

$(img(2i-1), img(2i))$ or $(img(2i), img(2i+1))$ according to the decision of $S(i)$'s. Namely, we apply the same procedure in the case of $n = 6$.

$$\begin{bmatrix}0\\1\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\1\\0\end{bmatrix}.$$

We now discuss the security. Assume that $(A(1), B(1), A(2), \ldots, B(r)) = (S(1), S(2), \ldots, S(n))$. It is easy to see that if $B(i)$ is missing, then we cannot get any information about $img(2i-1)$ and $img(2i)$. So we shall next show that if two consecutive shares $A(k), B(k)$ or $B(k), A(k+1)$ are missing, then we cannot get any information about the three secret images $(Image(2k-2)$, $img(2k-1)$, $img(2k))$ or $(img(2k-1)$,$img(2k)$, $img(2k+1))$. By symmetry, we may assume that $A(k)$ and $B(k)$ are missing. It is clear that no information about $(img(2k-1)$, $img(2k))$ leaks because of a missing of $B(k)$. By the statement (iii) of Lemma 1, we cannot guess $A(k)$ from $\{A(i) \mid 1 \leq i \leq r, i \neq k\}$, which implies no information about $img(2k-2))$ leaks. Consequently, the construction of VSS scheme with cycle access structure for even number images is secure in the sense (b*).

We conclude the paper with the following problem.

Problem. For every odd integer $n \geq 5$, can we construct a VSS scheme with cyclic access structure for n images that satisfies (a), (b*), (c) and has contrast 1/4?

References

1. Ateniese, G., Blundo, C., De Santis, A., Stinson, D.R.: Visual Cryptography for General Access Structures. Information and Computation 129, 86–106 (1996)
2. Blundo, C., De Bonis, A., De Santis, A.: Improved schemes for visual cryptography. Designs, Codes and Cryptography 24, 255–278 (2001)
3. Blundo, C., De Santis, A., Stinson, D.R.: On the Contrast in Visual Cryptography Schemes. J. Cryptology 12, 261–289 (1999)
4. Blundo, C., De Santis, A.: Visual cryptography schemes with perfect reconstruction of black pixels. Computer and Graphics 22(4), 449–455 (1998)
5. Droste, S.: New results on visual cryptography. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 401–415. Springer, Heidelberg (1996)
6. Iwamoto, M., Yamamoto, H.: A construction method of visual secret sharing schemes for plural secret image. IEICE Trans. Fundamentals E86-A(10), 2577–2588 (2003)
7. Koga, H., Ueda, E.: The optimal (t, n)-threshold visual secret sharing scheme with perfect reconstruction of black pixels. Designs, Codes and Cryptography 40(1), 81–102 (2006)
8. Koga, H.: A general formula of the (t, n)-threshold visual secret sharing scheme. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 328–345. Springer, Heidelberg (2002)
9. Naor, M., Shamir, A.: Visual cryptography. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 1–12. Springer, Heidelberg (1995)

10. Uno, M., Kano, M.: Visual Cryptography Schemes with Dihedral Group Access Structure for Many Images. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 344–359. Springer, Heidelberg (2007)
11. Yi, F., Wang, D., Luo, P., Huang, L., Dai, Y.: Multi secret image color visual cryptography schemes for general access structures. Progr. Natur. Sci. (English Ed.) 16(4), 431–436 (2006)

A Appendix A: Non-existence of the VSS Scheme with Contrast Greater Than 1/6 Where $n = 3$

We now prove that for the VSS of three secret images, the contrast 1/6 is best possible. Consider any construction of a perfect VSS scheme with cyclic access structure for three shares and three secret images. We shall use the same notations as in Section 3. Assume that $S(i)$ consists of m subpixels, namely, each pixel is split into m subpixels, where $m \geq 2$. Let us define two subsets $W_i,\ B_i \subseteq \{1, 2, \cdots, m\}$ as in Section 3. Put $|W_i| = w_i$ and $|B_i| = b_i$. Then $m = w_i + b_i$ for every $i \in \{1, 2, 3\}$. Let λ denote the minimum number of $|W_i \cap W_{i+1}|$ such that $img(i)$ is white and $1 \leq i \leq 3$.

By considering the colors ($img(1)$, $img(2)$, $img(3)$) = (*black*, *black*, *black*), we have that W_i and W_j are disjoint for $i \neq j$, that is, $S(i)$ and $S(j)$ have no white subpixels in common (Fig. 6 (i),(ii),(iv)). Thus

$$m = |S(i)| \geq |W_1| + |W_2| + |W_3|. \tag{20}$$

Similarly, by considering the colors ($img(1)$, $img(2)$, $img(3)$) = (*white*, *white*, *black*), we have that $|W_1 \cap W_2| \geq \lambda$, $|W_2 \cap W_3| \geq \lambda$ and $W_3 \cap W_1 = \emptyset$ (Fig. 6 (i),(iii),(iv)). Hence $|W_2| \geq 2\lambda$. By considering other similar colors, we can obtain that $|W_1| \geq 2\lambda$ and $|W_3| \geq 2\lambda$. Therefore it follows from (20) that $m = |S(i)| \geq 6\lambda$, which implies that $\lambda/m \leq 1/6$. Hence the contrast of a VSS scheme for three images satisfying the conditions (a), (b), (c) is less than or equal to 1/6.

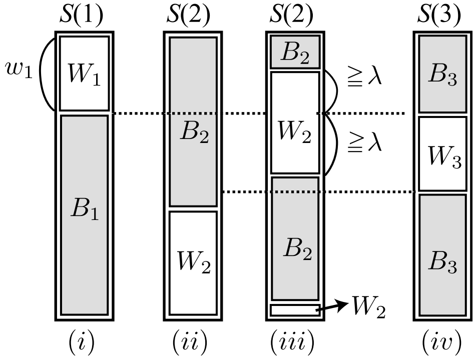


Fig. 6. W_i and B_i denote the indices of white and black subpixels of $S(i)$, respectively

A Appendix B: An Example of Cyclic VSS Scheme Where $n = 6$

An example of VSS scheme with cyclic access structure for six shares and six secret images are shown below. Here we encode secret images of 100×100 pixels. Two shares $Share(i)$ and $Share(i+1)$ recover $Image(i)$, where $Share(1) = Share(7)$.

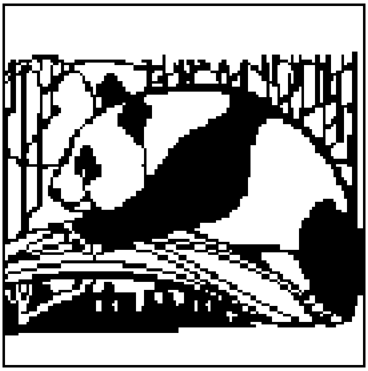


Fig. 7. The secret image $Image(2)$

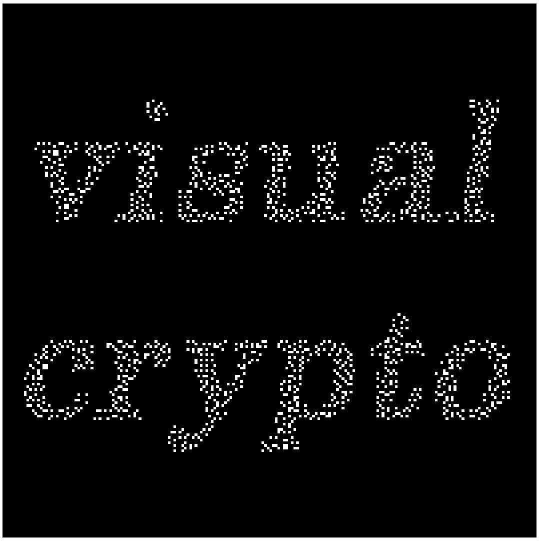


Fig. 8. A reconstructed image $Image(1)$

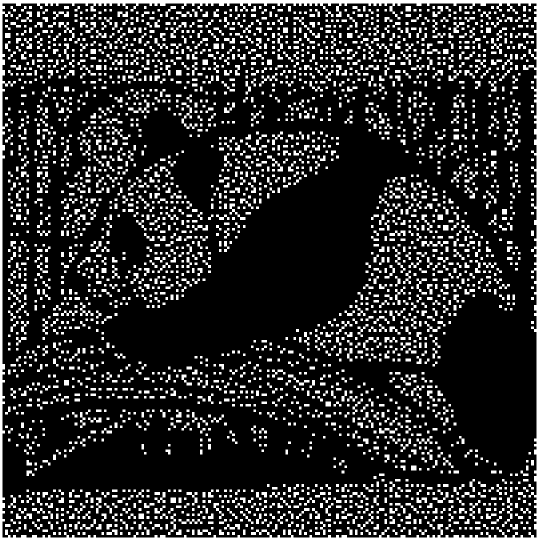


Fig. 9. A reconstructed image $Image(2)$

The Swiss-Knife RFID Distance Bounding Protocol

Chong Hee Kim*, Gildas Avoine, François Koeune*,
François-Xavier Standaert**, and Olivier Pereira**

Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium

Abstract. Relay attacks are one of the most challenging threats RFID will have to face in the close future. They consist in making the verifier believe that the prover is in its close vicinity by surreptitiously forwarding the signal between the verifier and an out-of-field prover. Distance bounding protocols represent a promising way to thwart relay attacks, by measuring the round trip time of short authenticated messages. Several such protocols have been designed during the last years but none of them combine all the features one may expect in a RFID system.

We introduce in this paper the first solution that compounds in a single protocol all these desirable features. We prove, with respect to the previous protocols, that our proposal is the best one in terms of security, privacy, tag computational overhead, and fault tolerance. We also point out a weakness in Tu and Piramuthu's protocol, which was considered up to now as one of the most efficient distance bounding protocol.

1 Introduction

Radio Frequency Identification (RFID) is a ubiquitous technology that enables identification of non-line-of-sight objects or subjects. Based on cheap RF - microcircuits – called tags – apposed on or incorporated into the items to identify, the RFID technology is widely deployed in our everyday lives. Several billion RFID tags are spread every year, in applications as diverse as pet identification, supply chain managment, Alzheimer's patient tracking, cattle counting, etc. RFID tags suited to such applications do not cost more than 0.20 USD.

The impressive potential of the RFID is not only exploited in identification solutions, but also in more evolved applications like access control, public transportation, payment, ePassport, etc. that require the tag to be cryptographically authenticated by the reader. To do so, a cipher and a pseudo-random number generator can be implemented on the tag while keeping its cost low – e.g. no more than 1 USD for a public transportation pass – but the number of calls to these cryptographic functions must be small enough to keep the authentication delay reasonable. Preserving privacy is also an expected feature of these protocols.

In practice, sensitive applications like those mentioned above rely on 2-pass or 3-pass challenge-response authentication protocols based on symmetric-key

* Research supported by the Walloon Region project E-USER (WIST program).

** Research Associates of the Fonds de la Recherche Scientifique - FNRS.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 98–115, 2009.

building blocks, typically block ciphers, although solutions based on asymmetric primitives have also been proposed. Such a design is secure in theory, but the real life is a bit different when dealing with RFID. Indeed, a tag is quite a simple device that automatically answers to any authentication query from a reader without alerting its holder. Hence the reader has no means to decide whether the tag's holder agreed to authenticate. Because the maximum reader-tag communication distance can not exceed a few decimeters with cryptography-compliant tags, the presence of the tag in the close environment of the reader is considered as an implicit authentication agreement from its holder. Providing the reader with a means to decide whether the distance to the tag is less than a given threshold is thus of the utmost importance to achieve practical security in RFID systems.

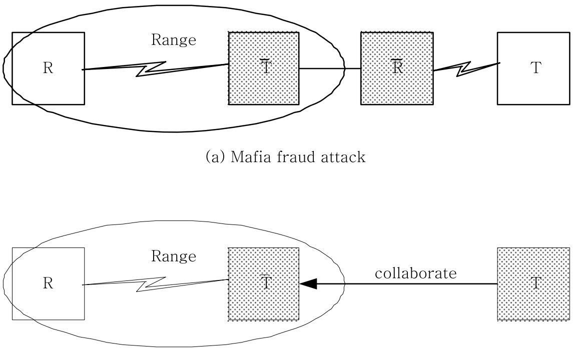
We introduce in this paper an RFID authentication protocol that allows such a verification. It is the first protocol that combines all the expected properties at the same time: it resists against both mafia fraud and terrorist attacks, reaches the best known false acceptance rate, preserves privacy, resists to channel errors, uses symmetric-key cryptography only, requires no more than 2 cryptographic operations to be performed by the tag, can take advantage of precomputation on the tag, and offers an optional mutual authentication. As an additional result, we also point out a weakness in the recent Tu and Piramuthu distance bounding protocol.

In Section 2, we introduce the relay attacks and the existing distance bounding protocols. We show that they all offer interesting features, but no one was yet able to combine all these features. We show in Section 3 a new attack against one of these protocols. We then describe our proposal in Section 4 and analyze it in Section 5. Finally, we provide a security and efficiency analysis.

2 Relay Attacks and Distance Bounding Protocols

2.1 Relay Attacks

There are two types of relay attacks: mafia fraud attack and terrorist fraud attack. **Mafia fraud attack** was first described by Desmedt [5]. In this attack scenario, both the reader R and the tag T are honest, but a malicious adversary is performing man-in-the-middle attack between the reader and the tag by putting fraudulent tag $\overline{T}$ and receiver $\overline{R}$. The fraudulent tag $\overline{T}$ interacts with the honest reader R and the fraudulent reader $\overline{R}$ interacts with the honest tag T. $\overline{T}$ and $\overline{R}$ cooperate together. It enables $\overline{T}$ to convince R as if R communicates with T, without actually needing to know anything about the secret information. **Terrorist fraud attack** is an extension of the mafia fraud attack. The tag T is not honest and collaborates with fraudulent tag $\overline{T}$. The dishonest tag T uses $\overline{T}$ to convince the reader that he is close, while in fact he is not. $\overline{T}$ does not know the long-term private or secret key of T. The problem with Mafia fraud attack is that this attack can be mounted without the notice of both the reader and the tag. It is difficult to prevent since the adversary does not change any data between the reader and the tag. Therefore mafia fraud attack cannot be



(a) Mafia fraud attack

(b) Terrorist fraud attack

Fig. 1. Mafia and terrorist fraud attacks

prevented by cryptographic protocols that operate at the application layer. Although one could verify location through use of GPS coordinates, small resource limited devices such as RFID tags do not lend themselves to such applications. *Distance bounding protocols* are good solutions to prevent such relay attacks. These protocols measure the signal strength or the round-trip time between the reader and the tag. However the proof based on measuring signal strength is not secure as an adversary can easily amplify signal strength as desired or use stronger signals to read from afar. Therefore many works are devoted to devise efficient distance bounding protocols by measuring round-trip time [2,3,7,10,11,4,14,15,16].

2.2 Distance Bounding Protocols

In 1993, Brands and Chaum presented their distance bounding protocol [2]. It consists of a *fast bit exchange* phase where the reader sends out one bit and starts a timer. Then the tag responds to the reader with one bit that stops the timer. The reader uses the round trip time to extract the propagation time. After series of n rounds (n is a security parameter), the reader decides whether the tag is within the limitation of the distance. In order to extract the propagation time, the processing time of the tag must be as short and invariant as possible. The communication method used for these exchanges is different from the one used for the ordinary communication. It does not contain any error detection or correction mechanism in order to avoid the introduction of variable processing cycles.

Although the idea has been introduced fifteen years ago, it is only quite recently that distance-bounding protocols attracted the attention of the research community. In 2005, Hancke and Kuhn proposed a distance bounding protocol (HKP) [7] that has been chosen as a reference-point because it is the most popular distance bounding protocol in the RFID framework. As depicted in Fig. 2, the protocol is carried out as follows. After exchanges of random nonces (N_a and N_b), the reader and the tag compute two n-bit sequences, v^0 and v^1, using a pseudorandom function (typically a MAC algorithm, a hash function, etc.).

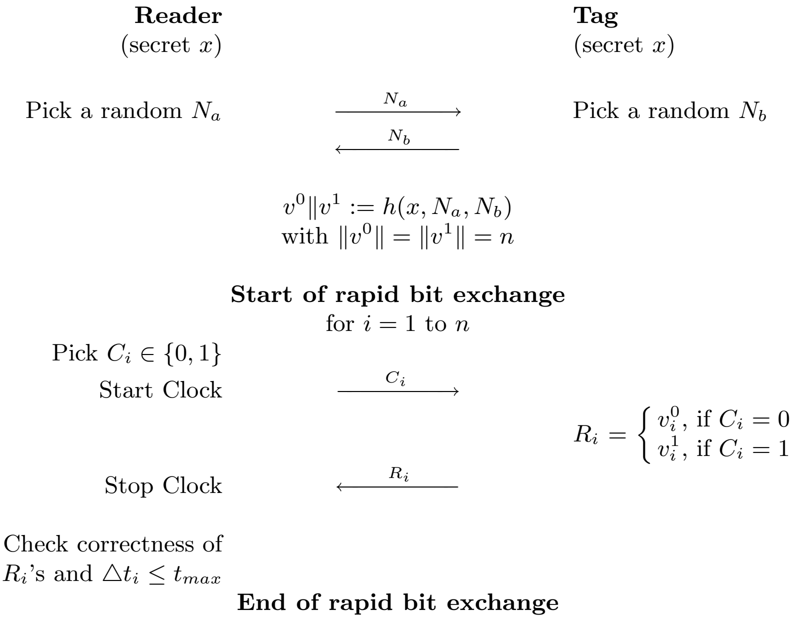

Reader
(secret x)
Tag
(secret x)
Pick a random N_a
N_a
Pick a random N_b
N_b
$v^0 \| v^1 := h(x, N_a, N_b)$
with $\|v^0\| = \|v^1\| = n$
Start of rapid bit exchange
for $i = 1$ to n
Pick $C_i \in \{0, 1\}$
Start Clock
C_i
$R_i = \begin{cases} v_i^0, \text{ if } C_i = 0 \\ v_i^1, \text{ if } C_i = 1 \end{cases}$
Stop Clock
R_i
Check correctness of
R_i's and $\triangle t_i \leq t_{max}$
End of rapid bit exchange

Fig. 2. Hancke and Kuhn's protocol

Then (and repeating this step n times) the reader sends a random bit. Upon receiving a bit, the tag sends back a bit from v^0 if the received bit C_i equals 0. If C_i equals 1, then it sends back a bit from v^1. After n iterations, the reader checks the correctness of R_i's and computes the propagation time. In each round, the probability that adversary sends a correct response is not $\frac{1}{2}$ but $\frac{3}{4}$. This is because the adversary could slightly accelerate the clock signal provided to the tag and transmit an anticipated challenge C'_i before the reader sends its challenge C_i. In half of all cases, the adversary will have the correct guesses, that is $C'_i = C_i$, and therefore will have obtained in advance the correct value R_i that is needed to satisfy the reader. In the other half of all cases, the adversary can reply with a guessed bit, which will be correct in half of all cases. Therefore, the adversary has $\frac{3}{4}$ probability of replying correctly.

Since this protocol's publication, several solutions have been proposed to improve its effectiveness and/or enhance its functionalities.

A solution to reduce the aforementioned probability below $\frac{3}{4}$ is to include a signing message (or message authentication code) as used in other protocols [2,14,15]. However a signing message could not be sent with the channel for fast bit exchanges as it is very sensitive to the background noise. It should be sent by normal communication method with error detection or correction technique. Therefore this approach would put an overhead on computation of a tag as well as communication, which causes the protocol to be slower.

In 2006, Munilla et al. modified the Hancke and Kuhn protocol by applying "void challenges" in order to reduce the success probability of the adversary

[10]. Their protocol is the first and only approach not using any additional signing message to reduce the success probability of the adversary. However the disadvantage of their solution is that it requires three (physical) states: 0, 1, and *void*, which is practically very difficult to implement.

HKP is vulnerable to the terrorist fraud attack but this can be solved, as proposed by Reid et al. in [13], making the bit-strings, v^0 and v^1, and the long-term key x intermingled ($v^0 = Enc_{v^1}(x)$). Thus, if a legitimate tag wants to reveal the secret, then it will allow the adversary to impersonate it in more than a single run of the protocol. However, Reid et al.'s protocol does not provide privacy as it sends identities without any protection. Furthermore, as described by Piramuthu [12], the probability of the success for an attack is higher than for HKP.

In 2007, Tu and Piramuthu proposed a protocol to reduce the success probability of an adversary [15]. They used four for-loop iterations for fast bit exchanges, which made the success probability of an adversary equal to $(9/16)^n$. That is, the reader sends different hash values after $n/4$-bit exchanges. They also used the combination of "$v^0 = v^1 \oplus x$" to prevent terrorist fraud attack. However, we will show in Section 3 that their protocol is in fact not secure against an active adversary.

Capkun et al. extended Brands and Chaum's protocol to mutual authentication, so called MAD (mutual authentication with distance-bounding) in 2003 [4]. However their protocol is not resilient to bit errors during the fast bit exchanges.

Singelée and Preneel proposed a noise resilient protocol in 2007 [14]. They used error correcting code (ECC) and MAC for the sake of channel error resistance, but this made the protocol slower.

Recently Nikov and Vauclair proposed a protocol [11]. They used more than a bit for fast exchanges. However it is susceptible to channel noise. Furthermore the tag needs to compute $2k$ secret key functions (HMAC or AES) and store the result.

Finally, Waters and Felten [16] and Bussard and Bagga [3] proposed distance bounding protocols using public key cryptography respectively. However the adoption of public key cryptography in a small device such as low-cost RFID is not applicable yet.

3 New Attack on Tu and Piramuthu Protocol

We show in this section an attack against the protocol described by Tu and Piramuthu [15].

3.1 The Protocol

The protocol is depicted in Figure 3 (for convenience, we use the same notations as in the original paper). The Reader first generates a nonce r_B and sends $x \oplus r_B$ to the tag, where x is a shared long-term secret. Similarly, the tag generates a random nonce r_A and sends $x \oplus r_A$. The reader and the tag then derive a common session key $k = h(r_A, x||r_B)$ and use this key to split the secret x in two shares,

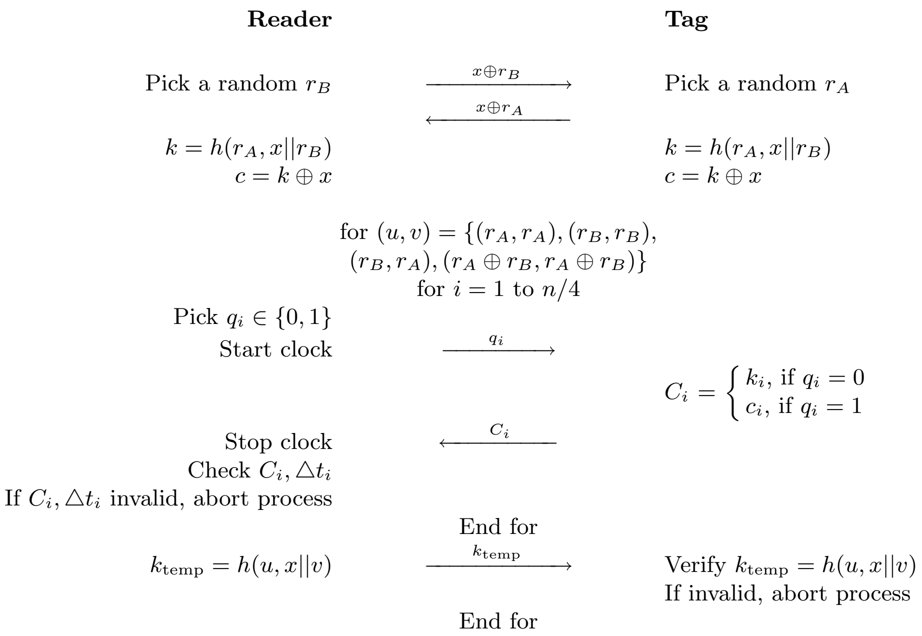

Reader
Tag
Pick a random r_B
$x \oplus r_B$
Pick a random r_A
$x \oplus r_A$
$k = h(r_A, x||r_B)$
$c = k \oplus x$
$k = h(r_A, x||r_B)$
$c = k \oplus x$
for $(u, v) = \{(r_A, r_A), (r_B, r_B), (r_B, r_A), (r_A \oplus r_B, r_A \oplus r_B)\}$
for $i = 1$ to $n/4$
Pick $q_i \in \{0, 1\}$
Start clock
q_i
$C_i = \begin{cases} k_i, \text{ if } q_i = 0 \\ c_i, \text{ if } q_i = 1 \end{cases}$
C_i
Stop clock
Check $C_i, \triangle t_i$
If $C_i, \triangle t_i$ invalid, abort process
End for
$k_{\text{temp}} = h(u, x||v)$
k_{temp}
Verify $k_{\text{temp}} = h(u, x||v)$
If invalid, abort process
End for

Fig. 3. Tu and Piramathu's protocol

k and $c = k \oplus x$. Then an outer loop is iterated four times, for four different combination values of (u, v), namely $(r_A, r_A), (r_B, r_B), (r_B, r_A), (r_A \oplus r_B, r_A \oplus r_B)$. During each iteration of this outer loop, an inner loop is iterated $n/4$ times. The inner loop is a rapid bit exchange consisting in a challenge bit q_i being sent by the reader and the corresponding answer C_i being sent by the tag, where $C_i = k_i$ (resp. $C_i = k_i \oplus x_i$) if the challenge was equal to 0 (resp. 1). After this inner loop, a reader verification is performed by the tag by having the reader compute and transmit a value $k_{\text{temp}} = h(u, x||v)$, which is then verified by the tag (this verification step is thus performed 4 times, one at each iteration of the outer loop). The idea of this verification step is to have the reader validated by the tag at intermediary steps of the rapid bit exchange, in order to prevent an adversary from sending queries (random) q_i's and retrieving corresponding C_i's in advance. According to [15], this step can also use bit streaming and clocking to measure (on tag's side) the distance between tag and reader.

3.2 Our Attack

We show an attack allowing an adversary to recover the long-term key x. To learn bit x_i of that key, the attacker can, during the fast bit exchange, toggle the value of challenge bit q_i when it is transmitted from reader to tag and leave all other messages untouched. The attacker then observes the reader's reaction. As a matter of fact, if the reader accepts the tag, it means that the tag's answer C_i was nevertheless correct, and thus that $c_i = k_i$. As $c_i = k_i \oplus x_i$, the adversary can conclude that $x_i = 0$. Similarly, if the reader refuses the tag, the adversary can conclude that $x_i = 1$.

4 Proposed Scheme

4.1 Adversary

We consider an active adversary who entirely controls the channel. That is, she can eavesdrop, intercept, modify or inject messages. She can also increase the transmission speed on the channel up to a given bound. We define this bound as the speed of light. On the other hand, we consider that our adversary cannot correctly encrypt, decrypt, or sign messages without knowledge of the appropriate key. We assume that she has no way to obtain such keys except those of colluding tags.

We assume that the communication protocols are public, enabling an adversary to potentially communicate with a reader or a tag. While communicating with a tag, the adversary is able to increase or decrease its clock frequency and thus the computation speed.

We define a *neighborhood* as a geographical zone around a reader whose limits are clearly defined and publicly known. We consider that a tag present in a neighborhood agrees to authenticate. We say that a tag T has been *impersonated* if an execution of the protocol convinced a reader that it has authenticated T while the latter was not present inside the neighborhood during the said execution. In the same vein, a reader can be impersonated.

4.2 Goals

Authentication. The primary goal of the protocol is to ensure tag authentication, that is, at the end of the execution of the protocol, the reader gets the conviction that it communicates with the claimed entity. Mutual authentication is achieved if the tag also gets the conviction that it communicates with the claimed reader.

Mafia fraud attack resistance. A tag cannot be impersonated, except if it colludes with the adversary.

Terrorist fraud attack resistance. A tag cannot be impersonated, except if it reveals its secret key to the adversary.

Low computation complexity. In order to get a practical authentication delay, the number of cryptographic operations performed by the tag during the authentication process must be as small as possible. Due to their efficiency compared to asymmetric cryptography, the use of symmetric primitives is certainly desirable in this respect.

When several tags are present in the field of the reader, each tag must be singulated through a collision-avoidance protocol before starting the authentication protocol. During this process when tags are powered but mostly idled, or whenever tags can be powered without having to authenticate, they are able to perform some precomputation “for free”.

Low false acceptance rate. In order to get a practical authentication delay, the number of rounds of the fast phase and the total number of bits exchanged between the tag and the reader must be kept as small as possible for a given false acceptance rate. For accuracy reason on the round trip time, we assume that only one bit can be included per message in the fast phase.

Privacy. The protocol should not reveal the tag identifier except to the legitimate reader.[1] Moreover, given any set of recorded protocol executions, only the legitimate reader should be able to determine whether a tag is involved in two or more executions.

Channel error resistance. We assume that the channel used during the slow phase is error-free. This assumption is quite realistic in the sense that there is no specific time-constraint on that channel. An error-correcting mechanism can therefore be used.

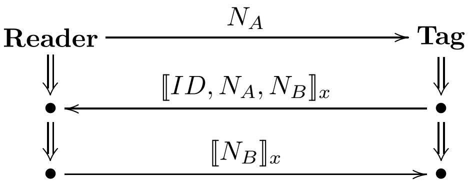
However, the channel used during the fast phase may suffer from Byzantine errors. In such a case, the authentication property must be ensured up to a given error rate threshold. Above this threshold, the reader must abort the protocol.

4.3 Description

Our authentication protocol is based on the MAP1 protocol of Bellare and Rogaway [1], in the MAP1.1 variant proposed by Guttman et al. [6]. To this protocol, which provides mutual authentication, a rapid bit exchange step has been added in order to achieve distance-bounding, and some cleartext information has been removed to ensure the privacy of the tag.

The MAP1.1 protocol works as follows. Tag and reader are assumed to share a secret key x.

1. The reader chooses a random nonce N_A and transmits it to the tag.
2. The tag chooses a random nonce N_B and transmits $[\![ID, N_A, N_B]\!]_x$ to the reader, where $[\![m]\!]_x$ means $m \| f_x(m)$, f is a pseudorandom function (PRF), x is the key of the tag, and ID is the concatenation of the reader's and tag's identifier.
3. The reader computes $[\![N_B]\!]_x$ and transmits it to the tag. Note that this extra step is only required if mutual authentication must be achieved. If it is not necessary for the tag to authenticate the reader, this last step can be discarded.



[1] Note that this is in fact not the most stringent notion of privacy that can be considered. A stronger notion, in which even the reader does not learn the identity of the tag, can also be useful in some contexts.

Taking this protocol as our starting point, four adaptations are proposed.

1. Since tags do not make any difference between the readers, we simply represent the identity of teh readers as the empty string.
2. Since we want to preserve the privacy of the tags, we do not transmit their identity in clear. The reader will then need to access a database storing the identity and key (x, ID) of each tag and to perform an exhaustive search over this DB, trying all possible keys until a match is found.[2]
3. Since honest tags and readers are not involved in concurrent sessions, we can also avoid repeating the transmission of nonces in clear after their initial transmission, so N_A (resp. N_B) does not need to be transmitted in clear during the second (resp. third) round.
4. Since we want our protocol to be distance bounding, a rapid bit exchange phase is added. The role of this phase is to prove to the reader that it is directly interacting with the tag, preventing relay attacks.

It can be observed that the first three adaptations do not have any impact on the authentication properties of the protocol, keeping the analyzes of [1,6] valid. The last adaptation will be designed in such a way that it does not interfere with the other parts of the protocol.

Basic version. We first describe a basic version of our protocol and discuss its security. A more efficient variant is discussed in Section 5.2.

First a preparation phase is performed, involving the generation of nonces, one application of the PRF and a few XORs. We will discuss below how precomputing can be used for low-resource devices. No delays are measured during this phase.

- Following the MAP1.1 protocol, the reader chooses a random nonce N_A and transmits it to the tag.
- The tag chooses a random N_B and computes a temporary key $a := f_x(C_B, N_B)$ using its permanent secret key x and N_B (here C_B is just a system-wide constant).
- The tag splits his permanent secret key x in two shares by computing $Z^0 := a, Z^1 := a \oplus x$.
- The tag transmits N_B to the reader (which constitutes the first part of the second message of the MAP1.1 protocol).

After this preparation, the rapid bit exchange phase starts. This phase is repeated n times, with i varying from 1 to n, and the challenge-response delay is measured for each step. As explained in Section 2.2, this communication goes over a channel that does not contain any error detection or correction mechanism, so we must take into account the fact that channel errors might occur (either randomly or by action of the attacker) in this phase. Moreover, the protocol

[2] Note that the protocol is always initiated on reader's side, so that a reflection attack, in which a genuine answer from one execution of the protocol is used by an attacker in another one, is not possible here.

must involve as few tag operations as possible in this phase, and we make this number of operations fairly minimal: in each round, the tag only needs to select one out of two pre-computed bits.

- The reader chooses a random bit c_i, starts a clock and transmits c_i to the tag. We will denote by c'_i the (possibly incorrectly transmitted) value received by the tag.
- The tag answers by $r'_i := Z_i^{c'_i}$. We will denote by r_i the value received by the reader.
- Upon receiving r_i, the reader stops the clock, stores the time delay $\triangle t_i$ and answer received (note that answer's correctness is not checked at this time), and moves to the next step.

After the rapid bit exchange phase, the final phase begins. This phase also involves significant computing overhead, and no delays are measured during it.

- The tag computes $t_B := f_x(c'_1, \dots, c'_n, ID, N_A, N_B)$ and transmits t_B and the challenges $c'_1, \dots, c'_n$ it received during the rapid bit exchange phase. Together with the sending of N_B that took place earlier, this is the second round of the MAP1.1 protocol, with the addition that t_B also authenticates the challenges $c'_1, \dots, c'_n$ received during the rapid bit exchange phase.
- The reader performs an exhaustive search over its tag database until it finds a pair (ID, x) such that $t_B := f_x(c'_1, \dots, c'_n, ID, N_A, N_B))$.
- The reader computes the values Z^0 and Z^1.
- The reader checks the validity of the responses made during rapid bit exchange phase, i.e.:
  - it counts the number err_c of positions for which $c_i \neq c'_i$;
  - it counts the number of positions err_r for which $c_i = c'_i$, but $r_i \neq Z_i^{c_i}$;
  - it counts the number of positions err_t for which $c_i = c'_i$ and $r_i = Z_i^{c_i}$, but the response delay $\triangle t_i$ is above the time threshold t_{max};
  - if $err_c + err_r + err_t$ is above the fault tolerance threshold T, authentication fails and the protocol aborts.
- The reader computes $t_A := f_x(N_B)$ and transmits it to the tag (this is the last step of the MAP1.1 protocol).
- The tag checks the correctness of t_A. As stated before, the last two steps are only required if mutual authentication must be achieved. If it is not necessary for the tag to authenticate the reader, they can be omitted. Still, in both cases, privacy is guaranteed.

5 Analysis

5.1 Security

To make the security discussion easier, let us first ignore the fault tolerance parameter. That is, we will consider that the threshold T is 0. The effect on security of a larger threshold will be discussed in Section 5.3.

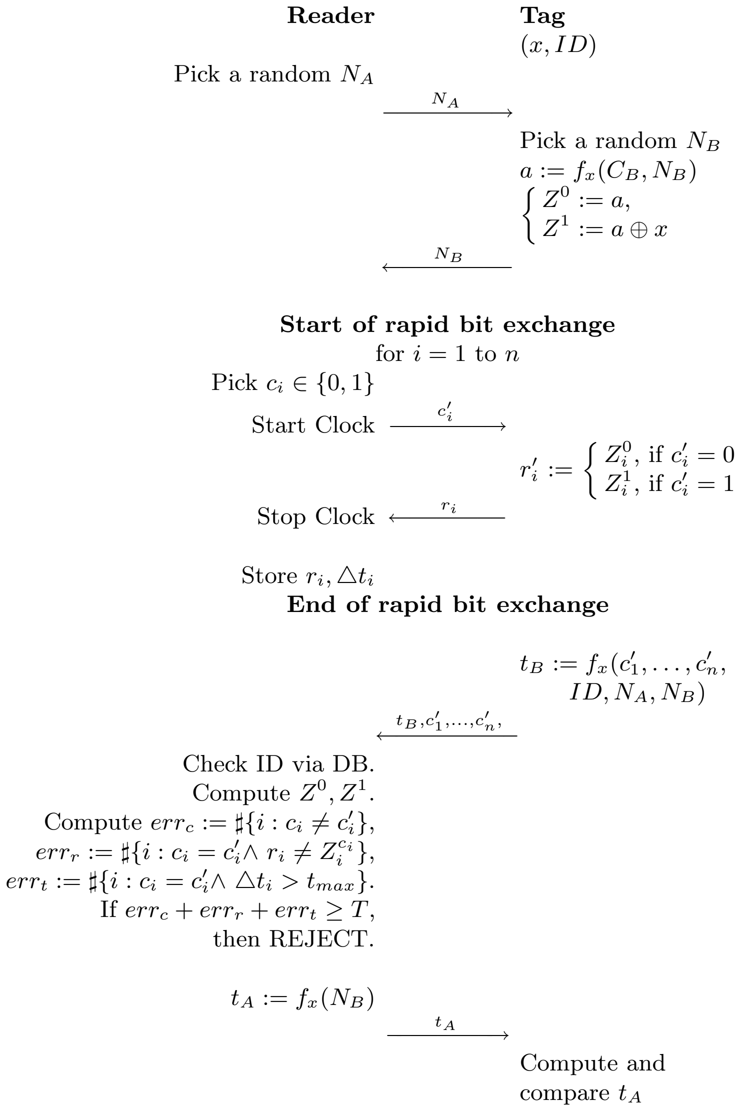

Reader
Tag
(x, ID)
Pick a random N_A
N_A
Pick a random N_B
$a := f_x(C_B, N_B)$
$\begin{cases} Z^0 := a, \\ Z^1 := a \oplus x \end{cases}$
N_B
Start of rapid bit exchange
for $i = 1$ to n
Pick $c_i \in \{0, 1\}$
Start Clock
c'_i
$r'_i := \begin{cases} Z^0_i, \text{ if } c'_i = 0 \\ Z^1_i, \text{ if } c'_i = 1 \end{cases}$
Stop Clock
r_i
Store $r_i, \triangle t_i$
End of rapid bit exchange
$t_B := f_x(c'_1, \ldots, c'_n, ID, N_A, N_B)$
$t_B, c'_1, \ldots, c'_n,$
Check ID via DB.
Compute Z^0, Z^1.
Compute $err_c := \sharp\{i : c_i \neq c'_i\}$,
$err_r := \sharp\{i : c_i = c'_i \wedge\ r_i \neq Z^{c_i}_i\}$,
$err_t := \sharp\{i : c_i = c'_i \wedge\ \triangle t_i > t_{max}\}$.
If $err_c + err_r + err_t \geq T$,
then REJECT.
$t_A := f_x(N_B)$
t_A
Compute and
compare t_A

Fig. 4. Our basic authentication protocol secure against relay attacks

Authentication. The security of the basic authentication protocol has been studied in [1] and [6]. Basically, the presence of N_A – a fresh nonce generated by the reader – in the input of the PRF f authenticates the tag to the reader (as only the tag and the reader know the value x used to key f). Similarly, the presence of N_B as an argument of f_x in $f_x(N_B)$ guarantees the tag that it was successfully authenticated by the reader. We refer to the aforementioned papers for more details.

Let us show that our modifications to MAP1.1 do not modify the security of the protocol. We did the following modifications to MAP1.1:

- some values that were transmitted in clear in MAP1.1 (e.g., the tag and reader ID) are not transmitted anymore;
- the values $(c_1, \ldots, c_n)$ are additionally transmitted in clear during the rapid bit exchange phase, and included as additional argument of the function f.

It is obvious that removing the cleartext transmissions from the protocol cannot harm security.

The $(c_1, \ldots, c_n)$ bits do not depend on any secret parameter involved in the protocol, and they are transmitted in the authenticated text transmission mode proposed for the MAP1 protocol [1].

Considering the rapid bit exchange, we will first argue that a passive observation of the rapid bit exchange does not reveal any information on x. If we consider the PRF as a random oracle, a can be seen as a pure random string, and the construction of Z^0, Z^1 is a classical secret sharing of x. As only one share of each bit is revealed during the rapid bit exchange, no information on x is disclosed by the responses r_i. Besides, no information on x can be revealed by the challenges c_i, as these do not depend on x.

Things get a bit different when we consider an *active* adversary, allowed to manipulate the messages exchanged during the rapid bit exchange. As in the attack described in Section 3, if the reader checked for the correctness of r_i without verifying the value of the challenge bits used by the tag (using t_B), one easy attack would be for the adversary to flip one bit c_i during transmission from reader to tag, and then to simply forward the answer of the tag to the reader. If the authentication is successful, the attacker can conclude that $Z_i^0 = Z_i^1$ and hence that $x_i = 0$; if the authentication is unsuccessful, she can similarly conclude that $x_i = 1$. This is the reason why we authenticate the c_i' and place the verification steps after the reception and verification of t_B. More precisely, we protect against active attackers by ensuring that verification is only performed on bits for which the challenge was not manipulated by the tag. In this way,

- tampering with c_i does not reveal any additional information, as the corresponding answer will simply be ignored,[3] and
- of course, tampering with answer r_i does not reveal any information either.

Tampering with the messages can of course make the protocol fail by DoS, but, as the attacker anyway knows he is always turning a correct answer into an incorrect one, he cannot gain any information in this way. We have thus showed that authentication is well achieved by our protocol.

[3] Of course, the adversary will be able to choose which share, Z_i^0 or Z_i^1 will be revealed there, but we already showed that obtaining only one share does not compromise the secret. Also note that, although the adversary is able to modify the challenges c_i during the rapid bit exchange phase, the presence of t_B prevents her from reflecting this modification in the message sent by the tag, so that the reader will learn which values were incorrectly received.

Terrorist fraud resistance. We will show that only a tag knowing at least $n - v$ bits of the long-term key x is able to answer the requests issued during the rapid bit exchange phase with success probability at least $(\frac{3}{4})^v$. As discussed in Section 4.1, we will consider that only a tag present in the neighborhood of the reader is able to answer the requests c_i in due time (without the possibility to obtain assistance from any device not present in the neighborhood), and show that this tag must know x. Considering that, at step i of the protocol, the tag will have to respond Z_i^0 with probability $\frac{1}{2}$ and Z_i^1 with probability $1/2$, a tag ignoring the value of v bits of Z^0 or Z^1 can only succeed with probability $(\frac{3}{4})^v$. From the equalities $Z^0 = a, Z^1 = a \oplus x$, the knowledge of $2n - v$ bits of Z^0, Z^1 immediately yields the knowledge of at least $n - v$ bits of x. This security bound is for example reached if we consider a terrorist attack in which a genuine, but distant, tag transmits the value of Z^0 to his accomplice: although no bit of x has been revealed to the accomplice, she has now $\frac{3}{4}$ chance to answer correctly on each step of the rapid bit exchange.

Mafia fraud resistance. It is worth noting that, as opposed to the Hancke and Kuhn protocol, the attack described in Section 2.2 (anticipated challenge transmission) does not work against our protocol. As a matter of fact, the presence in t_B of the list of challenges received by the genuine tag prevents this attack in our context. Therefore, we argue that the security bound against mafia attacks is $(\frac{1}{2})^n$.

Privacy. We first show that an adversary who does not know the value of x does not learn any information on ID. The only message depending on ID is $t_B = f_x(c_1, \ldots, c_n, ID, N_A, N_B)$. It is easy to show that any adversary who could retrieve any information on the input ID could also distinguish f_x from a random function. As f is supposed to be a PRF, we can conclude that no information on ID is revealed by the protocol. As far as tracking between various protocol executions is concerned, the values $N_B, r_0, \ldots, r_n, t_B, c_0, \ldots, c_n$ observable by an attacker are either random or the output of a PRF with fresh input and unknown key. It is thus clear that all of them appear as random values to the attacker, and tracking is not possible.

5.2 A More Efficient Variant

Our protocol, in the version we just presented, involves one rapid bit exchange step per bit of the key. In a constrained environment, this communication overhead might be problematic. On the other hand, we cannot for example restrict the rapid bit exchange to the first m bits of Z^0, Z^1, because we would then loose resistance against terrorist attack: a device knowing only the first m bits of the key x would be able to succeed in the rapid bit exchange phase.

This problem could be solved by having the reader challenge the tag on m random bit positions of Z^0, Z^1. However, although very simple, fetching a specific bit from an integer might be a too complex operation for the minimal overhead we expect during the rapid bit exchange phase.

Yet, there is a possible compromise. Basically, the reader could send a list of m bit positions (but not the corresponding challenges) to the tag before the rapid bit exchange phase, enabling the extraction of these positions from Z^0 and Z^1 and preserving a fast treatment during the phase itself. Of course, in a terrorist attack, the genuine device could take advantage of that delay to transmit only the relevant parts of Z^0, Z^1 to his accomplice, revealing her only m bits of x. Yet, as the list changes for each authentication, m random bits of x would have to be revealed for each authentication, so that the full key would be quickly revealed.

The full protocol is depicted in Figure 5.2. Below we only describe the part that changed compared to the initial protocol.

– The reader chooses a random N_A; it also chooses a random d with hamming weight m. Intuitively, d corresponds to a mask pointing the positions on which the tag will be questioned during the rapid bit exchange. The reader transmits N_A and d to the tag.
– The tag chooses a random N_B and computes $a := f_x(C_B, N_B)$, Z^0 and Z^1 as before. Then it prepares the possible answers by extracting the relevant parts of Z^0, Z^1 according to the mask d, building the m-bit vectors R^0 and R^1.
– The remainder of the protocol is unchanged, except that R^0, R^1 is used instead of Z^0, Z^1.

It can be observed that the N_A and d protocol parameters could actually be merged to save bandwidth, by requiring the tag to use N_A as a mask. However, as the hamming weight of m must be fixed to some value (or range of values) in order to guarantee the appropriate security level for the rapid bit exchange, the set of admissible nonces of n bits is reduced, and the length of N_A must be increased accordingly.

5.3 Fault Tolerance

Our protocol is tolerant to faults occurring during the rapid bit exchange transmissions:[4] if some of the bits c_i, r_i get corrupted during transmission, or get inappropriately delayed, authentication can succeed anyway, provided the percentage of such errors is sufficiently small. Basically the threshold T must be chosen so that the probability for an adversary to be successful on $m - T$ challenges is acceptably small. Taking our most pessimistic context, i.e. terrorist fraud attacks, the chances of success follow a binomial distribution $A := \mathrm{Bi}(m, \frac{3}{4})$ and we want $\Pr[A > m - T] < \epsilon$ for an appropriate security parameter ϵ. For example, taking $m = 30$ rapid bit exchange steps and tolerating up to two errors (i.e. $T = 3$) yields a success probability for the adversary of about 1%. If we consider only resistance against mafia fraud attacks (so $A := \mathrm{Bi}(m, \frac{1}{2})$) and take $m = 20$ rapid bit exchange steps, then we can tolerate up to 4 errors and still have less than 1% fraud probability (or tolerate only one error and have the probability shrink to 0.01%).

[4] As stated before, we assume that other transmissions occur over a channel capable of error detection.

Reader | **Tag**

(x, ID)

Pick a random N_A

Pick a random d s.t. $H(d) = m$

$\xrightarrow{N_A, d}$

Pick a random N_B

$a = f_x(C_B, N_B)$

$$\begin{cases} Z^0 = a, \\ Z^1 = a \oplus x \end{cases}$$

for $i = 1$ to m

$$\begin{cases} j : \text{index of the next 1 of } d_{(2)}, \\ R_i^0 = Z_j^0,\ R_i^1 = Z_j^1 \end{cases}$$

$\xleftarrow{N_B}$

Start of rapid bit exchange

for $i = 1$ to m

Pick $c_i \in \{0, 1\}$

Start Clock $\xrightarrow{c_i'}$

$$r_i := \begin{cases} R_i^0, \text{ if } c_i' = 0 \\ R_i^1, \text{ if } c_i' = 1 \end{cases}$$

Stop Clock $\xleftarrow{r_i}$

Check: $\triangle t_i \leq t_{max}$

End of rapid bit exchange

$t_B := f_x(c_1', \ldots, c_n', ID, N_A, N_B)$

$\xleftarrow{t_B, c_1', \ldots, c_n',}$

Check ID via DB.

Compute R^0, R^1.

Compute $err_c := \sharp\{i : c_i \neq c_i'\}$,

$err_r := \sharp\{i : c_i = c_i' \wedge r_i \neq Z_i^{c_i}\}$,

$err_t := \sharp\{i : c_i = c_i' \wedge$

$\triangle t_i > t_{max}\}$.

If $err_c + err_r + err_t \geq T$

then REJECT.

$t_A := f_x(N_B)$

$\xrightarrow{t_A}$

Compute and compare t_A

Fig. 5. A more efficient variant of the protocol

5.4 Efficiency

Let us consider the amount of computation to be performed by the most constrained device involved in the protocol, i.e. the tag. The most time consuming part of the protocol is the computation of pseudo-random functions f. As shown in Fig. 4, the function f is used three times on the tag side: in computing a, t_B and t_A. If we need not mutual but unilateral authentication from tag to reader, we need just two computations, a and t_B.

As in [8], we can construct an RFID system which allows a precomputation in a tag. The contents of the input for the computation of a of our proposed protocol do not have any information from the reader. Therefore a can be computed before starting the protocol. Then we need two computations of pseudo-random functions to achieve mutual authentication (one if we need unilateral authentication only).

As it involves an exhaustive search in a key database, the workload on reader's side is significantly higher, and grows linearly with the number of keys deployed in the system. To the best of our knowledge, there is no existing method providing better performance without sacrificing some security, and this is certainly an interesting subject for further research. In particular, we note that most of the protocols discussed in Section 2.2 simply consider that the reader knows the identity of the tag it is questioning, or transmits this identity in clear during the protocol. Clearly, a variant of our scheme in which a single key is shared throughout the system would not have this computing overhead while still preserving privacy regarding outsiders (i.e. without knowledge of the key).

6 Protocols Comparison

Table 1 compares the proposed protocol with previous ones on several points of view: mafia fraud and terrorist attack resistance, error resistance, privacy preservation, mutual authentication and computational overhead inside the most restricted resource, i.e. the tag.

Regarding the security against mafia fraud attack, we compare the success probabilities for an adversary, in other words, false acceptance ratio against mafia fraud attack (M-FAR). This is the probability that the reader accepts the adversary as a legitimate tag. Although Reid et al. claim the M-FAR of their protocol to be $(3/4)^n$, Piramuthu later showed it to be equal to $(7/8)^n$.

The security against terrorist fraud attack and its success probability for an adversary (T-FAR) are compared in a similar way.

Then we compare the resilience against channel errors. This resilience is pretty important for protocol's robustness, as fast bit exchanges are typically sensitive to channel errors.

As far as privacy is concerned, Reid et al.'s protocol discloses identities in cleartext during protocol execution, and is thus not privacy-preserving. Most of the other protocols assume that the reader knows the identity and secret key of the tag before starting distance bounding protocol, hence ignoring the privacy issue or assuming a single secret is shared by all tags. Our protocol allows the reader to learn the tag's identity during execution, although we admit the corresponding overhead is pretty high, since an exhaustive search among all possible keys is necessary for this identification.

Table 1. Comparison of distance bounding protocols

	Mafia	M-FAR	Terrorist	T-FAR	Err. resis.	Privacy	MA	Comp.
BC [2]	Yes	$(1/2)^n$	No	-	No	-	No	2
HK [7]	Yes	$(3/4)^n$	No	-	Yes	-	No	1
Reid et al. [13]	Yes	$(7/8)^n$	Yes	$(3/4)^v$	Yes	No	No	2
SP [14]	Yes	$(1/2)^n$	No	-	Yes	-	No	1 + ECC
Capkun et al. [4]	Yes	$(1/2)^n$	No	-	No	-	Yes	4
NV [11]	Yes	$(1/2)^n$	No	-	No	-	No	$2k$
Proposed (MA)	Yes	$(1/2)^n$	Yes	$(3/4)^v$	Yes	Yes	Yes	3 (2)
Proposed (no MA)	Yes	$(1/2)^n$	Yes	$(3/4)^v$	Yes	Yes	No	2 (1)

We measure the amount of computation needed in the tag as the required number of computation of pseudo-random functions such as hash functions, symmetric key encryptions, etc.[5] We propose our protocol in two flavors, with and without mutual authentication. The number of computations of our protocol is three with mutual authentication and two without it. Additionally, one of these values can be pre-computed in each case (the values between parentheses in Table 1 refer to the number of computations that must be computed on-line).

References

1. 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)
2. Brands, S., Chaum, D.: Distance-Bounding Protocols. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 344–359. Springer, Heidelberg (1994)
3. Bussard, L., Bagga, W.: Distance-bounding proof of knowledge to avoid real-time attacks. In: IFIP/SEC (2005)
4. Capkun, S., Buttyan, L., Hubaux, J.-P.: SECTOR: secure tracking of node encounters in multi-hop wireless networks. In: 1st ACM Workshop on Security of Ad Hoc and Sensor Networks – SASN 2003, pp. 21–32 (2003)
5. Desmedt, Y.: Major security problems with the "Unforgeable" (Feige)-Fiat-Shamir proofs of identiy and how to overcome them. In: SecuriCom 1988, pp. 15–17 (1988)
6. Guttman, J.D., Thayer, F.J., Zuck, L.D.: The faithfulness of abstract protocol analysis: Message authentication. Journal of Computer Security 12(6), 865–891 (2004)
7. Hancke, G., Kuhn, M.: An RFID distance bounding protocol. In: The 1st International Conference on Security and Privacy for Emergin Areas in Communications Networks (SECURECOMM 2005), pp. 67–73. IEEE Computer Society, Los Alamitos (2005)
8. Hofferek, G., Wolkerstorfer, J.: Coupon recalculation for the GPS authentication scheme. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 162–175. Springer, Heidelberg (2008)

[5] We note that Singelée and Preneel's protocol (SP) requires additional error correcting codes (ECC), which normally requires significant computation overhead.

9. Munilla, J., Peinado, A.: Distance bounding protocols with void-challenges for RFID. In: Workshop on RFID Security - RFIDSec 2006 (2006)
10. Munilla, J., Peinado, A.: Distance bounding protocols for RFID enhanced by using void-challenges and analysis in noisy channels. Wireless communications and mobile computing (2008); published online: January 17, 2008, an earlier version appears in [9]
11. Nikov, V., Vauclair, M.: Yet another secure distance-bounding protocol, `http://eprint.iacr.org/2008/319`; an earlier version appears in SECRYPT 2008
12. Piramuthu, S.: Protocols for RFID tag/reader authentication. Decision Support Systems 43, 897–914 (2007)
13. Reid, J., Nieto, J.G., Tang, T., Senadji, B.: Detecting relay attacks with timing-based protocols. In: Bao, F., Miller, S. (eds.) Proceedings of the 2nd ACM symposium on Information, computer and communications security, pp. 204–213. ACM, New York (2007), `http://eprint.qut.edu.au/view/year/2006.html`
14. Singelée, D., Preneel, B.: Distance bounding in noisy environments. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 101–115. Springer, Heidelberg (2007)
15. Tu, Y.-J., Piramuthu, S.: RFID distance bounding protocols. In: The 1st International EURASIP Workshop in RFID Technology, Vienna, Austria (2007)
16. Waters, B., Felten, E.: Secure, private proofs of location. Princeton Computer Science, TR-667-03 (2003)

Protecting Location Privacy through a Graph-Based Location Representation and a Robust Obfuscation Technique

Jafar Haadi Jafarian, Ali Noorollahi Ravari, Morteza Amini, and Rasool Jalili

Sharif Network Security Center; Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
{jafarian@ce.,noorollahi@ce.,m_amini@ce.,jalili@}sharif.edu
http://nsc.sharif.edu

Abstract. With technical advancement of location technologies and their widespread adoption, information regarding physical location of individuals is becoming more available, augmenting the development and growth of location-based services. As a result of such availability, threats to location privacy are increasing, entailing more robust and sophisticated solutions capable of providing users with straightforward yet flexible privacy. The ultimate objective of this work is to design a privacy-preserving solution, based on obfuscation techniques (imprecision and inaccuracy), capable of handling location privacy, as required by users and according to their preferences. To this aim, we propose an intuitive graph-based location model, based on which users can express their regional privacy preferences. We present an obfuscation-based solution which allows us to achieve location privacy through degradation of location information, as well as measuring the reliability of such information. The proposed approach is robust and efficient, and covers some of the deficiencies of current obfuscation-based privacy solutions. We also propose two privacy-aware architectures for our solution.

Keywords: Location Privacy, Obfuscation Techniques, Reliability, Middleware Architecture, Distributed Architecture.

1 Introduction

With the emergence of low-cost accurate location sensing technologies and their widespread adoption, information regarding physical locations of individuals is becoming more available. The temptation to reveal location information, mainly aroused by services offered by location-based service providers (LBSPs), is compelling.

Leaked to an unscrupulous entity, information about physical location of individuals may be abused for stalking, physical harassment, or malicious commercial activities. Therefore, failure to protect location privacy of individuals can result in their reluctance to reveal such information or to use location-based services.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 116–133, 2009.

Location privacy is a special type of information privacy which concerns the claim of individuals to determine for themselves when, how, and to what extent location information about them is communicated to others [1,2]. Location privacy can be provided through *obfuscation*; i.e. deliberately degrading the accuracy and quality of information about an individual's location.

The ultimate objective of this work is to design an obfuscation-based technique able to handle location privacy, as required by users according to their preferences. To this aim, a graph-based location representation based on the concept of *region* is introduced. A region represents a spatial area characterized by a symbolic name, such as $22nd$ *Street*. The set of regions create a directed acyclic graph (DAG) in which nodes represent regions and edges represent containment relationships among them.

Privacy preferences of users are expressed in terms of regions and managed by DAG of regions. Specifically, to obfuscate an individual's location information, her measured location is mapped to a region and the region is obfuscated based on the privacy preferences specified by the user for that region.

Key to this work is the concept of *reliability* as the metric for the reliability (accuracy and precision) of location information. Reliability allows service providers to distinguish reliable location information from unreliable one, resulting in more appropriate and reliable service provision.

We also explain how our solution can be deployed on distributed and middleware-based architectures, and also describe the advantages and shortcomings of both approaches. Moreover, as a case study and to exemplify how our solution may interact with an LBSP, an integration of our middleware architecture with a location-based access control engine is presented. The case shows how reliability metrics provided by the solution can be exploited by an LBSP to provide more reliable services.

The remainder of this paper is organized as follows. Section 2 presents a background on categories and techniques of location privacy followed by related work. Section 3 introduces the way locations are handled in our solution. Section 4 illustrates our approach for defining location privacy preferences, introduces the concept of reliability, and presents our obfuscation technique. Section 5 presents two privacy-aware architectures for our solution. In section 6 we study how our solution can be exploited in satisfying the needs of a common LBSP such as a location-based access control engine. Section 7 addresses some of the key issues, challenges, and benefits in adoption of our solution, and section 8 concludes the paper and gives future work.

2 Background and Related Work

Although location information can be effectively used to provide enhanced location-based services, sensitivity of such information concern users about their privacy. Failure to protect location privacy allows exploitation of such information for unethical purposes such as stalking, physical harassment, or malicious commercial activities [3]. As a result of such threats, location privacy issues are the subject of growing research efforts.

Location privacy techniques can be partitioned into three main classes: anonymity-based, policy-based, and obfuscation-based [3,4]. The main branch of research on location privacy relies on anonymity-based techniques. Beresford and Stajano [5] present mix zones as a technique to provide location privacy managed by trusted middlewares. In this solution, each user has some unregistered geographical regions, called mix zones, where she cannot be tracked. Bettini et al. [6] present a framework capable of evaluating the risk of sensitive location-based information dissemination, and a technique aimed at supporting k-anonymity. Gruteser and Grunwald [7] define k-anonymity in location obfuscation, and present a middleware architecture and an adaptive algorithm to adjust location information resolution, in spatial or temporal dimensions, based on specified anonymity requirements.

Another branch of research on location privacy aims at protecting user privacy through the definition of complex rule-based policies [8,9]. Although suitable for privacy protection, such policies are often hard to understand and manage by end users.

The branch of research closest to our work aims at adoption of obfuscation techniques in protection of location privacy. In obfuscation-based techniques, instead of anonymizing user identities, location privacy is protected through intentional degradation of the accuracy of information regarding an individual's location. Duckhum and Kulic [3] define a distributed architecture that provides a mechanism for balancing individual needs for high-quality information services and location privacy. The model assumes a graph-based representation of the environment, and degrades location information by adding n nodes with the same probability of being the real user position. The main drawback of the model is that it ignores measurement errors and assumes the geographical coordinates of user's location is known with high precision and accuracy, which is hardly satisfiable by current technologies.

Other proposals rely on a trusted middleware, a gateway between location providers and LBSPs, for enforcing privacy on location information. Openwave [10] includes a location gateway that obtains users' information from multiple sources, enforces user preferences if available, and disseminates it to other parties. Bellavista et al. [11] proposes a middleware-based solution that is based on points of interest with symbolic location granularity (e.g. country, city). The main drawback of the model is its excessive rigidness in specification of privacy requirements of services, and its lack of adaptability to privacy preferences of users. Furthermore, the model does not take measurement errors and unreliability of location information into consideration.

Ardagna et. al [4,12,13,14,15] present a middleware-based privacy-enhanced location-based access control model, capable of managing location-based access control policies and enforcement of user privacy preferences. In order to take measurement errors into consideration, the solution assumes that location of users is measured as circular areas, the accuracy of which depends on the deployed technology. Users specify their privacy requirements through the definition of a *relative privacy preference*. The model introduces three obfuscation techniques;

namely enlarging the radius, shifting the center, and reducing the radius of user's circular location measurement. The techniques can also be mixed to provide more robust privacy. The solution also measures the accuracy of obfuscated information by introducing the concept of *relevance* as a adimensional metric for location accuracy. The main shortcoming of the solution is that it does not consider situations where a user might require different obfuscation levels in different locations. Also, efficient deployment of the solution requires users to gain some technological knowledge about geographical coordinates of locations.

3 Location Representation

In presenting the obfuscation algorithms, we adopt a discrete model of geographic space in which the world is a planar (2-D) coordinate space consisting of a non-empty set of unique areas called regions.

Definition 1. *(Region). A region is a unique spatial area identified with four elements: a symbolic label, a geographical representation, a region type, and a set of parent regions. Uniqueness property implies that no two regions can have the same symbolic labels or geographical representations.*

The symbolic label indicates the name by which the region is known in the real world; e.g. *22nd street*, *Sharif network security center*, etc. Figure 1 shows possible regions of a hospital.

The geographical representation of a region specifies the actual location of the region on the map. To represent the geographical location of a region, different approaches can be adopted. Regions might be represented as circles using a point and a radius, as rectangles using a pair of points, as polygons using a non-empty set of points, or even as a range of IP addresses. Furthermore, each point is a 2-dimensional coordinate which might represent a global location, such as latitude and longitude, or a location on a local map.

Region types indicate the nature or main functionality of a region; e.g. hospital, city, street, etc. For instance, the region type of all hospitals in the world may be selected as *hospital.*

Definition 2. *(Region type). A region type is a symbolic label describing a set of regions with identical properties.*

The set of parent regions of a region r determines the set of regions in which r is geographically contained (The containment relationship is shown by $\subseteq$). Note that, if r_i and r_j belong to the set of parent regions of r, then neither $r_i \subseteq r_j$ nor $r_j \subseteq r_i$. Specifically, if r_i and r_j contain r, and r_j contains r_i, then only r_i is included in the set.

Containment relationship between regions form a directed graph, $G = (V, E)$, in which V is the set of regions, and every edge (r, w) represents membership of w in the parent set of r (Figure 2). Since containment relationship is asymmetric and no two regions have a same geographical representation, graph G does not include any cycle, and therefore it is a directed acyclic graph (DAG). Moreover,

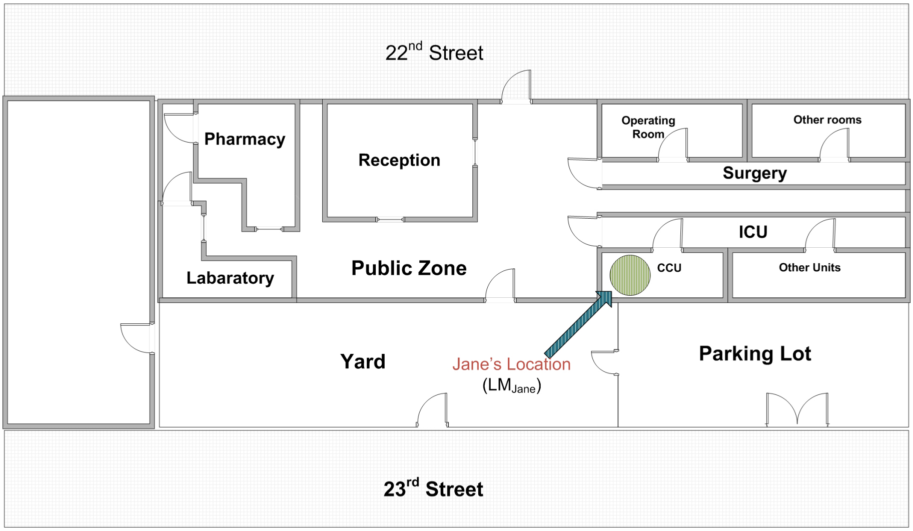


Fig. 1. Example of regions in a hospital

G contains no transitive edges, because according to the above restriction if r,u, $w \in V$ and (r,u),$(u,w) \in E$, then $(r,w) \notin E$.

4 Location Obfuscation and User Preferences

Location obfuscation is the means of intentionally degrading the accuracy of information about an individual's location information still maintaining association with the real user identity. Location obfuscation can be achieved by imposing deliberate imprecision on spatial information. Three distinct types of imperfection in spatial information are commonly identified in the literature: *inaccuracy*, *imprecision*, and *vagueness* [3]. Inaccuracy involves a lack of correspondence between information and reality; imprecision involves a lack of specificity in information; and vagueness involves the existence of boundary cases in information [16,17].

Potentially, any or all of these types of imperfection could be used to obfuscate an individual's location. For instance, for $User\ A$ located at the corner of $22nd\ Street$ and $23rd\ Street$, the following obfuscations of her location might be provided:

- *User A* is located at the corner of *22nd Street* and *21st Street* (inaccuracy)
- *User A* is in the city (imprecision)
- *User A* is near *22nd Street* (vagueness)

In this paper, we consider the use of imprecision and inaccuracy in location obfuscation. Deliberately introducing inaccuracy into location information raises serious questions, since it essentially requires the obfuscation system to lie about an individual's location. While supporting inaccuracy-based obfuscation, our solution provides a specific reliability metric which determines the level of inaccuracy introduced in the location information. Specifically, if the obfuscated region

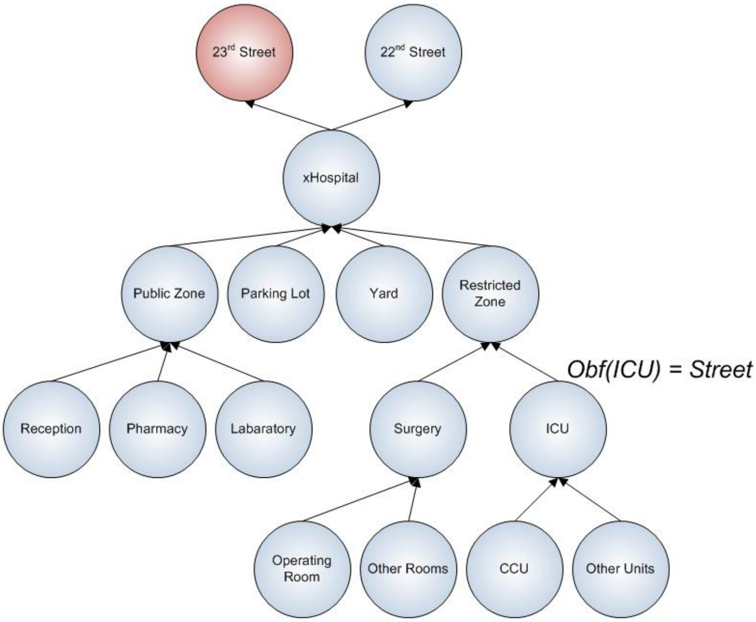


Fig. 2. Example of regions in a region graph

has no intersection with user's location measurement, the accuracy metric tends to zero, indicating the highest level of inaccuracy in spatial information.

4.1 Location Measurement

Location measurement of a user is the area returned as user's current location. Since the result of each location measurement is inevitably affected by errors, a spatial circular area is returned, rather than a single point. This assumption represents a good approximation for actual shapes resulting from many location technologies [8]; e.g. cellular phones location.

Definition 3. *(Location measurement). A location measurement of a user u, represented by LM_u, is a circular area $Circle(r, x_c, y_c)$, centered on the coordinate (x_c, y_c) and with radius r, which includes the real user's position (x_u, y_u) with probability $P((x_u, y_u) \in Circle(r, x_c, y_c)) = 1$ [15].*

Another assumption is that the probability distribution of points within an area or region is uniform. Specifically, the probability that a random point (x, y) belongs to a region is uniformly distributed. Formally, $f_r(x, y)$, the probability density function (pdf) of a region or location measurement r, is:

$$f_r(x, y) = \begin{cases} \frac{1}{s(r)} & (x, y) \in r \\ 0 & otherwise \end{cases}$$

where $s(r)$ represents the area of r.

The Same assumption can be found in other works [15,18] on this topic. Assuming a uniform distribution simplifies the model with no loss of generality.

4.2 User Preferences

There is always a trade-off between expressiveness and usability in expression of user preferences. Complex policy specifications, fine-grained configurations, and explicit technological details discourage users from fully exploiting the provided functionalities.

Many solutions presented in the literature [3,10] allow users to express privacy preferences using a minimum distance and radius enlargement techniques. The solution presented in [15] provides an intuitive variation of this technique by letting users specify a relative privacy preference λ; e.g. $\lambda = 0.2$ means 20% of degradation. While providing a simple policy specification and avoiding explicit technological details, the solution does not consider common cases where a user might require different obfuscation in different locations. For example, the privacy a doctor needs at hospital may differ from the privacy she needs at home. Also, accurate assignment of values to λ requires users to possess technological knowledge about these regions; e.g. in order to conceal her presence in restricted zone of a hospital and obfuscate it to the hospital, the doctor must have some knowledge about their geographical properties.

The objective of our solution is to let users specify their privacy preferences using symbolic labels of the regions and the containment relationship among them. Such solution avoids any technological complexity and makes policy specification straightforward. Since in practice, users usually require the same privacy preferences except for some specific regions (e.g. home, office), fine-grained configurations can be easily avoided using default preferences.

To this aim, we first introduce the function *obf* which allows users to determine their privacy preferences by mapping regions or region types to their obfuscated region/region type; e.g. $obf(Hospital) = Zone$ simply means that for this user, hospital regions must be obfuscated to their enclosing $Zones$. Therefore, four types of obfuscation are possible, mentioned here in order of priority: obfuscating a region to another region; obfuscating a region to a region type; obfuscating a region type to a region, and obfuscating a region type to another region type.

Obfuscating a region to another region. When region A is specified to be obfuscated to region B ($obf(A) = B$) by a user u, it simply means that if u is in A and her current location is queried, B will be returned.

Obfuscating a region to a region type. When region A is specified to be obfuscated to region type C ($obf(A) = C$) by a user u, it means that if u is in A and her current location is queried, the nearest ancestor of A with type C (including itself) is returned. If no such ancestor is found, A will be returned. If two qualified ancestors with the same distance to A are found, one of them will be randomly selected as the result (Figure 2).

Obfuscating a region type to a region. When region type C is specified to be obfuscated to region B ($obf(C) = B$) by a user u, it means that if u is in an arbitrary region A of type C and the region itself has no privacy preference ($obf(A) = \bot$), B is returned as the result of a query about current location of u.

Obfuscating a region type to another region type. When region type C is specified to be obfuscated to a region type D ($obf(C) = D$) by user u, it means that if u is in an arbitrary region A of type C, and A itself has no privacy preference ($obf(A) = \bot$), the nearest ancestor of A with type D (including itself) is returned. Again, if no such ancestor is found, A will be returned. Also, if two qualified ancestors with the same distance to A are found, one of them will be randomly selected as the result.

The following algorithm provides obfuscation using user preferences:

```
$ObfuscateRegion(region\ A)$
$\{returns\ a\ region\}$
  $t := A$
  if $obf(A) \neq \bot$ then $t := obf(A)$
  else if $obf(RgnType(A)) \neq \bot$ then $t := obf(RgnType(A))$
  if $t$ is a $region$ then
      return $t$
  else if $t$ is a $region\ type$ then
      $Queue\ q$
      $q.enqueue(A)$
      while $q$ is not emtpy
          $s := q.dequeue$
          if $RgnType(s) = t$ then
              return $s$
          foreach region $w$ in $ParentSet(s)$ in a random order
              $q.enqueue(w)$
  return $A$
```

The algorithm takes a region as input, and returns its corresponding obfuscated region according to user preferences. To this aim, firstly obfuscation preference of the user, t, is selected based on values of $obf(A)$, $obf(RgnType(A))$, and A. In case t is a region, the result is simply t. Otherwise, the nearest ancestor of A with type t must be chosen. To this aim, a breadth-first search started from A is used on the region graph.

Since privacy preferences of users are expressed in terms of regions, location measurement of the user must be mapped into a region. Before addressing this issue, we need to introduce reliability metrics of our solution.

4.3 Reliability

To evaluate the accuracy and precision of location information, the concept of reliability is introduced as the adimensional metric of both the reliability and the privacy of location information. The idea of such metrics has been taken from Ardagna et al.'s work [15]. A reliability metric μ is a value in $(0, 1]$ associated with location information, which depends on the measurement errors and privacy preferences of users. The reliability metric tends to 0 when location information must be considered unreliable for application providers; it is equal to 1 when location information has the best accuracy and precision; and a reliability value in $(0, 1]$ corresponds to some degree of reliability. Accordingly, the location privacy provided by an obfuscated location is evaluated by $(1 - \mu)$, since in our solution, privacy is the result of imprecision and inaccuracy. In our solution, all locations have an associated reliability attribute, from an initial location affected by a measurement error to all possible subsequent manipulations to provide privacy. Three reliability values are introduced in our solution:

- **Technological reliability (μ_{Tech}).** The metric for the reliability of a user location measurement as returned by the sensing technology. In [19], available sensing technologies such as *E-OTD* for *GSM*, *OTDOA* for *WCDMA*, and *Cell-ID* are introduced.
- **Mapping reliability (μ_{Map}).** The metric for the accuracy achieved in mapping user location measurement to a region.
- **Obfuscation reliability (μ_{Obf}).** The metric for the reliability of an obfuscated region and therefore the level of privacy provided to the users.

Among these reliability values, μ_{Tech} is assumed to be known. In [15] a technique for evaluating the reliability of the area returned by a sensing technology is proposed. μ_{Map} is calculated based on user location measurement and the region to which it is mapped. μ_{Obf} is derived from the privacy preferences of the user. In other words, μ_{Obf} represents the reliability of the obfuscated region that is calculated starting from the location measurement with reliability μ_{Tech} and by degrading its reliability according to the privacy preferences of user.

4.4 Obfuscation Technique

Since privacy preferences of users are specified in terms of regions, user location measurement must be mapped into a region. Obviously, the selected region is the one having the largest intersection with user location measurement, compared to other regions. If two regions have same intersection with location measurement in terms of area, either of them is contained in the other one, or there exists no containment relationship between them. In the former case, the contained region will be selected, while in the latter case, one of them will be randomly chosen.

The following algorithm takes user location measurement of a user u (shown by LM_u) and maps it into the desired region (called $region_u$). Note that $s(a)$ represents the area of a.

```
$function\ MapLMtoRegion(LM_u)$
$\{returns\ a\ region\}$
  $v :=\ \emptyset$
  $reg :=\ \emptyset$
  foreach region $r$ where $(r\ \cap\ LM_u\ \neq\ \emptyset)$
    $v'\ :=\ r\ \cap\ LM_u$
    if $s(v')\ >\ s(v)$ or $(s(v') = s(v)$ and $v' \subset\ v)$ then
```

```
        $v := v'$
        $reg := r$
    $Map := \frac{s(v)}{s(LM_u)}$
    return $reg$
```

The algorithm determines the desired region by computing the intersection area of all regions with LM_u. It also computes mapping reliability; i.e. μ_{Map}. In fact, μ_{Map} represents the probability that the real user's position belongs to the chosen region:

$$\mu_{Map} = P((x_u, y_u) \in (LM_u \cap region_u) | (x_u, y_u) \in LM_u)$$
$$\mu_{Map} = \frac{s(LM_u \cap region_u)}{s(LM_u)}$$

Since the obfuscation technique used in our solution is based on $region_u$, inaccurate mapping may have serious repercussions. Therefore, μ_{Map} is proposed as a reliability criterion for the LBSPs to distinguish and filter mapping errors based on a predefined threshold. In the strictest case, an LBSP might reject location information with μ_{Map} lower than 1; i.e. no mapping error is tolerated.

Having determined current region of the user, the next step is to apply its privacy preferences using the algorithm presented in 4.2. The returned obfuscated region is referred to as $region_{Priv}$. Obfuscation reliability, μ_{Obf}, is measured after the algorithm is applied.

Beginning from the reliability of user location measurement, μ_{Obf} represents the reliability (precision and accuracy) of an obfuscated region. To measure the obfuscation effect, two probabilities must be composed: i) the probability that the real user's position belongs to the intersection of LM_u and $region_{Priv}$, and ii) the probability that a random point selected from the whole obfuscated area belongs to the intersection.

$$\mu_{Obf} = P((x_u, y_u) \in (LM_u \cap region_{Priv}) | (x_u, y_u) \in LM_u)$$
$$P((x, y) \in (LM_u \cap region_{Priv}) | (x, y) \in region_{Priv}) \mu_{Tech}$$

Based on the placement of LM_u and $region_{Priv}$, the formula for obfuscation reliability changes:

if $LM_u \subseteq region_{Priv}$

$$P((x_u, y_u) \in (LM_u \cap region_{Priv}) | (x_u, y_u) \in LM_u) = 1$$
$$P((x, y) \in (LM_u \cap region_{Priv}) | (x, y) \in region_{Priv}) = \frac{s(LM_u)}{s(region_{Priv})}$$
$$\mu_{Obf} = \frac{s(LM_u)}{s(region_{Priv})} \mu_{Tech}$$

if $LM_u \supset region_{Priv}$

$$P((x_u, y_u) \in (LM_u \cap region_{Priv}) | (x_u, y_u) \in LM_u) = \frac{s(region_{Priv})}{s(LM_u)}$$
$$P((x, y) \in (LM_u \cap region_{Priv}) | (x, y) \in region_{Priv}) = 1$$
$$\mu_{Obf} = \frac{s(region_{Priv})}{s(LM_u)} \mu_{Tech}$$

if $LM_u \nsubseteq region_{Priv}$ and $LM_u \not\supset region_{Priv}$

$$P((x_u, y_u) \in (LM_u \cap region_{Priv}) | (x_u, y_u) \in LM_u) = \frac{s(LM_u \cap region_{Priv})}{s(LM_u)}$$
$$P((x, y) \in (LM_u \cap region_{Priv}) | (x, y) \in region_{Priv}) = \frac{s(LM_u \cap region_{Priv})}{s(region_{Priv})}$$

$$\mu_{Obf} = \frac{s(LM_u \cap region_{Priv})^2}{s(LM_u)s(region_{Priv})}\mu_{Tech}$$

Note that if the obfuscated region has no intersection with the user location measurement ($LM_u \cap region_{Priv} = \emptyset$), then μ_{Obf} will be equal to 0, showing maximum unreliability of location information.

5 Privacy-Aware Architectures

One typical approach in the design of privacy-aware architectures is to provide a location middleware acting as a trusted gateway between sensing technology and LBSPs. Such a component is in charge of managing all interactions with sensing technology and enforcing privacy preferences of users. Ardagna et al. in [13,14,15] present a privacy-aware middleware architecture.

On the other hand, some works such as [3] present a distributed obfuscation architecture which allows an individual to connect directly with an LBSP, without the need to use a broker or other intermediary. In such architectures it is assumed that the client device provides required information about the client's location.

Both architectures have their advantages and drawbacks. The former is less distributed, and therefore unsuitable for ad-hoc networks and pervasive environments. However, the burden of privacy management falls on the shoulder of location middleware, relieving the user device of the responsibility. The latter architecture is more lightweight and distributed which makes it compatible with mobile ad-hock networks and pervasive environments. But since the device is in charge of location sensing and managing privacy preferences of individuals, the devices require more resources such as energy and processing capability.

5.1 A Middleware Architecture

Our solution can be integrated with both architectures. Figure 3 shows our privacy-aware architecture based on a location middleware. Location middleware is in charge of low-level communications with the location provider and enforcement of both the privacy preferences of the user and measurement of location and mapping reliability needed by LBSPs.

In this approach, the region graph and user preferences are stored in the location middleware. User location measurement is supplied to location middleware by location providers, and location middleware is in charge of mapping location measurement to region, obfuscating the region and calculating mapping and obfuscation reliability metrics.

LBSPs communicate with location middleware. When an LBSP queries current location of a user, the middleware replies with a triple: $\langle user's\ current\ location, \mu_{Map}, \mu_{Obf}\rangle$. The location-based service provider might then compare μ_{Map} and μ_{Obf} to some predefined thresholds in order to decide about accepting or rejecting middleware's reply. In section 6 we explain how such architecture can be integrated with an LBSP such as a location-based access control engine.

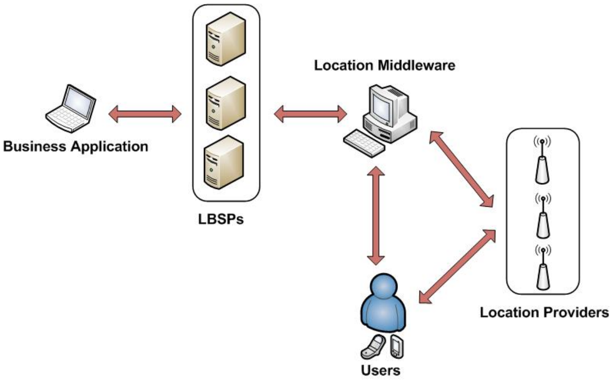


Fig. 3. Middleware Architecture

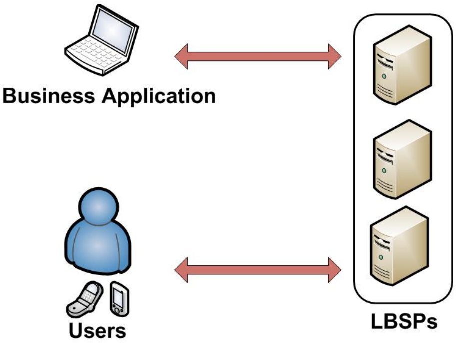


Fig. 4. Distributed Architecture

5.2 A Distributed Architecture

The distributed architecture can also be adopted by the solution presented in this paper. Figure 4 shows our distributed privacy-aware architecture in which user devices are in charge of providing user's location, applying user privacy preferences and low-level communication with LBSPs. In this approach, user device is in charge of storing region graph and preferences of the user, and calculating mapping and reliability metrics. When an LBSP queries current location of the user, the user device responses with a triple $\langle user's\ current\ location, \mu_{Map}, \mu_{Obf}\rangle$. Again, the location-based service provider can use μ_{Map} and μ_{Obf} to decide about accepting or rejecting client's request.

6 Integrating Middleware Architecture with a Location-Based Access Control Engine

As explained in section 5.1, in our middleware architecture, location middleware is the core component of the model acting as a trusted gateway between sensing technology and LBSPs. One specific LBSP which has received considerable attention in recent works is a location-based access control engine in charge of evaluating location-based predicates. The location-based predicates might be composed using the language presented in [12]. This language identifies three main classes of location-based conditions in composition of access control policies: position-based conditions on the location of the users; movement-based conditions on the mobility of the users; and interaction-based conditions relating multiple users or entities. Location-based access control policies can be considered as a means of enriching the expressive power of existing access control languages [12].

To evaluate location-based predicates such as $InRegion(u, rgn)$ which queries the presence of user u in region rgn, the access control engine requests the location middleware to determine current location of u. Location middleware queries location provider and receives LM_u and μ_{Tech} in response, determines user's current region ($region_u$) and mapping reliability (μ_{Map}) according to the location measurement, obfuscates current region of the user ($region_{Priv}$) using her privacy preferences, and measures obfuscation reliability metric (μ_{Obf}). After that, it sends $\langle region_{Priv}$, μ_{Map}, $\mu_{Obf}\rangle$ to the access control engine.

The access control engine introduces three reliability values to evaluate the response of location middleware: *i*) mapping reliability threshold, μ_{thr_Map}, to determine the acceptability of mapping reliability, *ii*) evaluation reliability, μ_{Eval}, to measure the reliability of predicate evaluation, and *iii*) evaluation reliability threshold, μ_{thr_Eval}, to determine the acceptability of evaluation reliability.

Among these values, the response of location measurement is rejected if the condition $\mu_{thr\_Map} \leq \mu_{Map}$ and $\mu_{thr\_Eval} \leq \mu_{Eval}$ does not hold; i.e. if the mapping reliability and evaluation reliability is below the thresholds acceptable by the access control engine. μ_{Eval} is computed based on obfuscation reliability.

Evaluation reliability represents the reliability of predicate evaluation. Specifically, it represents the probability that the real user's position, (x_u, y_u), belongs to rgn; i.e. the region requested in the predicate. Evaluation reliability can be measured using the following probabilities: *i*) the probability that the real user's position belongs to the intersection of LM_u and $region_{Priv}$, *ii*) the probability that a random point selected from the whole obfuscated area belongs to the intersection of LM_u and $region_{Priv}$, and *iii*) the probability that a random point selected from the whole obfuscated region belongs to the rgn.

$$\mu_{Eval} = P((x, y) \in (LM_u \cap region_{Priv})|(x, y) \in region_{Priv})$$
$$P((x_u, y_u) \in (LM_u \cap region_{Priv})|(x_u, y_u) \in LM_u)$$
$$P((x, y) \in (rgn \cap region_{Priv})|(x, y) \in region_{Priv})$$
$$\mu_{Tech}$$

The first two probabilities and μ_{Tech} were already considered in measuring μ_{Obf}. Therefore μ_{Eval} can be simplified as follows:

$$\mu_{Eval} = P((x, y) \in (rgn \cap region_{Priv})|(x, y) \in region_{Priv})\mu_{Obf}$$
$$\mu_{Eval} = \frac{s(rgn \cap region_{Priv})}{s(region_{Priv})}\mu_{Obf}$$

As an example, assume that the directed acyclic graph of regions is as depicted in Figure 2. User $Jane$, a doctor at the hospital, subscribes to location middleware by setting her privacy preferences in the following way:

..., $obf(CCU) = ICU$, ...

After that, $Jane$ requests the location-based access control engine for permission, like logging into a local network of hospital, which would be granted to her if she is currently in the hospital $(InRegion(JaneID, xHospital))$. Figure 1 shows hospital regions and location of $Jane$. Access control engine queries location middleware. Location middleware receives LM_{Jane} and $\mu_{Tech} = 0.9$ from location provider. Since CCU contains LM_{Jane} and is contained in all the other nominated regions, it is selected as $region_{Jane}$ and therefore μ_{Map} is equal to 1. After that, based on privacy preferences of $Jane$, her current region is obfuscated to ICU. Assume LM_{Jane} occupies about one-forth of ICU. Since $LM_{Jane} \subseteq ICU$, μ_{Obf} will be computed in the following way:

$$\mu_{Obf} = \frac{s(LM_{Jane})}{s(ICU)} \mu_{Tech}$$
$$\mu_{Obf} = 0.225$$

So the triple $\langle ICU, 1, 0.225 \rangle$ will be sent to access control engine as reply to its query about current location of user $Jane$. Access control engine, computes μ_{Eval} as follows:

$$\mu_{Eval} = \frac{s(xHospital \cap ICU)}{s(ICU)} \mu_{Obf}$$
$$\mu_{Eval} = \mu_{Obf} = 0.225$$

Assume $\mu_{thr\_Map} = 0.9$ and $\mu_{thr\_Eval} = 0.2$. Under such assumptions, conditions $\mu_{thr\_Map} \leq \mu_{Map}$ and $\mu_{thr\_Eval} \leq \mu_{Eval}$ hold, $InRegion(JaneID, xHospital)$ is evaluated to true, and the requested permission is granted.

Now assume $\mu_{thr\_Map} = 0.9$ and $\mu_{thr\_Eval} = 0.5$. Under such assumptions, condition $\mu_{thr\_Map} \leq \mu_{Map}$ holds, but condition $\mu_{thr\_Eval} \leq \mu_{Eval}$ fails, and although $InRegion(JaneID, xHospital)$ is evaluated to true, the result of evaluation is considered invalid. Several policies may be adopted here. The simplest approach is to query the location middleware about LM_{Jane} again, until a valid reply is received. If no valid reply is received after a certain number of steps, the requested permission will be rejected.

7 Issues, Challenges, and Benefits

In this section, we discuss some of the key issues, challenges and benefits in adoption of our solution.

7.1 Mapping Errors

One of the key challenges of our solution is the problems that may be caused by incorrect mapping of location measurement. To this aim, mapping reliability metric is introduced in the solution, showing the accuracy of mapping; i.e. for values lower than 1, mapping reliability implies possibility of mapping errors. In noncritical cases, such mapping errors can be tolerated to some extent. For instance, assume an LBSP disseminates price list of a restaurant to nearby individuals. However, if the price list is mistakenly sent to an individual not so close to the restaurant, she will probably ignore the message.

On the other hand, in critical uses, the repercussions of incorrect mapping may be catastrophic. For instance, consider an LBSP such as the location-based access control engine introduced in section 6. Incorrect mapping may result in granting user access to unauthorized resources, which would undermine the whole legitimacy of the system. To make our solution applicable to such applications, two measures must be adopted. Firstly, critical LBSPs must be warned to reject location information that are associated with mapping reliability values less than 1. Secondly, region granularity and measurement technology must be chosen carefully so that location measurements are substantially smaller than regions in terms of area; i.e. the probability of location measurements being totally contained in a region approaches 1.

7.2 Space Complexity

Although assignment of obfuscated region to every region or region type theoretically entails considerable space overhead ($O(nu)$ where n and u are the number of regions and users respectively), but it can be alleviated in practice, since an individual hardly requires obfuscation setting for every region of the world, and an efficient implementation method can substantially decrease space overhead.

In the middleware architecture all user preferences are stored on location middleware, while in the distributed architecture privacy preferences of each user is stored on her device. In the later case, complexity of space needed on each user device is $O(n)$.

7.3 Time Complexity

When a query about current location of a user is received, the following computational steps are taken: *i*) mapping user location measurement to a region using $MapLMToRegion$ algorithm, *ii*) obfuscating this region according to user privacy preferences using $ObfuscateRegion$ algorithm , and *iii*) computing reliability metrics.

Assuming that area calculation can be done in $O(c)$, c being any value, the time complexity of $MapLMToRegion$ is $\Theta(nc)$ where n indicates the number of regions.

$ObfuscateRegion$ uses a breadth-first search on the graph of regions. Remind from section 3 that the directed graph of regions has two restrictions: *i*) it is acyclic, and *ii*) it includes no transitive edge. Therefore, in breadth-first search every node (region) is enqueued at most once, so time complexity of the algorithm is $O(n)$. Reliability metrics can also be computed in $O(c)$. Therefore, the total computational time needed for responding to a given query is $\Theta(nc)$:

$O(n) + O(c) + \Theta(nc) = \Theta(nc)$

In practice, many improvements can be made, such as *i*) Use of simpler geographical representations for regions; e.g. regions can be modeled as circular areas, *ii*) Pre-calculating the areas of regions and their intersections, *iii*) Use of dynamic programming and optimization [20], and *iv*) Exploiting statistical methods in area computation, since in practice our solution only requires an approximation of these areas, not their exact values.

7.4 Contribution

The location modeling approach adopted in our work is intuitively simple both for users and LBSPs. On one hand, the graph-based model allows users to express their privacy preferences for each region in a straightforward way, which makes preference specification user-friendly and customizable. Allowing users to define different privacy preferences for different regions is totally in line with the real world privacy needs of users. On the other hand, regional approach to locations is beneficial to many LBSPs. Service provision in proximity location-based services (e.g. *finding the nearest hospital*) [17] or predicate evaluation in LBAC (e.g. $Inarea(John, Hospital)$) [12] are examples of applications for which such region-based model can be beneficial.

Another contribution of our work is introduction of reliability metrics which allow service providers to distinguish reliable location information from unreliable one, resulting in more appropriate and reliable service provision.

Also our solution can be deployed on both distributed and middleware architectures, which makes it applicable to client-server environments as well as ad-hock mobile networks and pervasive environments.

8 Conclusion and Future Work

We presented an obfuscation-based technique for preservation of location privacy based on the concept of regions. Our proposal aims at achieving a solution that measures reliability of location information, which is critical to reliable service provision, as well as providing ease of privacy management for the users. We also proposed two different privacy-aware architectures for our solution, namely a middleware architecture and a distributed architecture, and discussed their advantages and drawbacks. We also described how an exemplary LBSP such as a location-based access control engine can exploit our solution for provision of access control services. Finally, we addressed the key challenges to our solution, as well as evaluating its time and space complexity. Issues to be investigated in future include analysis of our solution assuming non-uniform distributions, robustness of the solution against privacy attacks, and use of vagueness in providing privacy.

References

1. Duckham, M., Kulik, L.: Location privacy and location-aware computing. In: Dynamic & Mobile GIS: Investigating Change in Space and Time, pp. 35–51. CRC Press, Boca Raton (2006)
2. Krumm, J.: A survey of computational location privacy. In: Workshop On Ubicomp Privacy Technologies, Innsbruck, Austria (2007)

3. Duckham, M., Kulik, L.: A formal model of obfuscation and negotiation for location privacy. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) PERVASIVE 2005. LNCS, vol. 3468, pp. 152–170. Springer, Heidelberg (2005)
4. Ardagna, C., Cremonini, M., Vimercati, S.D.C.d., Samarati, P.: Privacy-enhanced location-based access control. In: Gertz, M., Jajodia, S. (eds.) Handbook of database security: Applications and trends, pp. 531–552. Springer, Heidelberg (2007)
5. Beresford, A.R., Stajano, F.: Mix zones: User privacy in location-aware services. In: 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW 2004), pp. 127–131. IEEE Computer Society Press, Los Alamitos (2004)
6. Bettini, C., Wang, X.S., Jajodia, S.: Protecting privacy against location-based personal identification. In: Jonker, W., Petković, M. (eds.) SDM 2005. LNCS, vol. 3674, pp. 185–199. Springer, Heidelberg (2005)
7. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: 1st International Conference on Mobile Systems, Applications, and Services, pp. 31–42. ACM, New York (2003)
8. Hauser, C., Kabatnik, M.: Towards privacy support in a global location service. In: Workshop on IP and ATM Traffic Management (WATM/EUNICE 2001), Paris, France, pp. 81–89 (2001)
9. Langheinrich, M.: A privacy awareness system for ubiquitous computing environments. In: Borriello, G., Holmquist, L.E. (eds.) UbiComp 2002. LNCS, vol. 2498, pp. 237–245. Springer, Heidelberg (2002)
10. Openwave: Openwave location manager (2006), `http://www.openwave.com`
11. Bellavista, P., Corradi, A., Giannelli, C.: Efficiently managing location information with privacy requirements in wi-fi networks: a middleware approach. In: 2nd International Symposium on Wireless Communication Systems (ISWCS 2005), Siena, Italy, pp. 91–95 (2005)
12. Ardagna, C., Cremonini, M., Damiani, E., Vimercati, S.D.C.d., Samarati, P.: Supporting location-based conditions in access control policies. In: ACM Symposium on Information, Computer and Communications Security (ASIACCS 2006), Taipei, Taiwan, pp. 212–222. ACM Press, New York (2006)
13. Ardagna, C., Cremonini, M., Damiani, E., Vimercati, S.D.C.d., Samarati, P.: Managing privacy in lbac systems. In: Second IEEE International Symposium on Pervasive Computing and Ad Hoc Communications (PCAC 2007), Niagara Falls, Canada, pp. 7–12. IEEE Computer Society, Los Alamitos (2007)
14. Ardagna, C., Cremonini, M., Damiani, E., Vimercati, S.D.C.d., Samarati, P.: A middleware architecture for integrating privacy preferences and location accuracy. In: 22nd IFIP TC-11 International Information Security Conference (SEC 2007), Sandton, South Africa, vol. 232, pp. 313–324. Springer, Heidelberg (2007)
15. Ardagna, C.A., Cremonini, M., Damiani, E., De Capitani di Vimercati, S., Samarati, P.: Location privacy protection through obfuscation-based techniques. In: Barker, S., Ahn, G.-J. (eds.) Data and Applications Security 2007. LNCS, vol. 4602, pp. 47–60. Springer, Heidelberg (2007)
16. Beresford, A., Stajano, F.: Location privacy in pervasive computing. IEEE Pervasive Computing 2, 46–55 (2003)
17. Duckham, M., Mason, K., Stell, J., Worboys, M.: A formal approach to imperfection in geographic information. Computers, Environment and Urban Systems 25, 89–103 (2001)

18. Mokbel, M., Chow, C.Y., Aref, W.: The new casper: Query processing for location services without compromising privacy. In: 32nd International Conference on Very Large Data Bases, Korea, pp. 763–774 (2006)
19. Sun, G., Chen, J., Guo, W., Liu, K.R.: Signal processing techniques in network-aided positioning: A survey of state-of-the-art positioning designs. IEEE Signal Processing Magazine 22, 12–23 (2005)
20. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Dynamic programming. In: Introduction to Algorithms, 2nd edn., pp. 323–370 (2001)

Anonymous Fingerprinting for Predelivery of Contents

Kazuhiro Haramura, Maki Yoshida, and Toru Fujiwara

Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
{k-haramr,maki-yos,fujiwara}@ist.osaka-u.ac.jp

Abstract. Anonymous fingerprinting schemes aim to protect privacy and copyright in the contents delivery services. The schemes protect privacy by enabling buyers to purchase digital goods anonymously unless they redistribute purchased goods. The schemes also protect copyright by enabling the merchant of digital goods to identify the original buyer of a redistributed copy and convince an arbiter of this fact. The key idea of copyright protection is to embed some information which is linkable with the buyer into the contents. This paper presents a new anonymous fingerprinting scheme for a predelivery service where predelivered goods are encrypted by a timed-release encryption scheme so that the buyers can view the contents only after each release date. The major feature of our scheme is to allow the merchant to identify the original buyer of a redistributed ciphertext without decrypting it. Our idea is to force a buyer to attach the embedded information to a purchased ciphertext when the buyer redistributes it before the release date. We call this idea self-enforcing. This paper also proposes two constructions based on previous efficient anonymous fingerprinting schemes. These constructions realize self-enforcing in such a way that the merchant encrypts a purchased copy before encrypting by a timed-release encryption scheme so that the embedded data is needed for decryption. Note that no additional data needs to be sent by the buyer for self-enforcing because data sent by the buyer in the previous schemes is used as an encryption key. This implies efficient countermeasure against redistribution before the release date.

1 Introduction

Recently, various kinds of videos such as movies become available through online services. For some videos, a number of buyers access to the merchant on the release date. In such a case, the buyers cannot smoothly view the videos. One of the promising solutions to this problem is to enable the merchant to predeliver videos before the release date in such a way that buyers can view the videos only after the release date. Such predelivery can be realized by using a timed-release encryption scheme [1,3,4,5,6,7,8,10,12,16,17] which allows the merchant to encrypt digital goods so that they can be decrypted only after each release date.

The goal of this paper is to realize both privacy and copyright protection in a predelivery service where predelivered digital goods are encrypted by using a

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 134–151, 2009.

timed-release encryption scheme. The privacy protection means that buyers can purchase digital goods anonymously unless they redistribute purchased goods. The copyright protection means that the merchant of digital goods can identify the original buyer of a redistributed copy and convince an arbiter of this fact. These two security requirements are common to contents delivery services and considered to be satisfied. In fact, in [2,14,15], cryptographic schemes which satisfy both security requirements are proposed for the delivery services where digital goods are delivered only after each release date, that is, a buyer can view them soon after purchase. These schemes, called anonymous fingerprinting schemes, protect privacy by preventing the merchant from linking purchases of an honest buyer. In addition, the anonymous fingerprinting schemes protect copyright by enabling the merchant to sell a slightly different copy where the difference from the original contents represents some information, called embedded data, which enables the merchant not only to identify the original buyer but also to prove the correctness of the identification.

The previous anonymous fingerprinting schemes work fine in the predelivery services if identification is needed only after the release date. It is because the merchant can decrypt an encrypted copy and extract the embedded data. However, if identification is postponed after the release date, a fraudulent buyer can widely redistribute purchased copies without a risk to be identified before the release date. That is, the buyer can make much money and escape before identification. This means that the merchant's sales decreases and the merchant cannot even arrest the fraudulent buyer.

To solve this problem, the merchant should be able to obtain the embedded data even before the release date. This requirement is met by the timed-release encryption schemes with pre-open capability in [7,10], which allow only the merchant to decrypt an encrypted copy even before its release date. However, the use of pre-open capability requires decryption and extraction in order to obtain the embedded data. To counter redistribution of encrypted copies efficiently, it is more preferable that the merchant can obtain the embedded data directly without decryption and extraction.

Our contribution is to present a new anonymous fingerprinting scheme which enables the merchant to obtain the embedded data of an encrypted copy directly without decryption and extraction when the encrypted copy is redistributed. The key idea is to force a buyer to attach the embedded data to an encrypted copy when she redistributes it. We call this idea *self-enforcing*. Our method to realize self-enforcing is to encrypt a purchased copy so that any receiver of the ciphertext needs to use the embedded data of the copy for decrypting it on the release date. We present two constructions based on the previous efficient anonymous fingerprinting schemes. One is based on the scheme in [15] which uses a digital coin, and the other is based on the scheme in [2] which uses a group signature. To realize encryption for self-enforcing efficiently, our constructions use some data which is sent from the buyer to the merchant at purchase in the previous schemes as an encryption key. Thus, no additional data is needed for the buyer to send for self-enforcing. In addition, to reduce the computational complexity, we employ hybrid encryption, which is a basic technique to improve efficiency. That is, a purchased copy is encrypted by a symmetric key encryption, and then the key is encrypted for self-enforcing and timed-release. Thus, additional computation for self-enforcing is needed only for a much smaller data compared with purchased contents. In this way, the proposed anonymous fingerprinting schemes efficiently realize self-enforcing for predelivery of contents.

The rest of this paper is organized as follows. In Section 2, we define an anonymous fingerprinting scheme for predelivery of contents. In Section 3, we briefly review definitions of building blocks, which are a timed-release encryption scheme, the previous anonymous fingerprinting schemes in [2,15], and a symmetric encryption scheme. In Section 4, we define an encryption scheme for self-enforcing and propose a construction of the anonymous fingerprinting scheme for predelivery of contents using encryption for self-enforcing. In Section 5, we present two different constructions of an encryption scheme for self-enforcing based on the previous different anonymous fingerprinting schemes in [2,15]. Concluding remarks are given in Section 6.

2 Model

The model of a new anonymous fingerprinting scheme for predelivery of contents is based on those of the previous anonymous fingerprinting schemes of [2,15] and the time-server based timed-release encryption schemes of [1,3,4,5,7,8,10,12,17].

Participants. An anonymous fingerprinting scheme for predelivery of contents involves merchants, buyers, registration centers, a time server, and arbiters, denoted by $\mathcal{M}$, $\mathcal{B}$, $\mathcal{RC}$, $\mathcal{TS}$, and $\mathcal{AR}$, respectively. $\mathcal{M}$ has some digital product P_0 whose release date t_0 has been set. $\mathcal{B}$ has her identity $ID_{\mathcal{B}}$ and registers herself to $\mathcal{RC}$. Before the release date t_0, $\mathcal{B}$ buys an encrypted copy of P_0 from $\mathcal{M}$, whereas after the release date t_0, $\mathcal{B}$ buys a copy of P_0 from $\mathcal{M}$. $\mathcal{TS}$ periodically broadcasts a common absolute time reference including a time specific trapdoor which triggers decryption. If $\mathcal{M}$ finds a redistributed copy, $\mathcal{M}$ can identify its original buyer and convince $\mathcal{AR}$ the correctness of the identification regardless of whether the copy is an encrypted form or a decrypted one. Note that there is no special restriction on $\mathcal{AR}$.

Algorithms. An anonymous fingerprinting scheme for predelivery of contents, denoted AFPC, consists of the following nine probabilistic polynomial-time algorithms/protocols.

- $\mathsf{AFPC\text{-}RCKeyGen}$ is a key generation algorithm run by $\mathcal{RC}$ before the first purchase. It takes as input a security parameter 1^l and outputs a secret key $x_{\mathcal{RC}}$ and its public key $y_{\mathcal{RC}}$, which is published authentically.
- $\mathsf{AFPC\text{-}TSKeyGen}$ is a key generation algorithm run by $\mathcal{TS}$ before the first purchase. It takes as input a security parameter 1^l and outputs a secret key $x_{\mathcal{TS}}$ and its public key $y_{\mathcal{TS}}$, which is published authentically.

- AFPC-TstGen is a time specific trapdoor generation algorithm run by $\mathcal{TS}$ for each date instance. It takes as input $\mathcal{TS}$'s key pair $(x_{\mathcal{TS}}, y_{\mathcal{TS}}$ and a date t, and outputs a time specific trapdoor tst_t for t, which is broadcasted on t.
- AFPC-Reg is a registration protocol (AFPC-Reg-$\mathcal{RC}$, AFPC-Reg-$\mathcal{B}$) run by $\mathcal{RC}$ and $\mathcal{B}$ before the first or each purchase of $\mathcal{B}$. Their common input consists of $\mathcal{B}$'s identity $ID_{\mathcal{B}}$ and $\mathcal{RC}$'s public key $y_{\mathcal{RC}}$. $\mathcal{RC}$'s input is its secret key $x_{\mathcal{RC}}$. $\mathcal{B}$'s output consists of some secret $x_{\mathcal{B}}$ and related information $y_{\mathcal{B}}$. $\mathcal{RC}$ also obtains $y_{\mathcal{B}}$ and stores $(ID_{\mathcal{B}}, y_{\mathcal{B}})$.
- AFPC-Enc is a predelivery protocol (AFPC-Enc-$\mathcal{B}$, AFPC-Enc-$\mathcal{M}$) run by $\mathcal{B}$ and $\mathcal{M}$ at purchases before the release date t_0. Their common input consists of a text *text* which describes the purchased product and licensing conditions, the public keys $y_{\mathcal{RC}}$ and $y_{\mathcal{TS}}$, and the release date t_0 of P_0. $\mathcal{M}$'s secret input consists of P_0 and a transaction number j, and his output is a transaction record rec_j. $\mathcal{M}$ stores rec_j. $\mathcal{B}$'s secret input consists of her secret $x_{\mathcal{B}}$ and the related information $y_{\mathcal{B}}$ and her output consists of an encrypted copy $C_{t_0,P_{\mathcal{B}}}$ of a copy $P_{\mathcal{B}}$ into which some data *emb* is embedded, the embedded data *emb*, and some decryption information *dec*, where $C_{t_0,P_{\mathcal{B}}}$ can be decrypted if and only if all of tst_{t_0}, *emb*, and *dec* are obtained.
- AFPC-Dec is a decryption algorithm run by $\mathcal{B}$ on the release date t_0. $\mathcal{B}$'s input consists of $\mathcal{TS}$'s public key $y_{\mathcal{TS}}$, the time specific trapdoor tst_{t_0}, the embedded data *emb*, the decryption information *dec*, and the encrypted copy $C_{t_0,P_{\mathcal{B}}}$. $\mathcal{B}$'s output is the copy $P_{\mathcal{B}}$ into which *emb* is embedded.
- AFPC-Fin is a purchasing protocol (AFPC-Fin-$\mathcal{B}$, AFPC-Fin-$\mathcal{M}$) run by $\mathcal{B}$ and $\mathcal{M}$ at purchases after the release date t_0. Their common input consists of a text *text* and $\mathcal{RC}$'s public key $y_{\mathcal{RC}}$. $\mathcal{M}$'s secret input consists of P_0 and a transaction number j, and his output is a transaction record rec_j. $\mathcal{M}$ stores rec_j. $\mathcal{B}$'s secret input consists of her secret $x_{\mathcal{B}}$ and the related information $y_{\mathcal{B}}$ and her output consists of a copy $P_{\mathcal{B}}$ into which some data *emb* is embedded.
- AFPC-Ide is an identification protocol (AFPC-Ide-$\mathcal{M}$, AFPC-Ide-$\mathcal{RC}$) run by $\mathcal{M}$ and $\mathcal{RC}$ when $\mathcal{M}$ finds a redistributed copy P' which closes to P_0 or an encrypted copy C' with *emb* and *dec*. Their common input is $\mathcal{RC}$'s public key $y_{\mathcal{RC}}$. $\mathcal{RC}$'s secret input consists of its secret key $x_{\mathcal{RC}}$ and the list $\{(ID_{\mathcal{B}}, y_{\mathcal{B}})\}$. $\mathcal{M}$'s secret input consists of the digital product P_0, all transaction records $\{rec_j\}$, and either P' or *emb*. $\mathcal{M}$'s output consists of a/the fraudulent buyer's identity and a proof p that this buyer indeed bought a copy of P_0, or $\perp$ in case of failure.
- AFPC-Tri is a trial algorithm run by $\mathcal{AR}$ after a fraudulent buyer is identified by AFPC-Ide. It takes as input the identity $ID_{\mathcal{B}}$ of the fraudulent buyer, $\mathcal{RC}$'s public key $y_{\mathcal{RC}}$, and the proof p, and outputs 1 if and only if p is valid.

We require that the following conditions hold.

Correctness: All algorithms/protocols should terminate successfully whenever its players are honest (no matter how other players behaved in other protocols). Roughly speaking, $\mathcal{B}$ should be able to get $P_{\mathcal{B}}$ from tst_{t_0}, *emb*, *dec*, and $C_{t_0,P_{\mathcal{B}}}$, and $\mathcal{M}$ should be able to identify the buyer $\mathcal{B}$ from any of a redistributed copy

P' which closes to $P_{\mathcal{B}}$ and $\mathcal{B}$'s embedded data *emb* itself, and get a valid proof p even when buyers collude.

Privacy protection (Anonymity and unlinkability): Without obtaining a close copy to particular $P_{\mathcal{B}}$ or the embedded data *emb* itself, $\mathcal{M}$ (even when colluding with $\mathcal{RC}$) must not be able to identify $\mathcal{B}$ of the corresponding purchase. Furthermore, $\mathcal{M}$ (even when colluding with $\mathcal{RC}$) must not be able to tell whether two purchases were made by the same buyer. In other words, all data stored by $\mathcal{M}$ and $\mathcal{RC}$ and $\mathcal{M}$'s view of runs of AFPC-Enc and AFPC-Fin must be (computationally) independent of $\mathcal{B}$'s secret input $x_{\mathcal{B}}$ and $y_{\mathcal{B}}$.

Copyright protection (Prevention of framing buyers): No coalition of buyers, $\mathcal{M}$, and $\mathcal{RC}$ should be able to generate a proof $\tilde{p}$ such that AFPC-Tri$(ID_{\mathcal{B}}, y_{\mathcal{RC}}, \tilde{p}) = 1$, if $\mathcal{B}$ was not present in the coalition.

Control of decryption (Timed-release and self-enforcing): No one should be able to decide whether an encrypted copy emanates from digital product P_0 or P'_0 if some of the corresponding time specific trapdoor tst_{t_0}, embedded data *emb*, and decryption information *dec* are not obtained, where P_0 and P'_0 are chosen by her/him.

From the control of decryption, $\mathcal{B}$ needs to attach *emb* to $C_{t_0,P_{\mathcal{B}}}$ in order to enable anyone to decrypt $C_{t_0,P_{\mathcal{B}}}$ on t_0. Thus, self-enforcing is realized.

3 Building Blocks

3.1 Timed-Release Encryption

The timed-release encryption aims to control the timing of decryption, which was first mentioned by May [11] and discussed in detail by Rivest et. al. [16]. Recently, in [1,3,4,5,7,8,10,12,17], efficient schemes based on a time server have been proposed, where the time server periodically provides a common absolute time reference including a time specific trapdoor which triggers decryption. In these time-sever based schemes, the release date can be set precisely, and moreover the time server does not need to interact with another participant. In this paper, we use such time server based scheme.

Participants. A timed-release encryption scheme involves senders, receivers, and a time server. A sender can encrypt a message m so that the receiver can decrypt it with the time specific trapdoor of the release date t_0 broadcasted by the time server.

Algorithms. A timed-release encryption scheme, denoted TRE, is a tuple of four probabilistic polynomial-time algorithms (TRE-KeyGen, TRE-TstGen, TRE-Enc, TRE-Dec), where the input/output of TRE-KeyGen and TRE-TstGen are the same as those of AFPC-TSKeyGen and AFPC-TstGen, respectively. The input/output of TRE-Enc and TRE-Dec are given as follows.

- TRE-Enc, an encryption algorithm, takes as input the time server's public key $y_{\mathcal{TS}}$, a release date t_0, and a message m, and outputs a ciphertext $C_{t_0,m}$.

- TRE-Dec, a decryption algorithm, takes as input the time server's public key $y_{\mathcal{TS}}$, a time specific trapdoor tst_{t_0}, and a ciphertext $C_{t_0,m}$, and outputs m.

We require that the following conditions hold.

Correctness: All algorithms should terminate successfully whenever its players are honest. Roughly speaking, the receiver should be able to decrypt m from tst_{t_0} and $C_{t_0,m}$.

Control of decryption: No one should be able to decide whether a ciphertext emanates from message m_0 or m_1 if the corresponding time specific trapdoor tst_{t_0} is not obtained, where m_0 and m_1 are chosen by her/him.

3.2 Anonymous Fingerprinting

Participants. An anonymous fingerprinting scheme involves merchants, buyers, registration centers, and arbiters, denoted by $\mathcal{M}$, $\mathcal{B}$, $\mathcal{RC}$, and $\mathcal{AR}$, respectively. $\mathcal{M}$ has some digital product P_0. $\mathcal{B}$ has her identity $ID_{\mathcal{B}}$ and registers herself to $\mathcal{RC}$. Then, $\mathcal{B}$ buys a close copy of P_0 from $\mathcal{M}$. If $\mathcal{M}$ finds a redistributed copy, $\mathcal{M}$ can identify its original buyer and convince $\mathcal{AR}$ the correctness of the identification. There is no special restriction on $\mathcal{AR}$.

Algorithms. An anonymous fingerprinting scheme, denoted AF, is a tuple of five probabilistic polynomial-time algorithms/protocols (AF-RCKeyGen, AF-Reg, AF-Fin, AF-Ide, AF-Tri) where the input/output of these algorithms/protocols are almost the same as those of the corresponding algorithms/protocols of AFPC. The only difference is that $\mathcal{M}$'s secret input of AF-Ide consists of the digital product P_0, all transaction records $\{rec_j\}$, and a redistributed copy P' which closes to P_0, that is, *emb* is not directly taken as input.
We require that the following conditions hold.

Correctness: All algorithms/protocols should terminate successfully whenever its players are honest. Roughly speaking, $\mathcal{M}$ should be able to identify $\mathcal{B}$ from a redistributed copy P' which closes to $P_{\mathcal{B}}$, and get a valid proof p even when buyers collude.

Privacy protection (Anonymity and unlinkability): Without obtaining a close copy to particular $P_{\mathcal{B}}$, $\mathcal{M}$ (even when colluding with $\mathcal{RC}$) must not be able to identify $\mathcal{B}$ of the corresponding purchase. Furthermore, $\mathcal{M}$ (even when colluding with $\mathcal{RC}$) must not be able to tell whether two purchases were made by the same buyer. In other words, all data stored by $\mathcal{M}$ and $\mathcal{RC}$ and $\mathcal{M}$'s view of a run of AF-Fin must be (computationally) independent of $\mathcal{B}$'s secret input $x_{\mathcal{B}}$ and $y_{\mathcal{B}}$.

Copyright protection (Prevention of framing buyers): No coalition of buyers, $\mathcal{M}$, and $\mathcal{RC}$ should be able to generate a proof $\tilde{p}$ such that AFPC-Tri$(ID_{\mathcal{B}}, y_{\mathcal{RC}}, \tilde{p}) = 1$, if $\mathcal{B}$ was not present in the coalition.

3.3 Symmetric Encryption

Algorithms. A symmetric encryption scheme (or a data encapsulation mechanism), denoted DEM, consists of the following three deterministic polynomial-time algorithms.

- DEM-KeyDer, a deterministic key derivation algorithm, takes as input a seed s. It outputs a symmetric key K.
- DEM-Enc, an encryption algorithm, takes as input the message m and the symmetric key K and outputs a ciphertext C_m.
- DEM-Dec, a decryption algorithm, takes as input the ciphertext C_m and the symmetric key K and outputs the message m.

We require that the following conditions hold.

Correctness: All algorithms should terminate successfully whenever its players are honest. Roughly speaking, anyone should be able to decrypt C_m and get m with the seed s or the symmetric key K.

Secrecy: No one should be able to decide whether a ciphertext emanates from message m_0 or m_1 if neither the seed s nor the symmetric key K is obtained, where m_0 and m_1 are chosen by her/him.

4 Proposed Construction of Anonymous Fingerprinting

In this section, first, we overview a proposed construction of AFPC. Secondly, we define an encryption scheme for self-enforcing, denoted by ESE. Then, we present a construction of AFPC using an ESE. Finally, we briefly show its security and efficiency.

The algorithms/protocols of the proposed AFPC except for AFPC-Enc and AFPC-Dec are the same as the corresponding algorithms/protocols of TRE and AF. In contrast, we construct AFPC-Enc and AFPC-Dec by modifying AF-Fin based on its concrete constructions in [2,15]. AF-Fin in [2,15] consists of four subprocedures, request by $\mathcal{B}$, committing and proving by $\mathcal{B}$, embedding by $\mathcal{M}$, and decommitting by $\mathcal{B}$. More specifically, $\mathcal{B}$ first generates an embedded data *emb* randomly or based on her secret input, and then requests purchase by sending some data which is related to *emb* and used for linking *emb* and the purchase (i.e., the text *text*). Secondly, $\mathcal{B}$ generates secret/public parameters of commitment and commits to *emb*. The commitment of *emb* is denoted by $com(emb)$. She then proves to $\mathcal{M}$ that what is contained in the commitment $com(emb)$ is sound. Upon this, $\mathcal{M}$ runs the embedding process of [14] to obtain a committed copy $com(P_{\mathcal{B}})$ of $P_{\mathcal{B}}$ into which *emb* is embedded. Finally, $\mathcal{B}$ receives the committed copy $com(P_{\mathcal{B}})$ and obtains $P_{\mathcal{B}}$ by decommitting $com(P_{\mathcal{B}})$ using the secret parameter of commitment.

To realize self-enforcing induced by control of decryption, it is sufficient to enable $\mathcal{M}$ to encrypt the committed copy $com(P_{\mathcal{B}})$ so that $\mathcal{B}$ can decrypt it if and only if *emb* is obtained. Once such encryption is realized, $\mathcal{M}$ first encrypts the committed copy $com(P_{\mathcal{B}})$ by this encryption, and then encrypt the obtained ciphertext, denoted by $C_{com(P_{\mathcal{B}})}$, by TRE-Enc. The final ciphertext is sent to $\mathcal{B}$ as $C_{t_0,P_{\mathcal{B}}}$. From this order of encryption, $\mathcal{B}$ first needs to decrypt $C_{t_0,P_{\mathcal{B}}}$ by TRE-Dec and then decrypt $C_{com(P_{\mathcal{B}})}$ by using emb as the decryption key. Then, $\mathcal{B}$ can obtain $P_{\mathcal{B}}$ by decommitting using the secret parameter of commitment. In other words, emb can be used only after using the time specific trapdoor tst_{t_0} which is obtained on the release date. In this way, $\mathcal{B}$ needs to attach emb to $C_{t_0,P_{\mathcal{B}}}$ in order to enable anyone to decrypt $C_{t_0,P_{\mathcal{B}}}$ on the release date t_0, that is, self-enforcing is realized.

We note that AF-Fin of the previous anonymous fingerprinting schemes makes emb known to $\mathcal{B}$ only in order to satisfy copyright protection. So, the next question is how to enable $\mathcal{M}$ to encrypt $com(P_{\mathcal{B}})$ without knowing emb itself. Our answer is to use the data which is sent at request by $\mathcal{B}$ and used for linking emb and the purchase. A scheme of such encryption for self-enforcing is based on the concrete constructions of AF-Fin. We present two different constructions in Section 5. Here, we only define an encryption scheme for self-enforcing, denoted ESE, as a tuple of the following two algorithms.

- ESE-Enc, an encryption algorithm, takes as input some data pk_{emb} which links emb and the corresponding purchase and a message M, and outputs a ciphertext C_M.
- ESE-Dec, a decryption algorithm, takes as input the embedded data emb and the ciphertext C_M, and outputs the message M.

We require that the following conditions hold.

Correctness: All algorithms should terminate successfully whenever its players are honest. Roughly speaking, anyone should be able to get the message M from the embedded data emb and a ciphertext C_M.

Secrecy: No one should be able to decide whether a ciphertext emanates from message M_0 or M_1 if he does not obtain the embedded data emb, where M_0 and M_1 are chosen by her/him.

Note that we propose an ESE which provides secrecy not against adaptive chosen ciphertext attacks (CCA2), but against chosen plaintext attacks (CPA). That is, if an ESE is used alone, an adversary can modify an obtained ciphertext into a different one which is decrypted by some data generated from emb (i.e., emb itself is not needed for decryption). However, in the proposed AFPC, a ciphertext of an ESE is encrypted by a DEM. Thus, no adversary can modify the (encrypted) ciphertext before the release date and emb is needed for decryption. So, CPA-security is enough for the proposed AFPC.

Summarizing the above, the proposed AFPC is given as follows.

- AFPC-RCKeyGen(1^l): $\mathcal{RC}$ runs AF-RCKeyGen(1^l).
- AFPC-TSKeyGen(1^l): $\mathcal{TS}$ runs TRE-KeyGen(1^l).
- AFPC-TstGen($x_{\mathcal{TS}}, y_{\mathcal{TS}}, t$): $\mathcal{TS}$ runs TRE-KeyGen($x_{\mathcal{TS}}, y_{\mathcal{TS}}, t$).
- AFPC-Reg-$\mathcal{RC}$($ID_{\mathcal{B}}, y_{\mathcal{RC}}, x_{\mathcal{RC}}$), AFPC-Reg-$\mathcal{B}$($ID_{\mathcal{B}}, y_{\mathcal{RC}}$): $\mathcal{RC}$ runs AF-Reg-$\mathcal{RC}$($ID_{\mathcal{B}}, y_{\mathcal{RC}}, x_{\mathcal{RC}}$) and $\mathcal{B}$ runs AF-Reg-$\mathcal{B}$($ID_{\mathcal{B}}, y_{\mathcal{RC}}$).

- $\mathsf{AFPC\text{-}Enc\text{-}}\mathcal{B}(text, y_{\mathcal{RC}}, y_{\mathcal{TS}}, t_0, x_{\mathcal{B}}, y_{\mathcal{B}})$, $\mathsf{AFPC\text{-}Enc\text{-}}\mathcal{M}(text, y_{\mathcal{RC}}, y_{\mathcal{TS}}, t_0, P_0, j)$:

  Request of AF-Fin**.** $\mathcal{B}$ first generates an embedded data *emb* randomly or based on the secret input $x_{\mathcal{B}}$ and $y_{\mathcal{B}}$. Then, $\mathcal{B}$ generates data including pk_{emb}. $\mathcal{B}$ sends the generated data except for *emb* to $\mathcal{M}$ as request for purchase.

  Committing and proving of AF-Fin**.** $\mathcal{B}$ generates secret/public parameters of commitment and commits to the embedded data *emb*. The obtained commitment is denoted by $com(emb)$. She then proves to $\mathcal{M}$ that what is contained in the commitment $com(emb)$ is sound.

  Embedding of AF-Fin**.** $\mathcal{M}$ runs the embedding process and obtains the committed copy $com(P_{\mathcal{B}})$ of $P_{\mathcal{B}}$ into which *emb* is embedded.

  Encryption for self-enforcing. We assume that the seed space of DEM is the union of the message spaces of ESE and TRE. $\mathcal{M}$ randomly chooses a seed s_1 from the message space of ESE, runs DEM-KeyDer(s_1), and obtains a symmetric encryption key K_1. $\mathcal{M}$ runs DEM-Enc$(K_1, com(P_{\mathcal{B}}))$ to encrypt $com(P_{\mathcal{B}})$. The obtained ciphertext is denoted by $C_{com(P_{\mathcal{B}})}$. Then, $\mathcal{M}$ runs ESE-Enc(pk_{emb}, s_1) to encrypt the seed s_1. The obtained ciphertext is denoted by C_{s_1}.

  Encryption for timed-release. $\mathcal{M}$ randomly chooses a seed s_2 from the message space of TRE, runs DEM-KeyDer(s_2), and obtains a symmetric encryption key K_2. $\mathcal{M}$ runs DEM-Enc(K_2, C_{s_1}) to further encrypt the ciphertext C_{s_1}. The obtained ciphertext is denoted by $C_{C_{s_1}}$. Then, $\mathcal{M}$ runs TRE-Enc$(pk_{\mathcal{TS}}, t_0, s_2)$ to encrypt the seed s_2. The obtained ciphertext is denoted by C_{s_2}.

  The encrypted copy $C_{t_0,P_{\mathcal{B}}}$ and decryption information *dec* of $\mathcal{B}$'s output are $(C_{s_2}, C_{C_{s_1}}, C_{com(P_{\mathcal{B}})})$ and the secret parameter of commitment, respectively. The transaction record of $\mathcal{M}$'s output is that of AF-Fin.
- AFPC-Dec$(y_{\mathcal{TS}}, tst_{t_0}, emb, dec, C_{t_0,P_{\mathcal{B}}})$:

  Decryption for timed-release. For $C_{t_0,P_{\mathcal{B}}} = (C_{s_2}, C_{C_{s_1}}, C_{com(P_{\mathcal{B}})})$, $\mathcal{B}$ runs TRE-Dec$(y_{\mathcal{TS}}, tst_{t_0}, C_{s_2})$ and obtains the seed s_2. $\mathcal{B}$ runs DEM-KeyDer(s_2) and obtains the symmetric encryption key K_2. Then, $\mathcal{B}$ runs DEM-Dec$(K_2, C_{C_{s_1}})$ and obtains the ciphertext C_{s_1}.

  Decryption for self-enforcing. $\mathcal{B}$ runs ESE-Dec(emb, C_{s_1}) and obtains the seed s_1. $\mathcal{B}$ runs DEM-KeyDer(s_1) and obtains the symmetric encryption key K_1. Then, $\mathcal{B}$ runs DEM-Dec$(K_1, C_{com(P_{\mathcal{B}})})$ and obtains the committed copy $com(P_{\mathcal{B}})$.

  Decommitting of AF-Fin**.** $\mathcal{B}$ decommits $com(P_{\mathcal{B}})$ with *dec* and obtains $P_{\mathcal{B}}$.
- AFPC-Fin-$\mathcal{B}(text, y_{\mathcal{RC}}, x_{\mathcal{B}}, y_{\mathcal{B}})$, AFPC-Fin-$\mathcal{M}(text, y_{\mathcal{RC}}, P_0, j)$: $\mathcal{B}$ runs AF-Fin-$\mathcal{B}(text, y_{\mathcal{RC}}, x_{\mathcal{B}}, y_{\mathcal{B}})$ and $\mathcal{M}$ runs AF-Fin-$\mathcal{M}(text, y_{\mathcal{RC}}, P_0, j)$.
- AFPC-Ide-$\mathcal{M}(y_{\mathcal{RC}}, P_0, \{rec_j\}, P'$ or $emb)$, AFPC-Ide-$\mathcal{RC}(y_{\mathcal{RC}}, x_{\mathcal{RC}}, \{(ID_{\mathcal{B}}, y_{\mathcal{B}})\})$: If $\mathcal{M}$'s input contains P' but not *emb*, $\mathcal{M}$ runs AF-Ide-$\mathcal{M}(y_{\mathcal{RC}}, P_0, \{rec_j\}, P')$ and $\mathcal{RC}$ runs AF-Ide-$\mathcal{RC}(y_{\mathcal{RC}}, x_{\mathcal{RC}}, \{(ID_{\mathcal{B}}, y_{\mathcal{B}})\})$. Otherwise, $\mathcal{M}$ runs AF-Ide-$\mathcal{M}(y_{\mathcal{RC}}, P_0, \{rec_j\}, emb)$ except the extracting process. and $\mathcal{RC}$ runs AF-Ide-$\mathcal{RC}(y_{\mathcal{RC}}, x_{\mathcal{RC}}, \{(ID_{\mathcal{B}}, y_{\mathcal{B}})\})$
- AFPC-Tri$(ID_{\mathcal{B}}, y_{\mathcal{RC}}, p)$: $\mathcal{AR}$ runs AF-Tri$(ID_{\mathcal{B}}, y_{\mathcal{RC}}, p)$.

We briefly prove that the privacy protection and the copyright protection hold in the proposed AFPC if the corresponding conditions also hold in the used AF. And we prove that the control of decryption holds in the proposed AFPC if the control of decryption holds in the used TRE and the secrecy holds in the used DEM and the proposed ESE.

Privacy protection and copyright protection. In the proposed AFPC, the data that $\mathcal{M}$ and $\mathcal{RC}$ obtain at purchases and registrations consists of those in the used AF and the public data of TRE. Here, the data of the used AF is independent of the data of TRE. Thus, information about buyers that $\mathcal{M}$ and $\mathcal{RC}$ obtain in the proposed AFPC is the same as that in the used AF. Thus, if the privacy protection and the copyright protection hold in the used AF, the corresponding conditions also hold in the proposed AFPC.

Control of decryption. In the proposed AFPC, if the control of decryption holds in the used TRE and the secrecy holds in the proposed ESE and the used DEM, the encrypted copy $C_{t_0,P_\mathcal{B}}$ that $\mathcal{B}$ obtains first needs to be decrypted by TRE-Dec with tst_{t_0} and DEM-Dec, and then needs to be decrypted by ESE-Dec with emb and DEM-Dec. Finally, $P_\mathcal{B}$ can be obtained by decommitting using dec. Thus, the control of decryption holds in AFPC. We recall that self-enforcing is resulted by the above order of decryption.

We note that there is a possibility that the attached data is not the embedded one. Even so, the proposed AFPC can deter the redistribution, because the attached data is either an embedded data generated by one of the colluded buyers or data meaningless to decryption. For the former case, $\mathcal{M}$ can identify one of the colluded buyers and gather evidence of the redistribution. For the latter case, the redistribution is not a threat. In addition, no one would buy such encrypted copy which may not be correctly decrypted by using the attached data. To make self-enforcing more effective, we can extend the proposed AFPC so that $\mathcal{M}$ can verify that the attached data is really embedded in the encrypted copy. The idea for making self-enforcing verifiable is to use TRE with pre-open capability and modify encryption for timed-release as follows:

1. After deriving the symmetric encryption key K_2, $\mathcal{M}$ generates a hash value $hash$ of $C_{com(P_\mathcal{B})}$. $\mathcal{M}$ generates a signature $sig_\mathcal{M}$ of $hash$ and obtains a $\mathcal{B}$'s signature $sig_\mathcal{B}$ of $hash$, where $sig_\mathcal{B}$ is anonymous and its signing/verification keys are used at request of AF-Fin. Then, $\mathcal{M}$ runs ESE-Enc$(pk_{emb}, hash)$ and obtains C_{hash}.
2. $\mathcal{M}$ runs DEM-Enc$(K_2, (C_{s_1}, C_{hash}, sig_\mathcal{M}, sig_\mathcal{B}))$ instead of DEM-Enc(K_2, C_{s_1}). The obtained ciphertext is used as $C_{C_{s_1}}$. And then, C_{s_2} is generated in the same way of the proposed construction.

If $\mathcal{M}$ finds the redistributed encrypted copy $(emb, dec, C_{s_2}, C_{C_{s_1}}, C_{com(P_\mathcal{B})})$, $\mathcal{M}$ verifies emb by using C_{s_2}, $C_{C_{s_1}}$, and the hash value of $C_{com(P_\mathcal{B})}$. $\mathcal{M}$ first decrypts $C_{C_{s_1}}$ using the pre-open capability and obtains C_{s_1}, C_{hash}, $sig_\mathcal{M}$, and $sig_\mathcal{B}$. Then, $\mathcal{M}$ decrypts C_{hash} using the attached data emb. If the obtained $hash$ is the same as the hash value of $C_{com(P_\mathcal{B})}$ and both signatures are valid, then the attached data emb is really the embedded one. Because of the unforgeability of signature $sig_{\mathcal{M}}$, no coalition of buyers can generate $(emb, C_{s_2}, C_{C_{s_1}}, C_{com(P_{\mathcal{B}})})$ such that emb is not the embedded one but $\mathcal{M}$ verifies that emb is embedded. If some of emb, C_{s_2}, $C_{C_{s_1}}$, and $C_{com(P_{\mathcal{B}})}$ are altered, then $\mathcal{M}$ can detect the altering. In the case that emb is altered, C_{hash} cannot be correctly decrypted. In the case that C_{s_2}, $C_{C_{s_1}}$, and $C_{com(P_{\mathcal{B}})}$ are altered, the decrypted $hash$ and signatures are changed because of the secrecy of the TRE. In this way, we can make self-enforcing verifiable. Furthermore, $\mathcal{M}$ can prove that emb is really embedded one by using $sig_{\mathcal{B}}$ to link emb with $C_{com(P_{\mathcal{B}})}$. A concrete construction is our future work.

We also evaluate additional complexities of required communication, memory, and computation of the proposed AFPC compared with the straightforward combination of the used AF, TRE, and DEM which does not realize self-enforcing.

Communication: The proposed AFPC requires $\mathcal{B}$ to send no additional data. In contrast, $\mathcal{M}$ needs to send $(C_{s_2}, C_{C_{s_1}}, C_{com(P_{\mathcal{B}})})$ instead of $(C_{s_1}, C_{com(P_{\mathcal{B}})})$, where C_{s_1} denote a ciphertext of s_1 by TRE. The size of C_{s_2} is the same as that of C_{s_1}. Thus, the increased amount of communication is the size of $C_{C_{s_1}}$ which is that of three elements of a finite group in the both proposed constructions of ESE. The size of an element of the finite group is small compared with the size of of $P_{\mathcal{B}}$. Therefore, the additional complexity of communication is small.

Memory: The proposed AFPC requires no additional memory to $\mathcal{M}$. In contrast, $\mathcal{B}$ needs to hold emb, dec, and $(C_{s_2}, C_{C_{s_1}}, C_{com(P_{\mathcal{B}})})$ instead of dec and $(C_{s_1}, C_{com(P_{\mathcal{B}})})$, where C_{s_1} denote a ciphertext of s_1 by TRE. Thus, the increased amount of memory is the size of emb and $C_{C_{s_1}}$ which is that of five elements of a finite group. Thus, additional complexity of memory is also small.

Computation: The proposed AFPC requires additional computation to both $\mathcal{B}$ and $\mathcal{M}$. The additional computation of $\mathcal{B}$ is the computation of DEM-Dec for $C_{C_{s_1}}$, ESE-Dec for C_{s_1}, and DEM-KeyDer for s_1. On the other hand, the additional computation of $\mathcal{M}$ is the computation of DEM-KeyDer and ESE-Enc for s_1 and DEM-Enc for C_{s_1}. Thus, the increased amount of computation is that for $C_{C_{s_1}}$, C_{s_1}, and s_1 where these size is that of six elements of some finite groups, in the both proposed constructions of ESE. Thus, the additional complexity of computation is small.

5 Proposed Constructions of Encryption for Self-Enforcing

This section presents the main part of the proposed AFPC, an encryption scheme for self-enforcing ESE. Specifically, Sections 5.1 and 5.2 present different constructions based on the coin-based AF of [15] and the group signature based AF of [2], respectively. As mentioned in Section 4, it is enough for the proposed ESEs to provide secrecy against chosen plaintext attacks (CPA). It is readily proved that the proposed ESEs provide secrecy against chosen plaintext attacks (CPA) under the decisional (bilinear) Diffie-Hellman assumption because the proposed ESEs are a simple generalization of ElGamal and ElGamal is CPA-secure. A proof of the security of the coin-based ESE is given in Appendix A. Here we omit a proof of the security of the group signature based ESE, because the proof of the group signature based ESE is essentially the same as the proof of the coin-based one.

5.1 Coin Based Construction

First, we briefly recall the scheme of [15] and then present a construction of ESE.

Coin-based anonymous fingerprinting scheme of [15]. The anonymous fingerprinting scheme of [15] uses a digital cash system with double-spender identification, where the double-spender of coins can be identified as a fraudulent buyer(the "coins" only serve as cryptographic primitive and have no monetary value). Specifically, registration AF-Reg corresponds to withdrawing a coin. During purchasing AF-Fin, an unused coin and data which link the coin and data to be embedded is given to $\mathcal{M}$, and in principle a first payment with the coin is made. So far, the untraceability of the cash system should guarantee that the views of $\mathcal{RC}$ and $\mathcal{M}$ are unlinkable. Then, a second payment with the same coin is started. Now, instead of giving $\mathcal{B}$'s response to $\mathcal{M}$, it is embedded in the digital products as *emb*. This embedding is both secret and verifiable. After a redistribution, $\mathcal{M}$ can extract the second response from the digital product, and identify the fraudulent buyer as the double-spender of the corresponding coin.

Proposed encryption scheme for self-enforcing. To realize an encryption scheme for self-enforcing, we use the data which link the coin and *emb* as pk_{emb}. Let $\mathbb{G}_q$ be a group of prime order q.

In the coin-based scheme, *emb* and pk_{emb} are given by

$$emb = (is, s), \quad pk_{emb} = g_1^{is} g_2^{s},$$

where $i, s \in \mathbb{Z}_q^*$, $g_1, g_2 \in \mathbb{G}_q$. Here, *emb* and pk_{emb} can be considered as a decryption key and its encryption key of a generalized ElGamal encryption scheme, respectively. Therefore, we present the following ESE.

- ESE-Enc(pk_{emb}, M): For $pk_{emb} = g_1^{is} g_2^{s}$ and $M \in \mathbb{G}_q$, it randomly chooses an element $l \in \mathbb{Z}_q^*$, and computes $C_1 = M \cdot pk_{emb}^l$, $C_2 = g_1^l$, and $C_3 = g_2^l$. The output ciphertext is $C_M = (C_1, C_2, C_3)$.
- ESE-Dec(emb, C_M): For $emb = (is, s)$ and $C_M = (C_1, C_2, C_3)$, it outputs $M = C_1 \cdot C_2^{-is} \cdot C_3^{-s}$.

5.2 Group Signature Based Construction

First, we briefly recall the scheme of [2] and then present a construction of ESE.

Group signature based anonymous fingerprinting scheme of [2]. A group signature scheme allows an user to sign a message on the group's behalf. The scheme protects the privacy of signers by preventing the verifier from determining which member originated a signature or whether two signatures issued by the same signer. However, to handle special cases of misuse by some user, there is a designated revocation manager who can identify the signature's originator. The idea underlying the group signature based anonymous fingerprinting is to have the buyer issuing a group signature on a text *text* describing the purchased product and licensing conditions. Opposed to an ordinary group signature scheme, there is no revocation manager. Instead, $\mathcal{B}$ chooses a secret and public key pair $(sk_{\mathcal{RM}}, pk_{\mathcal{RM}})$ for the revocation manager. This public key $pk_{\mathcal{RM}}$ is then used for issuing the group signature, whereas the secret key $sk_{\mathcal{RM}}$ gets embedded into digital products as *emb*. Thus, finding a redistributed copy puts $\mathcal{M}$ in the position of the revocation manager for that particular group signature and he can identify the fraudulent buyer as the signature's originator. Due to the properties of group signature schemes, each buyer must register only once.

Note that a group signature scheme which can be used for AF of [2] should have a feature that the key setup of the revocation manager can run after the registration of group members. Recently, such group signature schemes are proposed in [9,13,18,19]. Among these schemes, we use the scheme of [13], which is the most secure and efficient.

Proposed encryption scheme for self-enforcing. To realize an encryption scheme for self-enforcing, we use the public key $pk_{\mathcal{RM}}$ for the revocation manager as the encryption key pk_{emb}. Let $\mathbb{G}$ be a cyclic additive group of prime order p, and $\mathbb{G}_M$ be a cyclic multiplicative group with the same order p. And let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_M$ be a bilinear paring.

In the group signature based scheme, *emb* and pk_{emb} are given by

$$\begin{aligned} emb &= sk_{\mathcal{RM}} = (x'_a, x'_b), \\ pk_{emb} &= pk_{\mathcal{RM}} = (\Theta_a, \Theta_b) = (e(G,G)^{x'_a}, e(G,G)^{x'_b}), \end{aligned}$$

where $x'_a, x'_b \in \mathbb{Z}_p^*$ and $G \in \mathbb{G}$. Here, *emb* and pk_{emb} can be also considered as a decryption key and its encryption key of a generalized ElGamal encryption scheme, respectively. Therefore, we present the following ESE.

- ESE-Enc(pk_{emb}, M): For $pk_{emb} = (\Theta_a, \Theta_b) = (e(G,G)^{x'_a}, e(G,G)^{x'_b})$ and $M \in \mathbb{G}_M$, it randomly selects the value $r_a, r_b \in \mathbb{Z}_p^*$, and computes $C_1 = M \cdot \Theta_a^{r_a} \cdot \Theta_b^{r_b}$, $C_2 = e(G,G)^{r_a}$, and $C_3 = e(G,G)^{r_b}$. The output ciphertext is $C_M = (C_1, C_2, C_3)$.
- ESE-Dec(emb, C_M): For $emb = (x'_a, x'_b)$ and $C_M = (C_1, C_2, C_3)$, it outputs $M = C_1 \cdot C_2^{-x'_a} \cdot C_3^{-x'_b}$.

5.3 Efficiency

The both constructions require the same amount of the additional communication, computation, and memory to realize self-enforcing in the proposed AFPC. However, the group signature based construction is more efficient than the coin-based construction in terms of communication and memory. This follows from the fact that the group signature based AF is more efficient than the coin-based

AF in the same sense. In addition, the group signature based construction has the good property that each buyer must register only once, while the coin-based construction requires registration once per purchase.

Therefore, if each merchant has enough computing power to provide the predelivery service for a number of buyers, the group signature construction works better. Otherwise, the coin-based construction works better.

6 Conclusion

In this paper, we have introduced a new anonymous fingerprinting scheme for a predelivery service where digital goods are predelivered in a form of timed-release encryption so that a buyer can view the contents only after the release date. The advantage of the proposed scheme is to realize "self-enforcing" to deter redistribution of an encrypted copy before the release date. The self-enforcing here means that the buyer is forced to attach the embedded data to an encrypted copy when she redistributes it. Our idea of realizing self-enforcing is to encrypt a purchased copy so that any receiver of the ciphertext needs to use the embedded data of the copy for decrypting it on the release date. Then, we have presented two constructions based on the coin based anonymous fingerprinting scheme in [15] and the group signature based anonymous fingerprinting schemes in [2]. A possible future work is to present another effective and efficient method to deter redistribution of an encrypted copy or to consider another type of contents delivery services.

References

1. Blake, I.F., Chan, A.C.-F.: Scalable, Server-Passive, User-Anonymous Timed Release Public Key Encryption from Bilinear Pairing. In: ICDCS 2005, pp. 504–513 (2005)
2. Camenisch, J.: Efficient Anonymous Fingerprinting with Group Signatures. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 415–428. Springer, Heidelberg (2000)
3. Cheon, J.H., Hopper, N., Kim, Y., Osipkov, I.: Timed-Release and Key-Insulated Public Key Encryption. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 191–205. Springer, Heidelberg (2006)
4. Chalkias, K., Hristu-Varsakelis, D., Stephanides, G.: Improved Anonymous Timed-Release Encryption. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 311–326. Springer, Heidelberg (2007)
5. Cathalo, J., Libert, B., Quisquater, J.-J.: Efficient and Non-Interactive Timed-Release Encryption. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 291–303. Springer, Heidelberg (2005)
6. Chalkias, K., Stephanides, G.: Timed Release Cryptography from Bilinear Pairings Using Hash Chains. In: Leitold, H., Markatos, E.P. (eds.) CMS 2006. LNCS, vol. 4237, pp. 130–140. Springer, Heidelberg (2006)
7. Dent, A.W., Tang, Q.: Revisiting the Security Model for Timed-Release Encryption with Pre-Open Capability. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 158–174. Springer, Heidelberg (2007)

8. Hristu-Varsakelis, D., Chalkias, K., Stephanide, G.: Low-cost Anonymous Timed-Release Encryption. In: IAS 2007, pp. 77–82 (2007)
9. Huang, X., Susilo, W., Mu, Y.: Breaking and Repairing Trapdoor-free Group Signature Schemes from Asiacrypt 2004. JCST 42(1), 71–74 (2005)
10. Hwang, Y.H., Yum, D.H., Lee, P.J.: Timed-Release Encryption with Pre-Open Capability and Its Application to Certified E-Mail System. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 344–358. Springer, Heidelberg (2005)
11. May, T.C.: Timed-Release Crypto (1993), `http://cypherpunks.venona.com/date/1993/02/msg00129.html`
12. Nali, D., Adams, C., Miri, A.: Time-Based Release of Confidential Information in Hierarchical Settings. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 29–43. Springer, Heidelberg (2005)
13. 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)
14. Pfitzman, B., Sadeghi, A.-R.: Coin-Based Anonymous Fingerprinting. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 150–164. Springer, Heidelberg (1999)
15. Pfitzman, B., Sadeghi, A.-R.: Anonymous Fingerprinting with Direct Non-Repudiation. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 401–414. Springer, Heidelberg (2000)
16. Rivest, R.L., Shamir, A., Wagner, D.A.: Time Lock Puzzles and Timed Release Crypto. In: MIT/LCS/TR-684 (1996)
17. Rabin, M.O., Thorpe, C.: Time-Lapse Cryptography. Technical Report TR-22-06, Harvard University School of Engineering and Computer Schience (2006)
18. Wei, V.K.: Short (resp. Fast) CCA2-Fully-Anonymous Group Signatures using IND-CPA-Encrypted Escrows. Cryptology ePrint Archive, Report 2005/410
19. Zhong, J., He, D.: A New Type of Group Signature Scheme. Cryptology ePrint Archive, Report 2006/440

A Proof of Security of the Coin-Based Construction

In this section, we prove that the proposed ESE in Section 5.1 provides the secrecy in the proposed AFPC in the sense of "Indistinguishability against Chosen-Plaintext Attacks (IND-CPA)" under the decisional Diffie-Hellman assumption.

Let $\mathcal{G}_1$ be a group instance generator that takes as input a security parameter 1^l and returns an uniformly random tuple $t = (q, \mathbb{G}_q)$, including a prime number q of size l, and the unique subgroup $\mathbb{G}_q$ of prime order q of the multiplicative group $\mathbb{Z}_p^*$ where p is a prime with $q|(p-1)$.

Decisional Diffie-Hellman assumption. For every PPT algorithm $\mathcal{A}_{\text{DDH}}$, the following function $Adv_{\text{DDH}}(l)$ is negligible.

$$Adv_{\text{DDH}}(l) = |Pr[\mathcal{A}_{\text{DDH}}(t, g', g'^x, g'^y, g'^{xy}) = 1] - Pr[\mathcal{A}_{\text{DDH}}(t, g', g'^x, g'^y, g'^z) = 1]|,$$

where $t = (q, \mathbb{G}_q) \leftarrow \mathcal{G}_1$, g' is chosen uniformly random from $\mathbb{G}_q$ and x, y, and z are chosen uniformly random from $\mathbb{Z}_q^*$.

An adversary here considered is a receiver $\mathcal{B}'$ of an encrypted copy which is redistributed with its decryption information but not with the embedded data. He tries to find two messages m_0, m_1 for which he can distinguish $C_{s_1} = \mathsf{ESE\text{-}Enc}(emb, s_1)$ of $C_{t_0,P_{\mathcal{B}}}$ with $s_1 = m_0$ from that with $s_1 = m_1$ in the environment that he can obtain the public parameters of the proposed AFPC, which are five generators of $\mathbb{G}_q$ denoted by (g, g_1, g_2, g_3, g_4), and the public keys $y_{\mathcal{RC}}$ and $y_{\mathcal{TS}}$, has any identity $ID_{\mathcal{B}'} \neq ID_{\mathcal{B}}$ (i.e., $\mathcal{B}'$ is not the original buyer of an encrypted copy), and is allowed to register himself to $\mathcal{RC}$, purchase products from $\mathcal{M}$ before/after the release date, and receive all time specific trapdoors from $\mathcal{TS}$.

We start with the definition of the CPA attack.

Setup. The challenger first generates the public parameters which are given to all algorithms and protocols as the common input. Then, the challenger runs $\mathsf{AFPC\text{-}RCKeyGen}(1^l)$ and $\mathsf{AFPC\text{-}TSKeyGen}(1^l)$ to obtain random instances of secret and public key pairs $(x_{\mathcal{RC}}, y_{\mathcal{RC}})$ and $(x_{\mathcal{TS}}, y_{\mathcal{TS}})$. It gives the public parameters and the public keys to the adversary.

Query phase 1. The adversary has adaptive access to four oracles: registration oracle, predelivery oracle, purchase oracle, and time specific trapdoor oracle corresponding to $\mathsf{AFPC\text{-}Reg}$, $\mathsf{AFPC\text{-}Enc}$, $\mathsf{AFPC\text{-}Fin}$, and $\mathsf{AFPC\text{-}TstGen}$, respectively. The challenger simulates these oracles.

Challenge. The adversary outputs two (equal length) messages m_0 and m_1. The challenger picks a random $b \in \{0,1\}$ and sets $s_1 = m_b$. Then, the challenger generates an encrypted copy $C_{t_0,P_{\mathcal{B}}} = (C_{s_2}, C_{C_{s_1}}, C_{com(P_{\mathcal{B}})})$ and decryption information *dec* where s_2 is a randomly and uniformly chosen seed, $P_{\mathcal{B}}$ is any digital product, and *dec* is the secret parameter of randomly generated parameters of commitment.

Query phase 2. The adversary continues to be access to the four oracles as in Query phase 1.

Guess. The adversary outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$.

We refer to this interaction as the ESE-CPA game, and define the advantage of an adversary $\mathcal{A}_{\mathsf{ESE}}$ as $\mathsf{AdvCPA}_{\mathcal{A}_{\mathsf{ESE}}}(l) = |\Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1]|$. We say that ESE provides the secrecy if the advantage of any PPT adversary $\mathcal{A}_{\mathsf{ESE}}$ is negligible.

Lemma 1. *The proposed* ESE *provides the secrecy in the proposed* AFPC *if the decisional Diffie-Hellman assumption holds.*

Proof. Suppose there is a PPT algorithm $\mathcal{A}_{\mathsf{ESE}}$ which breaks the secrecy of the proposed ESE. We build a PPT algorithm $\mathcal{A}_{\mathrm{DDH}}$ that breaks the decisional Diffie-Hellman assumption. Algorithm $\mathcal{A}_{\mathrm{DDH}}$ is given as input a random 4-tuple (g', g'^x, g'^y, g'^z), that is sampled either from the distribution of "true" instances where $g'^z = g'^{xy}$, or from the distribution of "false" instances where g'^z is uniform and independent in $\mathbb{G}_q$. Algorithm $\mathcal{A}_{\mathrm{DDH}}$'s goal is to output 1 if $g'^z = g'^{xy}$ and 0 otherwise. Algorithm $\mathcal{A}_{\mathrm{DDH}}$ works by interacting with $\mathcal{A}_{\mathsf{ESE}}$ in an ESE-CPA game as follows:

Setup. Algorithm $\mathcal{A}_{\rm DDH}$ randomly chooses $g, g_3, g_4 \in \mathbb{G}_q$ and $a \in \mathbb{Z}_q^*$, and sets $g_1 = g'^a$ and $g_2 = g'$. $\mathcal{A}_{\rm DDH}$ runs $\mathsf{AFPC\text{-}RCKeyGen}(1^l)$ and $\mathsf{AFPC\text{-}TSKeyGen}(1^l)$ and obtains random instances of secret and public key pairs $(x_{\mathcal{RC}}, y_{\mathcal{RC}})$ and $(x_{\mathcal{TS}}, y_{\mathcal{TS}})$. It gives the public parameters (g, g_1, g_2, g_3, g_4) and the public keys $(y_{\mathcal{RC}}, y_{\mathcal{TS}})$ to $\mathcal{A}_{\sf ESE}$.

Query phase 1. Algorithm $\mathcal{A}_{\rm DDH}$ responds to four kinds of queries (registration queries, predelivery queries, purchase queries, and time specific trapdoor queries) as follows.

- **Registration query.** $\mathcal{A}_{\rm DDH}$ runs $\mathsf{AFPC\text{-}Reg}$ in the position of $\mathcal{RC}$ using the secret and public keys $(x_{\mathcal{RC}}, y_{\mathcal{RC}})$, that is, runs $\mathsf{AFPC\text{-}Reg\text{-}}\mathcal{RC}(ID_{\mathcal{B}'}, y_{\mathcal{RC}}, x_{\mathcal{RC}})$.
- **Predelivery query.** $\mathcal{A}_{\rm DDH}$ runs $\mathsf{AFPC\text{-}Enc}$ in the position of $\mathcal{M}$ for any digital product P_0, that is, runs $\mathsf{AFPC\text{-}Enc\text{-}}\mathcal{M}(text, y_{\mathcal{RC}}, y_{\mathcal{TS}}, t_0, P_0, j)$.
- **Purchase query.** $\mathcal{A}_{\rm DDH}$ runs $\mathsf{AFPC\text{-}Fin}$ in the position of $\mathcal{M}$ for any digital product P_0, that is, runs $\mathsf{AFPC\text{-}Fin\text{-}}\mathcal{M}(text, y_{\mathcal{RC}}, P_0, j)$.
- **Time specific trapdoor query for date t.** $\mathcal{A}_{\rm DDH}$ runs $\mathsf{AFPC\text{-}TstGen}(x_{\mathcal{TS}}, y_{\mathcal{TS}}, t)$ in the position of $\mathcal{TS}$ using the secret and public keys $(x_{\mathcal{TS}}, y_{\mathcal{TS}})$ and sends tst_t to $\mathcal{A}_{\sf ESE}$.

Challenge. Algorithm $\mathcal{A}_{\sf ESE}$ will next submit two messages m_0 and m_1. Algorithm $\mathcal{A}_{\rm DDH}$ flips a fair coin, b, sets $s_1 = m_b$, and generates an encrypted copy $C_{t_0,P_{\mathcal{B}}} = (C_{s_2}, C_{C_{s_1}}, C_{com(P_{\mathcal{B}})})$ and decryption information dec as
follows.

1. **Generation of a committed copy and a decryption information.** The $\mathcal{A}_{\rm DDH}$ generates $C_{com(P_{\mathcal{B}})}$ as a commitment of any digital product $P_{\mathcal{B}}$ for randomly generated secret/public parameters of commitment, and sets the secret parameter of commitment as dec.
2. **Generation of a ciphertext of ESE.** First, $\mathcal{A}_{\rm DDH}$ runs $\mathsf{DEM\text{-}KeyDer}(s_1)$, and obtains a symmetric encryption key K_1. $\mathcal{A}_{\rm DDH}$ runs $\mathsf{DEM\text{-}Enc}(K_1,$
$com(P_{\mathcal{B}}))$ to encrypt $com(P_{\mathcal{B}})$ and obtains a ciphertext $C_{com(P_{\mathcal{B}})}$. Then, $\mathcal{A}_{\rm DDH}$ randomly chooses $a' \in \mathbb{Z}_q^*$ and sets $pk_{emb} = g_1^{a'} \cdot g'^x = g_1^{a'} \cdot g_2^x$. Thus, the corresponding (unknown) emb is (a', x). $\mathcal{A}_{\rm DDH}$ computes $C_1 = s_1 \cdot (g'^y)^{aa'} \cdot g'^z = s_1 \cdot g_1^{a'y} \cdot g_2^z$, $C_2 = (g'^y)^a = g_1^y$, and $C_3 = g'^y = g_2^y$. $\mathcal{A}_{\rm DDH}$ sets $C_{s_1} = (C_1, C_2, C_3)$.
3. **Generation of an encrypted copy.** First, $\mathcal{A}_{\rm DDH}$ randomly chooses a seed s_2 from the message space of TRE, and runs $\mathsf{DEM\text{-}KeyDer}(s_2)$ and obtains a symmetric encryption key K_2. Secondly, $\mathcal{A}_{\rm DDH}$ runs $\mathsf{DEM\text{-}Enc}(K_2, C_{s_1})$ and obtains a ciphertext $C_{C_{s_1}}$. Then, $\mathcal{A}_{\rm DDH}$ runs $\mathsf{TRE\text{-}Enc}(y_{\mathcal{TS}}, t_0, s_2)$ and obtains a ciphertext C_{s_2}. $\mathcal{A}_{\rm DDH}$ sets $C_{t_0,P_{\mathcal{B}}} = (C_{s_2}, C_{C_{s_1}}, C_{com(P_{\mathcal{B}})})$.

$\mathcal{A}_{\rm DDH}$ gives the challenge $(C_{t_0,P_{\mathcal{B}}}, dec)$ to $\mathcal{A}_{\sf ESE}$.

Query phase 2. Algorithm $\mathcal{A}_{\sf ESE}$ continues to issue any queries. Algorithm $\mathcal{A}_{\rm DDH}$ responds as in Query phase 1.

Guess. Algorithm $\mathcal{A}_{\mathsf{ESE}}$ outputs its guess $b' \in \{0,1\}$. Algorithm $\mathcal{A}_{\mathrm{DDH}}$ sets own output $b'' = 1$ if $b' = b$, and otherwise $b'' = 0$, where $b'' = 1$ means that it guesses that $g'^{z} = g'^{xy}$.

In the case that the input $(g', g'^{x}, g'^{y}, g'^{z})$ to $\mathcal{A}_{\mathrm{DDH}}$ above satisfies $g'^{xy} = g'^{z}$, the ciphertext C_{s_1} which $\mathcal{A}_{\mathsf{ESE}}$ sees is distributed exactly like an ESE encryption of m_b under the public key pk_{emb}. We use this to see that

$$\begin{aligned}\Pr[\mathcal{A}_{\mathrm{DDH}}(t, g', g'^{x}, g'^{y}, g'^{xy}) = 1] &= \frac{1}{2} \cdot \Pr[b' = 0 | b = 0] + \frac{1}{2} \cdot (1 - \Pr[b' = 0 | b = 1]) \\ &= \frac{1}{2} + \frac{1}{2}\mathsf{AdvCPA}_{\mathcal{A}_{\mathsf{ESE}}}(l). \qquad (1)\end{aligned}$$

In the case that g'^{z} is uniform and independent in $\mathbb{G}_q$, the inputs g', g'^{x}, g'^{y}, and g'^{z} to $\mathcal{A}_{\mathrm{DDH}}$ above are all uniformly distributed over $\mathbb{G}$. Algorithm $\mathcal{A}_{\mathsf{ESE}}$ sees the ciphertext $C_{s_1} = (m_b \cdot (g'^{y})^{aa'} \cdot g'^{z}, g'^{ya}, g'^{y})$. Since g'^{z} is selected uniformly at random, $m_b \cdot (g'^{y})^{aa'} \cdot g'^{z}$ is also an uniform random value. Here, with probability at least $1 - 1/q$ it is true that $z \neq xy \mod q$. Assuming this is true, the reply to query gives $\mathcal{A}_{\mathsf{ESE}}$ no information about b. So

$$\Pr[\mathcal{A}_{\mathrm{DDH}}(t, g', g'^{x}, g'^{y}, g'^{z}) = 1] \leq \frac{1}{2} \cdot \left(1 - \frac{1}{q}\right) + \frac{1}{q} = \frac{1}{2} + \frac{1}{2q}. \qquad (2)$$

The $1/q$ term accounts for the probability that $z = xy \mod q$. Subtracting Equations (1) and (2), we get

$$\begin{aligned}Adv_{\mathrm{DDH}}(l) &= \Pr[\mathcal{A}_{\mathrm{DDH}}(t, g', g'^{x}, g'^{y}, g'^{xy}) = 1] - \Pr[\mathcal{A}_{\mathrm{DDH}}(t, g', g'^{x}, g'^{y}, g'^{z}) = 1] \\ &\geq \frac{1}{2} \cdot \mathsf{AdvCPA}_{\mathcal{A}_{\mathsf{ESE}}}(l) - \frac{1}{2q}.\end{aligned}$$

As $\mathcal{A}_{\mathsf{ESE}}$ can break the secrecy of the proposed ESE, $\mathcal{A}_{\mathrm{DDH}}$ outputs the correct b'' with non-negligible probability.

Instruction Set Limitation in Support of Software Diversity

Bjorn De Sutter*, Bertrand Anckaert, Jens Geiregat, Dominique Chanet, and Koen De Bosschere

Ghent University, Electronics and Information Systems Department
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
bjorn.desutter@elis.ugent.be

Abstract. This paper proposes a novel technique, called instruction set limitation, to strengthen the resilience of software diversification against collusion attacks. Such attacks require a tool to match corresponding program fragments in different, diversified program versions. The proposed technique limits the types of instructions occurring in a program to the most frequently occurring types, by replacing the infrequently used types as much as possible by more frequently used ones. As such, this technique, when combined with diversification techniques, reduces the number of easily matched code fragments. The proposed technique is evaluated against a powerful diversification tool for Intel's x86 and an optimized matching process on a number of SPEC 2006 benchmarks.

Keywords: diversity, binary rewriting, code fragment matching, software protection.

1 Introduction

Collusion attacks on software involve the comparison of multiple versions of an application. For example, an attacker can learn how encryption keys are embedded in an application by comparing two or more versions that embed different keys. Similarly, an attacker can compare two application versions to discover how fingerprints are embedded in them. Or an attacker can compare an application before a security patch has been applied to the same application after the patch has been applied to discover the vulnerability that was addressed by the patch, and then use that information to attack unpatched versions.

Attackers also want to distribute their cracks of popular software. These cracks are nothing more than scripts that automate the attack that was devised manually by the attacker. Generating these scripts is rather easy: it suffices to perform a checksum on the application on which the script will be applied to make sure that that application is identical to the one originally cracked, and to apply the necessary transformations as simple offset-based binary code patches.

* The authors want to thank the Fund for Scientific Research - Flanders (FWO), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), and Ghent University for their support.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 152–165, 2009.

In the above scenarios attackers exploit the valid assumption that different copies of the software are lexically equivalent where they are semantically equivalent. So a natural way to defend against these attacks is to break this assumption. The goal of diversification is therefore to make sure that semantically equivalent code fragments are not lexically equivalent, and that the semantically equivalent, corresponding code fragments constituting two program versions are not easily recognized as being corresponding fragments. If diversification is successful, more effort will be needed to discover true semantical differences between program versions (in the form of keys, fingerprints or patched vulnerabilities), and it will make it much harder to develop an automated script that applies a crack to all versions of some application.

A defender can start from one single master copy of an application, and make diversified copies of the master by applying a unique set of transformations to each copy. These transformations include compiler transformations such as inlining, code factoring, tail duplication, code motion, instruction rescheduling, register reallocation, etc. as well as transformations that have been developed for obfuscating programs, such as control flow flattening, branch functions, and opaque predicates. For each copy, the transformations can be applied at different locations, or with different parameters. While none of these transformations in isolation make it very hard to match corresponding fragments in diversified programs, the combined application of all these techniques does make it harder.

In this paper, we propose a novel transformation for software diversification that will make the matching even harder. *Instruction set limitation* (ISL) replaces infrequently occurring types of instructions in programs with more frequently occurring types of instructions. By itself, this transformation does not make programs more diverse. But by eliminating infrequently occurring instructions, this technique does limit the number of easier targets for a tool that tries to match corresponding fragments in diversified program versions. As such, the proposed technique strengthens the resilience of existing diversification techniques against tools for matching program fragments.

The remainder of this paper is structured as follows. Section 2 provides background information and related work on program fragment matching, diversifying transformations, and instruction selection. The concept of ISL is discussed in Section 3, and an algorithm is proposed in Section 4. ISL is evaluated on native Intel's x86 code in Section 5, and conclusions are drawn in Section 6.

2 Background

In any of the above attacks against diversified software, an attacker first has to try to match corresponding code fragments in different software versions. For any non-trivial program and any non-trivial diversity, automated support is needed in the form of a tool that generates a list of estimated matches between code fragments. The accuracy of such a matcher can be described in terms of false positives and false negatives. These are the fractions of the estimated matches that are not real matches, and the fraction of the real matches that are not included in the estimated matches. Any diversification should try to increase the false positive matching rate and the false negative matching rate. In the remainder of this section, we briefly introduce the inner workings of matchers, of diversification tools, and of the role of instruction selection in these tools.

The attack model we will use here is that of an attacker that can observe a program or a program's execution in every detail. In this malicious host attack model, the attacker has full control over the host machine(s) running the software under attack. These might be real machines, or virtual machines, or a combination of both, that can run, for example, binary instrumentation tools such as valgrind [1] or Diota [2].

2.1 Code Matching

Attackers are most often only interested in understanding or changing the behavior of software. They are not interested in parts of the software that do not contribute to its behavior. Hence attackers are only interested in the code that actually gets executed. This implies that attackers can run a program, observe it, collect data on the executed code, and then use that data in a matching tool. In other words, the matching tool can be guided by dynamic information. A formal description of how to construct matchers using dynamic information is presented in [3]. Here we focus on the fundamental concepts.

Several kinds of information can be used by a matching tool to compare code fragments such as instructions or basic blocks. The *instruction encodings* can be considered, which consist of opcodes and type of operands. Furthermore, the values of *data produced and consumed* by instructions can be used. Or the *execution count*, i.e. the number of times that an instruction is executed for a specific input to the program. An excellent base for comparison is also provided by the first execution count: i.e. the order in which instructions are executed for the first time. And a matcher can consider the locations at which *system calls* occur, together with the arguments passed to the operating system. Using any combination of these types of information, a matcher can assign confidence levels to instruction pairs, indicating with what confidence the matcher believes the pair to be an actual match. The final estimated mapping then consists of all those pairs of which the confidence level surpasses a certain threshold. Obviously, when the threshold is increased, fewer false positives will be found, but this will likely be at the expense of increasing the false negative rate. And vice versa.

None of the above types of matching information take context into account. Instead they describe local properties of single instructions. On top of that, *control flow* and *data flow* information can be considered. Suppose that we already have an estimated mapping based on the local information. By observing a program's execution, an attacker can reconstruct a dynamic control flow graph (including only the executed code and executed control transfers), and a dynamic data dependence graph. The existing mapping can then be refined by taking into account, for each instruction or basic block in these graphs, how well their context matches. For example, consider two instructions in two program versions that the matcher considers as potential matches, albeit at a very low confidence level. The matcher can then take into account these instructions' successors and predecessors in the control flow graph and data flow graph of both program versions. If the successors and predecessors were previously matched with high confidence, the matcher can increase the confidence of the match between the two instructions themselves as well.

There are four mechanisms to combine matchers based on different types of information: combination, limitation, iteration, and bounding. First, matchers can take into account *combinations* of confidence levels, and use combined thresholds instead of thresholds on the individual confidence levels. Secondly, matchers can limit the number of estimated matching pairs per instruction to a certain upper limit. This *limitation* heuristic relies on the assumption that an instruction in one program version usually corresponds to at most a few instructions in another version. Besides resulting in more accurate results, limitation can also speed up the matching because smaller sets of possible matching candidates will be considered. This is particularly the case in *iterative matchers*. In each iteration, an iterative matcher either extends the existing estimated mapping by adding new pairs that surpass its confidence level, or it can filters the existing mapping by throwing out pairs that fall below its confidence threshold. Finally, *bounding* can be used to speed up the matching process, and to consider more contextual information of instructions. With bounding, matchers are first applied to basic blocks instead of instructions. There are much fewer basic blocks, so matching basic blocks will be a faster process. Furthermore, considering a fixed number of successor or predecessor basic blocks in the control flow graphs or data dependency graphs will take into account much more context than considering the same number of successor or predecessor instructions.

During our research on matching and diversification, we developed the matching system described in Table 1. This system was developed and optimized by means of an interactive tool that enables easy exploration of the matching system design space and that shows statistical results on false rates, as well as individual cases of false matches. A more detailed discussion of all the material discussed in this section and the components of our prototype system, including the merits and caveats of different types of matchers, are discussed in detail in [3].

2.2 Diversification

Many program transformations have been developed in compiler research. By applying them selectively in different places, different versions of an application can be generated starting from the master program. To implement this, it suffices to add an additional precondition (i.e., a validity check) for each transformation that is based on two parameters. One of them will be a user-specified probability p, which can be different for each type of transformation, and the other will be a number n generated by a pseudo-random generator. If the generated numbers are in the interval $[0, 1]$ and the added precondition is $n >= p$, then p denotes the probability with which a transformation will be applied. The diversity is maximized by maximizing $p(1 - p)$, which happens for $p = 0.5$.

Table 1. Settings of the default matching system: 21 matchers are applied iteratively, some of which combine difference types of information. The first 9 operate at the basic block level, the next 5 perform the transition from basic blocks to instructions by executing instruction-level matchers bound by the basic block result. Finally, 7 matching steps are performed at the instruction level, not bound by the basic block results. All confidence thresholds are for a confidence scale of [0, 1]. For each iteration, the maximum number of selected matches is presented. For the context-aware matchers, the direction is given in which neighboring nodes are traversed, and the distance, i.e. the length of that traversal. UP refers to predecessors, DOWN to successors.

Iteration	Granularity	Phase	#Matches	Classifier	Threshold	Direction	Δ
1	bbl	Init	1	Syscalls	0.3		
2	bbl	Extend	1	Encoding	0.5		
				Data	0.5		
				Order	0.5		
				Freq.	0.5		
3	bbl	Extend	2	Encoding	0.1		
				CF	0.1	BOTH	3
4	bbl	Filter		CF	0.1	UP	3
5	bbl	Filter		CF	0.1	DOWN	3
6	bbl	Filter		DF	0.1	UP	3
7	bbl	Filter		DF	0.1	DOWN	3
8	bbl	Extend	2	Data	0.7		
				DF	0.3	BOTH	3
9	bbl	Extend	2	Encoding	0.7		
				CF	0.3	BOTH	3
10	trans	Init	1	Syscalls	0.3		
11	trans	Extend	1	Encoding	0.5		
				Data	0.5		
				Order	0.5		
				Freq.	0.5		
12	trans	Extend	2	Encoding	0.1		
				CF	0.1	BOTH	3
13	trans	Extend	2	DATA	0.7		
				DF	0.3	BOTH	3
14	trans	Extend	2	Encoding	0.7		
				CF	0.3	BOTH	3
15	ins	Extend	3	Encoding	0.6		
				CF	0.1	BOTH	5
16	ins	Filter		CF	0.1	BOTH	5
17	ins	Filter		DF	0.1	BOTH	5
18	ins	Filter		Encoding	0.1		
19	ins	Filter		Data	0.1		
20	ins	Filter		Freq.	0.1		
21	ins	Extend	1	Encoding	0.1		
				CF	0.5	BOTH	5

In some cases, more than two alternatives (to transform or not to transform) are available. For example, code schedulers can generate many different schedules, much more than two per code fragment, and many different types of opaque predicates can be inserted. In those cases, slightly more complex decision logic needs to be implemented. Still, they all can be normalized to binary choices.

As some transformations involve the insertion of extra code in a program or code duplication, applying all transformations with probability 0.5 may slow down or bloat the program code significantly. To limit the overhead, two approaches can be taken. First of all, smaller values for p can be used. Secondly, the application of the transformations that insert overhead in the program can be limited to certain parts of the program that are determined by profiling the programs. For example, to limit the code size overhead, one can limit the transformations to those code fragments that are executed for most of the common (types of) program input. Or to limit the performance overhead, one can limit the transformations to code that is executed only infrequently.

Finally, one needs to take into account the practicality of selectively applying transformations as a diversification technique. For example, applied transformations should likely survive later transformations, rather than being undone. Furthermore, as recompiling a whole application for each sold copy is not viable, applied transformations should not require an entire recompilation. For these reasons, we believe that the feasible transformations are limited to compiler back-end transformations or to transformations that can be applied in a post-pass program rewriter, such as a link-time rewriter. Our prototype diversifier is based on the x86 backend of the Diablo link-time rewriting framework [4,5] that has previously been used for obfuscation [6,7] and steganography [8]. This prototype diversifier applies the following transformations. *Inlining* [9], *tail duplication* [9] and *two-way predicate insertion* [10] involve code duplication. *Identical function elimination*, *basic block factoring* and *function epilogue factoring* [11] all involve the replacement of duplicate code fragments by a single copy. All of those transformations not only generate diversity, they also break the assumption that there is a one-to-one mapping between two program versions' instructions. Thus, they can fool matchers that limit the number of accepted matches as discussed in Section 2.1. Furthermore, our prototype applies *control-flow flattening* [12], *branch indirection through branch functions* [13], and *opaque predicate insertion* [10]. These transformations originate from the field of program obfuscation. Fundamentally, they add unrealizable control paths of which it is hard to recognize their unrealizability. As such, they make the number of paths in the control flow graph explode, and thus make it much harder for control flow based matchers. Finally, our prototype implements a number of randomized compiler back-end tasks [9]: randomized instruction selection (see the next section), randomized instruction scheduling, and randomized code layout. The latter two transformations thwart matchers that rely on fixed static instruction orders.

2.3 Instruction Selection

Compilers map source code operations onto the operations supported in their intermediate representation, and then map those operations onto instructions available in the instruction set architecture (ISA) for which they generate assembly code. During that second step, they often have multiple choices available, because usually many sequences of instructions are semantically equivalent. Compilers are deterministic, however, and strive not only for semantic correctness, but also optimal performance and code size, so they typically reuse the same instructions and instruction patterns a lot. Because some instructions are more useful than others for more frequently occurring operations, some instructions will be used much more frequently than others.

This instruction selection and its resulting non-uniform instruction frequency are important for two reasons. Relying on a tool like a superoptimizer that generates all possible different, but semantically equivalent code sequences, we can replace the deterministic behavior of a compiler's code selection by a randomized selection to diversify programs, as mentioned in Section 2.2. In our prototype diversifier, we randomized the instruction selection by selectively replacing single instructions by alternative single instructions. The alternatives are taken from a list of equivalent instructions that was produced by a superoptimizer [14].

Secondly, it is important to understand that the non-uniform distribution of instruction frequencies can be exploited by a matcher. For any matcher based on instruction encoding, the infrequently occurring instruction types will be easy targets, as the matcher has to find matches among less candidates than it needs to do for frequently occurring instructions. As a defense against collusion attacks and matchers, we hence propose to remove as many of the infrequently occurring instructions as possible by applying ISL. This will not only make it harder for instruction encoding based matchers, but it will also do so for matchers based on control flow and data flow. The latter will happen when single instructions are replaced by sequences of multiple instructions, as this replacement inserts new instructions and new data flow. Because of the new instructions, limiting matchers as discussed in Section 2.1, will also be hampered.

We should note that, to some extent, ISL can undo the diversification obtained through randomized instruction selection. This effect can be limited, however, by applying the ISL to different instruction types. Consider for example, the `lea` (load effective address) instruction in the x86 ISA. This (large) general-purpose instruction combines a lot of computations, and can be used as an alternative to many, more specific, shorter instructions during instruction selection randomization. As the `lea` instruction occurs frequently, however, it is not a good candidate for instruction set elimination.

3 Instruction Set Limitation

Figure 1 depicts a histogram for instructions occurring in the bzip2 benchmark. Some of the infrequently occurring instructions cannot easily be replaced by other instructions, such as the x86 instructions `hlt`, `cpuid`, `in`, `int`, `iret`, `lmsw`, `out`, and `smsw`. These instructions have very specific semantics for doing IO, for communicating with the operating system, and for accessing special processor components. Other instructions, however, can easily be replaced by alternative instruction sequences that contain only more frequently occurring instructions.

In theory, almost all of the potential candidates can be replaced by more frequently occurring instructions. The URISC computer' ISA consists of only one instruction (subtract and branch conditionally) which corresponds to two instructions on the x86. However, replacing all instructions is not practically feasible, as it would result in unacceptably slow and large programs. As an example, just imagine how slow a multiplication implemented by means of subtracts and conditional jumps would be.

On the other hand, ISL should not be limited to infrequently occurring instructions. Consider the `test` instruction in Figure 1. This instruction occurs

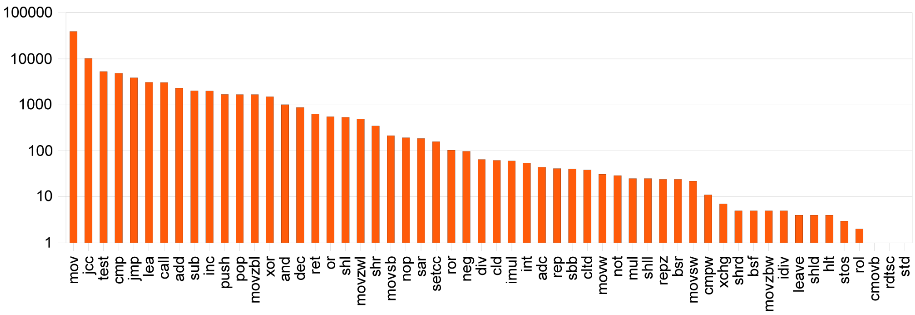


Fig. 1. Number of occurrences for each instruction in the bzip2 benchmark

Table 2. Candidate instructions for limitation. Instructions between brackets denote instructions that are not needed in all cases.

Instruction	Condition	Replacement
add	overflow and carry flags are dead	sub, (sub, push, mov, pop)
call	direct call	push, jmp
cmovcc		jcc, mov, (mov)
dec/inc	carry flag is dead	sub/add
jcc		jcc, (jmp)
leave		mov, pop
neg		sub, mov, mov
pop/push	flags are dead	mov, add/sub
ret	free register available	pop, jmp
sbb		jcc, sub, sub
setcc		jcc, mov, mov
test		and
xchg	program is single-threaded	mov, mov, mov

about five times more frequently than the `and` instruction. Still it makes sense to replace the `test` instruction by `and` instructions. Because all `test` instructions can be replaced with the `and` instruction, the final result will be that there will be six times more `and` instructions in the program, but not a single `test` instruction anymore. So even by replacing frequently occurring instructions, better distributions can be obtained to defend against matching tools.

For these reasons, we have selected the 16 instructions from Table 2 as candidates for ISL. Their replacements are shown, and the conditions in which the limitations can be applied. Some limitations can only be applied if condition flags are dead. This is the case for instructions of which the replacement sets more flags than the instruction that is replaced. Since the `xchg` instruction is used for atomic read-update-write memory accesses, it cannot be replaced by a sequence of `mov` instruction in multithreaded programs. On top of the instructions used in the replacement, additional instructions might be inserted to spill registers, i.e to free registers that are needed in the replacement code.

4 An Algorithm

Let us define the quality Q of the intruction type distribution of a program p as the sum of squares of the instruction occurrence frequencies:

$$Q(p) = \sum_{\text{instruction types } i} f(i,p)\ f(i,p). \tag{1}$$

in which $f(i,p)$ denotes the number of times an instruction type i occurs in program p.

Then consider the simple case where we want to replace x instructions of type i by x instructions of type j. This will be profitable for $Q(p)$ if $x > f(i,p) - f(j,p)$. Since x is by definition positive, this condition is always true if there are less instructions of type i than of type j. Otherwise, x has to be high enough to improve the quality of the type distribution.

Let us further define the cost of a program as the number of executed instruction is a program. This number can be obtained by profiling a program to collect basic block execution counts. While the number of executed instructions is usually not a correct measure of program execution time, it is good enough for our purpose, and it reduces the complexity of the optimization problem we are facing considerably.[1] This problem consists of optimizing the quality of the instruction type distribution given a cost budget, i.e. a maximal allowed increase in number of executed instructions. Replacing a single instruction I by a sequence of k other instructions involves a cost of $(k-1) * e(I)$, in which $e(I)$ is the execution frequency of the replaced instruction, which will also be the execution frequency of the replacements. Please note that k does not only depend in the type of I but also on its context, as in some cases it might be necessary to insert spill code to free registers or condition flags. Please also note that we will only consider instructions I of which $e(I) > 0$ as observed in the profiling runs. This follows from the fact that attackers are only interested in code that is actually executed, and hence we as defenders should also only take those instructions into account.

The algorithm we propose to solve our optimization problem works as follows. It is an iterative approach in which each iteration consists of 4 steps:

1. For each instruction type i, sort all its instructions in the program and their possible replacements in ascending order of replacement cost. Instructions I with $e(i) = 0$ are not considered at all.
2. Per instruction type i, compute the smallest set of instructions for which replacing the whole set results in a positive gain ΔQ in distribution type quality. Per type i, this set is built greedily by first considering the singleton set of the first instruction in the ordered list of step 1, and by adding the next instruction from that ordered list until the set becomes large enough to have a positive effect ΔQ on $Q(p)$ when all instructions in the set would be replaced.
3. From all such sets for all instruction types i, exclude sets of which the total replacement cost $Cost$, i.e. the summation of all the replacement costs of all instructions in that set, would result in exceeding the global cost budget (taking the cost of already performed replacements into account).

[1] Modeling execution time correctly for measuring the effects of static code transformations is practically infeasible.

4. From all remaining sets, take the one with the highest fraction of gain over cost $\frac{\Delta Q}{Cost}$, and replace all instruction in that set. If there is no remaining set, the ISL terminates, otherwise it continues with step 1.

The reason why we only consider replacement cost in step 1, and not the gain in distribution quality is that the gain depends on the order in which replacements are made, while cost does not. Computing the gains correctly for all possible replacement orders is too expensive and not worthwhile.

This algorithm may apply replacements that only have a positive ΔQ because a single instruction is replaced by multiple instructions. Since such replacements will always have a higher cost than replacements that do not increase the number of instructions, this is not problematic. Cheeper replacements will be chosen first if they are available.

5 Experimental Evaluation

To evaluate the strength of our proposed instruction set elimination as a technique to fool matching tools, we performed the following experiment. For each of five SPECint20006 benchmarks, we generated two versions A and B. On each of them, we applied instruction set limitation with cost budgets of 0%, 10%, 20% and 50%, generating binaries $A_0, A_{10}, A_{20}, A_{50}, B_0, B_{10}, B_{20}, B_{50}$. With a 50% budget, the instruction set limitation is allowed to increase the estimated number of instructions (obtained through profiling) with 50%. For each of these pairs A_i and B_i, we measured their code size growth compared to A and B, their execution time increase, and their increase in distribution quality as defined by equation 1. Furthermore, for all of the program versions pairs, we measured the false positive rates and false negative rates obtained with the matching system presented in Table 1. The results are depicted in figures 2 to 7.

As can be seen from Figure 2 a cost budget of 10% already allowed to perform almost all possible instruction replacements. Only the `sjeng` benchmark required a higher budget to perform all possible replacements that help in instruction set limitation. The resulting code size increases as depicted in Figure 3 have very similar graph shapes, albeit with slightly lower numerical values. So on average, the replacement are slightly less than twice as big as the instructions they replace.

The fact that the curves in Figure 4 are monotonically increasing when big slowdowns are seen (as for `libquantum` and `sjeng`) learns us that the cost used in Section 4 can be used to control the slowdown. And for all benchmarks but `sjeng`, the performance penalty is lower than or equal to the budget used. However, this is more due to lack of replacement opportunities than to the accuracy of our cost function. Indeed, `sjeng` shows that our cost function is not an accurate predication of the actual slowdown. New research for better cost functions is hence definitely needed. More research is also needed to understand the slight performance improvement witnessed for the 0% budget. We believe this to be a side-effect of the complex interaction between the different diversifying

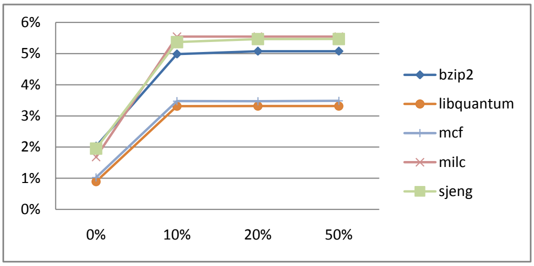


Fig. 2. The fraction of replaced instructions per cost budget

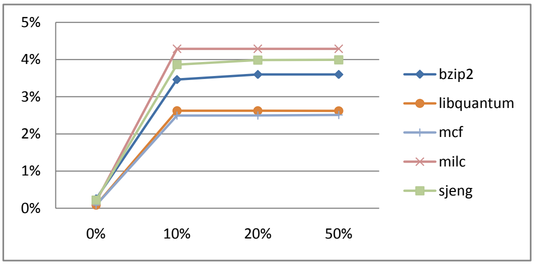


Fig. 3. The resulting code size increase

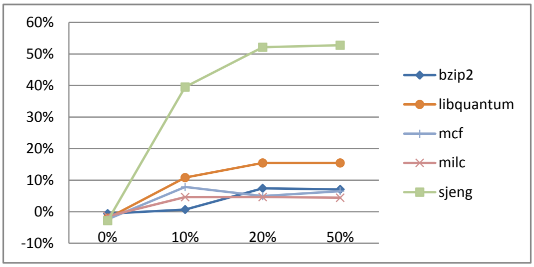


Fig. 4. Slowdown per cost budget

transformations applied by our tool and the ISL, but so far we have not been able to find the exact reason.[2]

The influence on the matching capabilities of our matching system described in Table 1 is depicted in Figures 5 and 6. First, in can be observed that the false negative rates increase significantly. This means that the matcher finds far fewer corresponding instruction pairs in the two diversified program versions. The false negative rates after ISL are roughly between 55% and 70%, while they were between about 25% and 35% before ISL, with the exception of `sjeng`, which already was at 54%. Thus, we can conclude that ISL indeed makes it harder for a matcher to find corresponding instructions in diversified program versions.

[2] Using performance counters, we were able to rule out accidental changes in branch predictor performance and cache performances.

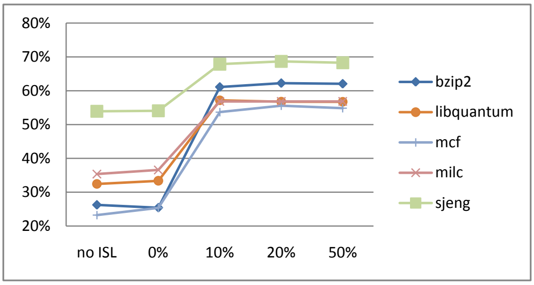


Fig. 5. False-negative rates of our matcher

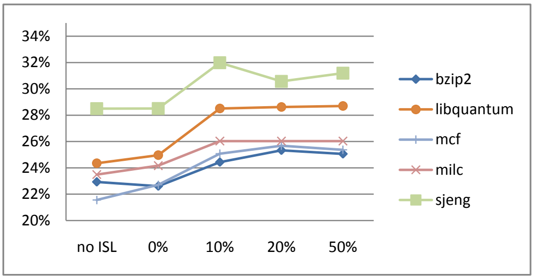


Fig. 6. False-positive rates of our matcher

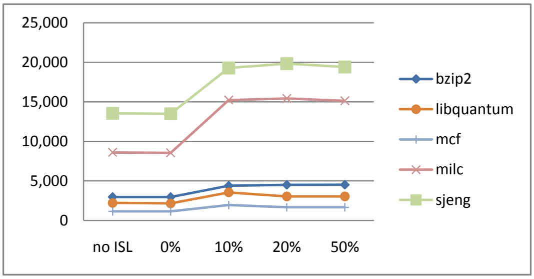


Fig. 7. Computation times (in seconds) required by the matcher

To some extent, this is the result of our matchers in iterations 2, 3, 9, 11, 12, 14, 15, 18 and 21, which rely on instruction types. However, without the these matchers, the false rates would have been worse when no ISL was applied at all. Furthermore, the small increment when using a 0% cost budget indicates that ISL only helps when all, or close to all, replaceable instructions are replaced.

The false positive ratios increase much less than the false negative ratios. The reason is that our matcher is rather conservative, and will not likely match instructions of different types. Given that we only change a small fraction of the instructions in a program, ISL will not make the matcher match much more instruction. The fact that the false positive rates are not monotonically increasing results from the complex and unpredictable interaction between the different iterations in our matching system.

Finally, it is clear from Figure 7 that ISL not only thwarts the matcher to the extent that it produces much higher false results, it also requires the matcher to perform much more computations. As a result, it requires up to 72 % more time to execute our matcher (programmed in non-optimized C#, and executed on a 2.8 GHz P4) after ISL has been applied. This is due to the fact that the sets of instructions that are compared to each other in the different iterations, are considerably larger when ISL has been applied. Thus, an attacker not only gets less useful results from his matching tool, he also needs to wait for them longer.

6 Future Work and Conclusions

This paper proposed instruction set limitation. By itself, this is not a strong software protection technique, but when it is used in combination with software diversification, our experiments have shown that instruction set limitation succeeds in making it more difficult for an automated matching system to find corresponding code fragments in diversified software versions. This thwarting happens at acceptable levels of performance overhead.

Future work includes developing specific attacks against instruction set limitation, and finding techniques to limit instruction sequences rather than individual instructions. The latter will make it much harder to develop effective attacks.

References

1. Nethercote, N., Seward, J.: Valgrind: a framework for heavyweight dynamic binary instrumentation. In: PLDI 2007: Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation, pp. 89–100 (2007)
2. Maebe, J., Ronsse, M., De Bosschere, K.: DIOTA: Dynamic Instrumentation, Optimization and Transformation of Applications. In: Compendium of Workshops and Tutorials in Conjunction with the 11th International Conference on Parallel Architectures and Compilation Techniques (2002) (count 11)
3. Anckaert, B.: Diversity for Software Protection. PhD thesis, Ghent University (2008)
4. De Bus, B.: Reliable, Retargetable and Extensivle Link-Time Program Rewriting. PhD thesis, Ghent University (2005)
5. De Sutter, B., Van Put, L., Chanet, D., De Bus, B., De Bosschere, K.: Link-time compaction and optimization of ARM executables. Trans. on Embedded Computing Sys. 6(1), 5 (2007)
6. Madou, M., Anckaert, B., De Sutter, B., De Bosschere, K.: Hybrid static-dynamic attacks against software protection mechanisms. In: Proceedings of the 5th ACM workshop on Digital Rights Management, pp. 75–82. ACM Press, New York (2005)
7. Madou, M., Van Put, L., De Bosschere, K.: Loco: An interactive code (de)obfuscation tool. In: Proceedings of ACM SIGPLAN 2006 Workshop on Partial Evaluation and Program Manipulation, PEPM 2006 (2006), `http://www.elis.ugent.be/diablo/obfuscation`

8. Anckaert, B., De Sutter, B., Chanet, D., De Bosschere, K.: Steganography for executables and code transformation signatures. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 425–439. Springer, Heidelberg (2005)
9. Muchnick, S.: Advanced Compiler Design and Implementation. Morgan Kaufmann, San Francisco (1997)
10. Collberg, C., Thomborson, C., Low, D.: Manufacturing cheap, resilient, and stealthy opaque constructs. In: Proceedings of the 25th Conference on Principles of Programming Languages, pp. 184–196. ACM, New York (1998)
11. De Sutter, B., De Bus, B., De Bosschere, K.: Sifting out the mud: low level C++ code reuse. In: OOPSLA 2002: Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, pp. 275–291 (2002)
12. Wang, Z., Pierce, K., McFarling, S.: Bmat – a binary matching tools for stale profile propagation. The Journal of Instruction-Level Parallelism 2, 1–20 (2000)
13. Linn, C., Debray, S.: Obfuscation of executable code to improve resistance to static disassembly. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, pp. 290–299. ACM Press, New York (2003)
14. Massalin, H.: Superoptimizer: a look at the smallest program. In: Proceedings of the 2nd International Conference on Architectual Support for Programming Languages and Operating Systems, pp. 122–126. IEEE Computer Society Press, Los Alamitos (1987)

Non-interactive Identity-Based DNF Signature Scheme and Its Extensions[*]

Kwangsu Lee, Jung Yeon Hwang, and Dong Hoon Lee

Graduate School of Information Management and Security,
Korea University, Seoul, Korea
{guspin,videmot,donghlee}@korea.ac.kr

Abstract. An ID-based DNF signature scheme is an ID-based signature scheme with an access structure which is expressed as a disjunctive normal form (DNF) with literals of signer identities. ID-based DNF signature schemes are useful to achieve not only signer-privacy but also a multi-user access control. In this paper, we formally define a notion of a (non-interactive) ID-based DNF signature and propose the first *non-interactive* ID-based DNF signature schemes that are secure under the computational Diffie-Hellman and subgroup decision assumptions. Our first scheme uses random oracles, and our second one is designed without random oracles. To construct the second one, we use a novel technique that converts a non-interactive witness indistinguishable proof system of encryption of one bit into a corresponding proof system of encryption of a bit-string. This technique may be of independent interest. The second scheme straightforwardly yields the first ID-based ring signature that achieves anonymity against full key exposure without random oracles. We finally present two extensions of the proposed ID-based DNF signature schemes to support multiple KGCs and different messages.

Keywords: Identity-Based Signature, Disjunctive Normal Form, Signer Anonymity, Access Structure.

1 Introduction

The notion of a digital signature is one of the most fundamental and useful inventions of modern cryptography. Since the first public key cryptosystem in [10] was introduced, various signature schemes have been suggested to meet various needs in practical circumstances. In particular, combining an access structure with a signature scheme enables users to achieve important cryptographic goals such as user anonymity and multi-user access control, etc. Traditionally, in large-scale computer systems, the security for the important resources is achieved by access controls that describe which user or component of a system is allowed to access what resources. One way to describe the access control is to use an access structure which is defined as a collection of subject sets that can access to the object. In signature systems, a signature may be viewed as a resource and a signer who

[*] This work was supported by the Second Brain Korea 21 Project.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 166–183, 2009.

generates the signature as a subject. That is, the access structure can be used to describe a collection of signer sets who participate in generating a signature. For examples, a public-key signature system implicitly includes an access structure that describes only one signer. A multi-signature system implicitly includes an access structure that describes multiple signers who participate in generating a signature. A ring signature system implicitly includes access structure such that any signer in members of a signer set generates a signature.

By applying ID-based cryptography to a signature scheme, we can construct an ID-based signature scheme in which user identity is used as a user public key [22,15,8,2]. Particularly, an ID-based signature scheme is more suitable for dealing with a complex access structure to represent an authorized set of signers, because it does not require additional information like certificates to verify a signature. In this paper, as a cryptographic primitive for more generalized access structure, we study an *ID-based DNF signature scheme*, that is an ID-based signature scheme associated with an access structure expressed as a disjunctive normal form (DNF) with "OR" and "AND" operators and an identity ID as a literal. An ID-based DNF signature is valid only if the evaluation of the corresponding DNF is true. A literal ID is evaluated to be true when a signature generated by a signer with ID is valid and false when the signature is invalid or no signature is provided. While several ID-based DNF signature schemes have been proposed [14,9], previously known schemes require interactive co-operation among signers in the access structure. That is, each signer broadcasts his random commitment and generates his own individual signature using others' random commitments. Individual signatures are then sent to a representor of signers who generates a final signature by combining the access structure. Since many parties participate in signing process, this interactive communication requires costly communication complexity with respect to system efficiency. Hence, it is highly desirable for an ID-based DNF signature scheme to be *non-interactive.*

Applications. An ID-based DNF signature scheme is a generalization of an ID-based multi-party signature scheme such as ID-based ring signature, multi-signature, designated-verifier signature, and threshold signature schemes. Thus an ID-based DNF signature scheme can be applied to various applications where an ID-based multi-party signature scheme is applied. Additionally, we can also apply it to other applications to which previous ID-based multi-party signature schemes are not suited because of inefficiency or inadequacy. For example, we may consider the situation that at least two valid signatures are necessary to guarantee the validity of a message without revealing the identities of the signers. A naive approach might be to use a ring signature scheme twice and generate two ring signatures, one for each signer. In case of using an ID-based DNF signature, two identities can be simply paired by "AND" operator in the access structure. Hence, to verify the validity of a message, we need only one signature, which in turns reduces the verification time of the signature.

Our Results. In this paper, we first give a formal definition of a non-interactive ID-based DNF signature scheme. To capture the non-interactive property, we allow individual signature queries to the adversary. Our unforgeability model captures the attacker of insider corruption and anonymity model captures the attacker of full key exposure. To construct ID-based DNF signature schemes, we extend Groth, Ostrovsky, and Sahai's non-interactive witness indistinguishable (NIWI) proof system [13] of encryption of 0 or 1 bit to encryption of two bit-strings. This extended GOS NIWI proof may be of independent interest. We use this extended one to facilitate all-or-nothing encryption of signer identities in our ID-based DNF signature without random oracles. Next we propose two non-interactive ID-based DNF signature schemes. Our first construction is efficient and the size of a signature is compact. The security of the construction is proven under the computational Diffie-Hellman (CDH) and the subgroup decision (SD) assumptions in the random oracle model. Our second construction is proven secure under the same assumptions without random oracles, while it is relatively inefficient and the size of a signature is not compact, compared to the first one. We note that the second construction directly yields the first ID-based ring signature to achieve signer anonymity against full key exposure without random oracles, because an ID-based ring signature scheme is a special case of an ID-based DNF signature scheme. Finally, we extend our ID-based DNF signature scheme with random oracles to support multiple key generation centers or different messages. Our first extension for multiple KGCs enables signers from different KGCs to generate a signature. Our second extension allows that each signer can independently generates individual signature of his own message.

Related Works. Bresson et al. [7] proposed the first signature scheme with an access structure by extending Rivest et al.'s ring signature scheme [18]. They called their scheme as ad-hoc group signature. Recently, Boyen [6] proposed Mesh signature that allows each signer to generate a signature for different messages by extending the access structure. To overcome the certificate management problem in public key signatures, ID-based signature was proposed [22,24,8,14,9,16]. The certificate management is a critical burden in signature schemes with an access structure, because the access structure contains many certificates to be verified. Thus an ID-based signature scheme with an access structure may be one of prominent solutions to resolve this problem. Herranz and Sáez [14] constructed the first ID-based signature with an access structure by extending their ID-based ring signature. Chow et al. [9] proposed another ID-based signature with an access structure by extending their ID-based ring signature that is based on Cha-Cheon ID-based signature scheme [8]. However previous two schemes require interactive communication between signers and are secure in the random oracle model. As noted above, interaction between signers greatly deteriorates the efficiency of system.

Another line of research that uses access structures is attribute-based encryption (ABE) schemes [19,12,4,17]. In attribute-based encryption schemes, the ciphertext is represented with multiple attributes and the user's private key is associated with an access structure that specifies what kinds of attributes are accepted as valid one. If attributes in the ciphertext satisfies the access structure in the user's private key, then the user can decrypt the ciphertext; otherwise, the user can't decrypt the ciphertext. The attribute-based encryption schemes are easily integrated with the role-based access control (RBAC) system [20], because roles in the RBAC are used for attributes in the ABE. The main difference between attribute-based systems and ID-based DNF systems is that in attribute-based systems, a user has a private key for multiple attributes, while in ID-based DNF systems, a user has a private key for a single attribute. So the non-interactiveness is not needed in attribute-based systems, but the collusion resistance that prevents the construction of new private key from different user's private keys is essential in attribute-based systems.

2 Backgrounds

We review the access structure, the disjunctive normal form, the bilinear groups and the complexity assumptions that our schemes are based on.

2.1 Access Structure

Let $\{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \cdots, P_n\}$, i.e., $A \subseteq 2^{\{P_1, P_2, \cdots, P_n\}} \backslash \{\emptyset\}$ [1]. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

For ID-based systems, the parties are replaced as a set of identities, Thus the access structure $\mathbb{A}$ contains the authorized set of identities.

2.2 Disjunctive Normal Form

A logical formula ψ is in disjunctive normal form (DNF) if and only if it is a disjunction ($\vee$) of one or more conjunctions ($\wedge$) of one or more literals where literal is an atomic formula (atom) or its negation. We define a DNF formula ψ as a logical formula ψ in disjunctive normal form with restriction that literal is an identity. That is, $\psi = \vee_{i=1}^{a} \wedge_{j=1}^{b_i} \mathsf{ID}_{i,j}$ where $\mathsf{ID}_{i,j}$ is an identity. We say that a set S of identities satisfies a DNF formula ψ if and only if there exist a set $S' \subseteq S$ such that $\psi(S') = 1$.

Note that an access structure $\mathbb{A}$ can be represented as a DNF formula ψ. That is, the conjunction and the disjunction of the DNF formula ψ are used to represent the subset of parties and the collection of subsets in the access structure $\mathbb{A}$ respectively.

2.3 Bilinear Groups of Composite Order

Let $n = pq$ where p and q are prime numbers. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multicative cyclic groups of same composite order n and g a generator of $\mathbb{G}$. The bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ has the following properties:

1. Bilinearity: $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_n$, we have $e(u^a, v^b) = e(u, v)^{ab}$ where the product in the exponent is a defined modulo n.
2. Non-degeneracy: $e(g, g) \neq 1$ and is thus a generator of $\mathbb{G}_T$ with order n.

We say that $\mathbb{G}$ is a bilinear group if the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map e are all efficiently computable. Note that $e(\cdot, \cdot)$ is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

2.4 Complexity Assumptions

We define two complexity assumptions: the Computational Diffie-Hellman and the Subgroup Decision assumptions.

Computational Diffie-Hellman (CDH) Assumption. Let $\mathbb{G}$ be a bilinear group of composite order $n = pq$. Let $\mathbb{G}_p$ be a subgroup of order p of $\mathbb{G}$ with a generator $g_p \in \mathbb{G}_p$. The CDH assumption in $\mathbb{G}_p$ with the composite order setting is that there is no probabilistic polynomial-time algorithm $\mathcal{A}$ that, given a tuple (g_p, g_p^a, g_p^b) additionally with the description of bilinear group $\mathbb{G}$ and its factorization (p, q) of order n, computes g_p^{ab} with non-negligible advantage. The advantage of $\mathcal{A}$ is defined as follows:

$$\mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{A},\mathbb{G},\mathbb{G}_p} = \Pr\big[\mathcal{A}((n, p, q, \mathbb{G}, \mathbb{G}_T, e), g_p, g_p^a, g_p^b) = g_p^{ab}\big]$$

where the probability is taken over the random choice of the generator $g_p \in \mathbb{G}_p$ and $a, b \in \mathbb{Z}_p$, and the random bits consumed by $\mathcal{A}$.

Subgroup Decision (SD) Assumption. Let $\mathbb{G}$ be a bilinear group of composite order $n = pq$. Let $\mathbb{G}_q$ be a subgroup of order q of $\mathbb{G}$. The Subgroup Decision (SD) assumption is that there is no probabilistic polynomial-time algorithm $\mathcal{A}$ that, given the description of $\mathbb{G}$ and h selected at random either from $\mathbb{G}$ or from $\mathbb{G}_q$, decides whether $h \in \mathbb{G}_q$ or not with non-negligible advantage. The advantage of $\mathcal{A}$ is defined as follows:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{SD}}_{\mathcal{A},\mathbb{G},\mathbb{G}_q} = \Big| & \Pr\big[h \in_R \mathbb{G} : \mathcal{A}((n, \mathbb{G}, \mathbb{G}_T, e), h) = 1\big] - \\ & \Pr\big[h \in_R \mathbb{G}_q : \mathcal{A}((n, \mathbb{G}, \mathbb{G}_T, e), h) = 1\big] \Big|\end{aligned}$$

where the probability is taken over the random choice of h and the random bits consumed by $\mathcal{A}$.

3 Definitions

Informally, an ID-based DNF signature scheme is an identity-based signature scheme expressing that the signature was generated by a signer set that satisfies a DNF formula, but it does not leak any information about the signer set. An ID-based DNF signature scheme should satisfy two security properties, namely, unforgeability and anonymity. Unforgeability is satisfied if an adversary cannot construct a valid signature on a DNF formula when he does not know private keys that satisfy the DNF formula. Anonymity is satisfied if an adversary cannot distinguish which signer set generated the signature. For security model, we adopt the strong definitions of ring signatures, namely, unforgeability against *insider corruption* and anonymity against *full key exposure* in [3].

3.1 Definition of Scheme

An ID-based DNF signature (IBDNFS) scheme consists of five algorithms (Setup, KeyGen, Sign, Merge, Verify). Formally it is defined as:

- $\mathsf{Setup}(1^\lambda)$. The setup algorithm takes as input the security parameter, outputs a public parameters PP and a master secret key MK.
- $\mathsf{KeyGen}(\mathsf{ID}, \mathsf{MK}, \mathsf{PP})$. The key generation algorithm takes as input an identity ID, the master secret key MK and the public parameters PP, outputs a private key $\mathsf{SK}_{\mathsf{ID}}$.
- $\mathsf{Sign}(M, \psi, \mathsf{SK}_{\mathsf{ID}}, \mathsf{PP})$. The individual signing algorithm takes as input a message M, a DNF formula ψ, the private key $\mathsf{SK}_{\mathsf{ID}}$ and the public parameters PP, then outputs an individual signature θ for M and ψ.
- $\mathsf{Merge}(M, \psi, \mathsf{SS}, \mathsf{PP})$. The merge algorithm takes as input a message M, a DNF formula $\psi = \vee_{i=1}^{a} \wedge_{j=1}^{b_i} \mathsf{ID}_{i,j}$, the individual signature set $\mathsf{SS} = \{(\mathsf{ID}_{i^*,j}, \theta_{i^*,j}) \mid i^* \in \{1, \ldots, a\}\}_{1 \le j \le b_{i^*}}$ and the public parameters PP, then outputs the ID-based DNF signature σ.
- $\mathsf{Verify}(\sigma, M, \psi, \mathsf{PP})$. The verification algorithm takes as input a signature σ, a message M, a DNF formula ψ and the public parameters PP, then outputs "accept" or "reject", depends on the validity of the signature.

For non-interactive ID-based DNF signature schemes, we separated the signature generation algorithm as Sign and Merge algorithms. Thus each user individually generates its own signature (without interactions), then someone merges the whole individual signatures as an ID-based DNF signature. For interactive ID-based DNF signature schemes, it is possible to combine Sign and Merge algorithms.

If a DNF formula is represented as $\psi = \vee_{i=1}^{1} \wedge_{j=1}^{b_i} \mathsf{ID}_{i,j}$, then the ID-based DNF signature of ψ equals with the ID-based multi-signature. If $\psi = \vee_{i=1}^{a} \wedge_{j=1}^{1} \mathsf{ID}_{i,j}$, then the ID-based DNF signature of ψ equals with the ID-based ring signature. In case of the ID-based threshold signature, the t-out-of-n threshold can be restated as a DNF formula ψ.

3.2 Definition of Security

Unforgeability against insider corruption is defined via the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

Setup: $\mathcal{C}$ runs the setup algorithm and keeps the master secret key MK to itself, then it gives the public parameters PP to $\mathcal{A}$.

Queries: Adaptively, $\mathcal{A}$ can request any queries described below.

- Private key extraction query: $\mathcal{A}$ requests the private key on the identity ID.

- Individual signature query: $\mathcal{A}$ requests an individual signature for a message M, a DNF formula ψ and the identity ID.
- Signature query: $\mathcal{A}$ requests a signature for a message M and a DNF formula ψ.

$\mathcal{C}$ accepts or responds to each request before accepting the next one. $\mathcal{A}$ makes q_E private key extraction queries, q_S signature queries (including individual signature queries).

Output: Finally, $\mathcal{A}$ outputs a pair (σ^*, M^*, ψ^*) and wins the game if (1) the corrupted identities set $C = \{\mathsf{ID}_i\}_{1 \leq i \leq q_E}$ by private key extraction queries does not satisfy the DNF formula ψ^*; (2) let S be the set of identities that was requested an individual signatures queries for (M^*, ψ^*), then $S \cup C$ does not satisfy the DNF formula ψ^*; (3) $\mathcal{A}$ did not request a signature for a pair (M^*, ψ^*); (4) $\mathsf{Verify}(\sigma^*, M^*, \psi^*, \mathsf{PP})$ = "accept".

Let Succ be the event that $\mathcal{A}$ wins the above game. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IBDNFS\text{-}UF}} = \Pr[\mathsf{Succ}]$ where the probability is taken over the coin tosses made by $\mathcal{A}$ and $\mathcal{C}$.

Definition 1. *An adversary $\mathcal{A}$ is said to (t, ϵ, q_E, q_S)-break an ID-based DNF signature scheme if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most q_E private key extraction queries and at most q_S signing oracle queries, and $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IBDNFS\text{-}UF}}$ is at least ϵ. An ID-based DNF signature scheme is (t, ϵ, q_E, q_S)-unforgeable if there exists no adversary that (t, ϵ, q_E, q_S)-breaks it.*

Anonymity against full key exposure is defined via the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup: $\mathcal{C}$ runs the setup algorithm and keeps the master secret key MK to itself, then it gives the public parameters PP to $\mathcal{A}$.

Queries: Adaptively, $\mathcal{A}$ can request any queries described below.

- Private key extraction query: $\mathcal{A}$ requests the private key on the identity ID.
- Individual signature query: $\mathcal{A}$ requests an individual signature for a message M, a DNF formula ψ and the identity ID.
- Signature query: $\mathcal{A}$ requests a signature for a message M and a DNF formula ψ.

$\mathcal{C}$ accepts or responds to each request before accepting the next one. $\mathcal{A}$ makes q_E private key extraction queries, q_S signature queries (including individual signature queries).

Challenges: $\mathcal{A}$ submits a challenge values (M, ψ, i_0, i_1) where $\psi = \vee_{i=1}^{a} \wedge_{j=1}^{b_i} \mathsf{ID}_{i,j}$ and $1 \leq i_0 \neq i_1 \leq a$. $\mathcal{C}$ chooses a random coin $c \in \{0, 1\}$ and computes $\sigma = \mathsf{Merge}(M, \psi, \mathsf{SS}_c, \mathsf{PP})$ where $\mathsf{SS}_c = \{(\mathsf{ID}_{i_c,j}, \theta_{i_c,j})\}_{1 \leq j \leq b_{i_c}}$ such that $\theta_{i_c,j}$ is an individual signature for (M, ψ) by the private key of $\mathsf{ID}_{i_c,j}$. Then $\mathcal{C}$ gives σ to $\mathcal{A}$.

Output: Finally, $\mathcal{A}$ outputs a guess c' of c and wins the game if $c' = c$.

Let Succ be the event that $\mathcal{A}$ wins the above game. The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IBDNFS\text{-}AN}} = \left|\Pr[\mathsf{Succ}] - \frac{1}{2}\right|$ where the probability is taken over the coin tosses made by $\mathcal{A}$ and the challenger.

Definition 2. *An adversary $\mathcal{A}$ is said to (t, ϵ, q_E, q_S)-break an ID-based DNF signature scheme if $\mathcal{A}$ runs in time at most t, $\mathcal{A}$ makes at most q_E private key extraction queries and at most q_S signing oracle queries, and $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IBDNFS\text{-}AN}}$ is at least ϵ. An ID-based DNF signature scheme is (t, ϵ, q_E, q_S)-anonymous if there exists no adversary that (t, ϵ, q_E, q_S)-breaks it.*

4 Extended GOS Proof

Boneh, Goh, and Nissim [5] proposed an encryption scheme that has homomorphic property that allows computations on ciphertexts involving arbitrary additions and one multiplication. Groth, Ostrovsky, and Sahai [13] constructed efficient non-interactive witness-indistinguishable proof system based on BGN encryption system. In this section, we construct an extended GOS proof system for encryption of two l-bit strings $(0, \ldots, 0)$ and $(1, \ldots, 1)$. Later, we use it for our construction of a DNF signature. The extended GOS proof is described as follows.

$\mathsf{Setup}(1^\lambda)$: The setup algorithm takes as input a security parameter λ, then it generates a bilinear group $\mathbb{G}$ of composite order $n = pq$ where p and q are random primes of bit size $\Theta(\lambda)$, and it selects random generators $g \in \mathbb{G}$ and $h \in \mathbb{G}_q$. Then the common reference string is set by $\mathsf{CRS} = (n, \mathbb{G}, \mathbb{G}_T, e, g, h)$.

$\mathsf{Statement}$: Let $A = (0, \ldots, 0)_l \in \mathbb{Z}_p^l$ and $B = (1, \ldots, 1)_l \in \mathbb{Z}_p^l$ where $l < p$. The statement is a ciphertext $C = (C_1, \ldots, C_l)$, and the claim is that there exists a witness $W = (M = (m_1, \ldots, m_l), Z = (z_1, \ldots, z_l)) \in \{A, B\} \times \mathbb{Z}_n^l$ such that $m_i \in \{0, 1\}$ and $C_i = g^{m_i} h^{z_i}$.

$\mathsf{Prove}(C, W, \mathsf{CRS})$: To generate a proof on the ciphertext $C = (C_1, \ldots, C_l)$ with the witness $W = (M = (m_1, \ldots, m_l), Z = (z_1, \ldots, z_l))$, it first checks $M \overset{?}{\in} \{A, B\}$ and $C_i \overset{?}{=} g^{m_i} h^{z_i}$ for all $i \in \{1, \ldots, l\}$. Next it defines f as $f = 0$ if $M = A$ and $f = 1$ if $M = B$. Then it outputs a proof of the claim as $P = \big(\pi_1 = g^{m_1} h^{z_1}, \ldots, \pi_l = g^{m_l} h^{z_l},\ \pi = (g^{l(2f-1)} \cdot h^{\Sigma_{i=1}^l z_i})^{\Sigma_{i=1}^l z_i}\big)$.

$\mathsf{Verify}(C, P, \mathsf{CRS})$: To verify the proof $P = (\pi_1, \ldots, \pi_l, \pi)$ for the ciphertext $C = (C_1, \ldots, C_l)$, it checks $e(C_i, C_i/g) \overset{?}{=} e(h, \pi_i)$ for all $i \in \{1, \ldots, l\}$, and checks $e(\prod_{j=1}^l C_j, \prod_{j=1}^l (C_j/g)) \overset{?}{=} e(h, \pi)$. If all tests are successful, then it outputs "accept"; otherwise it outputs "reject".

Remark 1. For $A = 0$ and $B = 1$ with bit-length 1, our extended GOS proof is exactly the original GOS proof.

Theorem 1. *The above extended GOS proof satisfies perfect completeness, perfect soundness, and computational witness indistinguishability under the subgroup decision assumption.*

The proof is omitted due to space constraints.

5 Construction with Random Oracles

In this section, we construct a non-interactive ID-based DNF signature scheme and prove the security of our scheme in the random oracle model. Design intuition for our construction is consistently combining Shacham-Waters ring signature scheme [21] with Gentry-Ramzan multi-signature scheme [11]. To combine these two schemes, we work in a bilinear group of composite order.

5.1 Description

Our construction is described as follows.

$\mathsf{Setup}(1^\lambda)$: The setup algorithm first generates a bilinear group $\mathbb{G}$ of composite order $n = pq$ where p and q are random primes of bit size $\Theta(\lambda)$. Next, it chooses random generators $g, w \in \mathbb{G}, h \in \mathbb{G}_q$, and a random exponent $s \in \mathbb{Z}_n$. Finally it chooses cryptographic hash functions $H_1, H_2 : \{0,1\}^* \rightarrow \mathbb{G}$. Then the public parameters PP and the master secret key MK are set by

$$\mathsf{PP} = \big(n, \mathbb{G}, \mathbb{G}_T, e, g, g_1 = g^s, h, h_1 = h^s, w, H_1, H_2\big), \qquad \mathsf{MK} = s.$$

$\mathsf{KeyGen}(\mathsf{ID}, \mathsf{MK}, \mathsf{PP})$: To generate a private key for the identity ID using the master secret key MK, the keygen algorithm computes $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$ and generates the private key $\mathsf{SK}_{\mathsf{ID}} = Q_{\mathsf{ID}}^s$.

$\mathsf{Sign}(M, \psi, \mathsf{SK}_{\mathsf{ID}}, \mathsf{PP})$: To generate an individual signature for a message M and a DNF formula ψ under the private key $\mathsf{SK}_{\mathsf{ID}} = Q_{\mathsf{ID}}^s$, the sign algorithm first computes $H_m = H_2(M, \psi)$ and chooses a random $r \in \mathbb{Z}_n$. Then the individual signature is constructed as,

$$\theta = (V, R) = \big(Q_{\mathsf{ID}}^s \cdot H_m^r,\ g^r\big) \in \mathbb{G}^2.$$

$\mathsf{Merge}(M, \psi, \mathsf{SS}, \mathsf{PP})$: The merge algorithm takes a message M, a DNF formula $\psi = \vee_{i=1}^{a} \wedge_{j=1}^{b_i} \mathsf{ID}_{i,j}$, an individual signature set $\mathsf{SS} = \{(\mathsf{ID}_{i^*,j}, \theta_{i^*,j})\}_{1 \le j \le b_{i^*}}$ where i^* is an index such that $1 \le i^* \le a$ and $\theta_{i^*,j}$ is an individual signature $(V_{i^*,j}, R_{i^*,j})$ that was generated by $\mathsf{ID}_{i^*,j}$. Let $\{f_i\}_{1 \le i \le a}$ be such that $f_i = 1$ if $i = i^*$ and $f_i = 0$ if $i \ne i^*$. To generate a signature, it proceeds as follows:

1. First, the set SS is used to create a multi-signature of the message M as $\tilde{V} = \prod_{j=1}^{b_{i^*}} V_{i^*,j}$ and $\tilde{R} = \prod_{j=1}^{b_{i^*}} R_{i^*,j}$.
2. For all $i \in \{1, \ldots, a\}$, it computes $Y_i = \prod_{j=1}^{b_i} H_1(\mathsf{ID}_{i,j})$ and chooses a random $z_i \in \mathbb{Z}_n$, then calculates $C_i = (Y_i/w)^{f_i} h^{z_i}$ and $\pi_i = ((Y_i/w)^{2f_i - 1} h^{z_i})^{z_i}$.
3. To convert $(\tilde{V}, \tilde{R})$ as a blinded one that is verifiable and anonymous, it computes $z = \sum_{i=1}^{a} z_i$ and sets $\sigma_1 = \tilde{V} \cdot h_1^z$ and $\sigma_2 = \tilde{R}$.
4. The final signature is output as,

$$\sigma = \big(\sigma_1,\ \sigma_2,\ \{(C_i, \pi_i)\}_{1 \le i \le a}\big) \in \mathbb{G}^{2a+2}.$$

$\mathsf{Verify}(\sigma, M, \psi, \mathsf{PP})$: The verify algorithm takes as input a signature σ, a message M and a DNF formula $\psi = \vee_{i=1}^{a} \wedge_{j=1}^{b_i} \mathsf{ID}_{i,j}$, then proceeds as follows.

1. For all $i \in \{1, \ldots, a\}$, it computes $Y_i = \prod_{j=1}^{b_i} H_1(\mathsf{ID}_{i,j})$ and checks if $e(C_i, C_i/(Y_i/w)) \stackrel{?}{=} e(h, \pi_i)$.
2. Next, it computes $H_m{=}H_2(M, \psi)$ and checks if $e(g, \sigma_1) \stackrel{?}{=} e(g_1, w \prod_{i=1}^{a} C_i) \cdot e(\sigma_2, H_m)$.
3. If all tests are successful, then it outputs "accept"; otherwise it outputs "reject".

5.2 Correctness

The correctness of the signature is obtained by the following equation:

$$\begin{aligned} e(g, \sigma_1) &= e(g, \prod_{j=1}^{b_{i^*}} (Q_{\mathsf{ID}_{i^*,j}}^{s} \cdot H_m^{r_j}) \cdot h_1^{z}) = e(g, (\prod_{j=1}^{b_{i^*}} Q_{\mathsf{ID}_{i^*,j}})^{s} \cdot (h^{z})^{s}) \cdot e(g, \prod_{j=1}^{b_{i^*}} H_m^{r_j}) \\ &= e(g^{s}, Y_{i^*} \cdot h^{z}) \cdot e(\prod_{j=1}^{b_{i^*}} g^{r_j}, H_m) = e(g_1, w \prod_{i=1}^{a} C_i) \cdot e(\sigma_2, H_m) \end{aligned}$$

where $Q_{\mathsf{ID}_{i^*,j}} = H_1(\mathsf{ID}_{i^*,j})$ and $H_m = H_2(M, \psi)$.

5.3 Security

Theorem 2. *The above ID-based DNF signature scheme satisfies unforgeability under the CDH assumption on $\mathbb{G}_p$ in the random oracle model.*

Theorem 3. *The above ID-based DNF signature scheme satisfies anonymity under the SD assumption in a bilinear group $\mathbb{G}$ of composite order n.*

The proofs are omitted due to space constraints.

6 Construction without Random Oracles

In this section, we construct an ID-based DNF signature without random oracles.

Design Principle. The main idea of our construction to remove random oracles is combining Shacham-Waters ring signature scheme [21] with Waters two-level signature scheme [23]. However, a simple combination of the two schemes does not lead to a provably secure scheme, because Waters two-level signature scheme reveals the number of actual signers through the size of signature. To overcome the problem, we first construct an ID-based DNF signature where the number of identities in conjunctions is same and then we remove the restriction.

For the construction where the number of identities in conjunctions is same, each signer first generates Waters two-level signature by re-randomizing the private key to break linkability of the signature. Next, a representor of signers combines these signatures to generate a DNF signature associated with a DNF formula $\psi = \vee_{i=1}^{a} \wedge_{j=1}^{b} \mathsf{ID}_{i,j}$. This DNF formula ψ can be represented as a $b \times a$

matrix where each column has identities in conjunction of ψ. We use Shacham-Waters ring signature techniques for each row by constructing BGN encryptions and GOS proofs for each entry in the matrix. To guarantee that the actual signers come from the same column in the matrix, we apply our extended GOS proof technique to each column. Additionally, we construct BGN encryptions and GOS proofs for each bit value of actual signers. Since the product of BGN encryptions of each bit value is same with the product of BGN encryption of rows in the matrix, these are redundant values. However we need these values for our security proof. The construction is described as follows.

6.1 Description

$\mathsf{Setup}(1^\lambda)$: The setup algorithm first generates a bilinear group $\mathbb{G}$ of composite order $n = pq$ where p and q are random primes of bit size $\Theta(\lambda)$. Next, it chooses random generators $g, g_2, u', u_1, \dots, u_l, v', v_1, \dots, v_m, w \in \mathbb{G}, h \in \mathbb{G}_q$, a random exponent $\alpha \in \mathbb{Z}_n$, and a collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^m$. Then the public parameters PP and the master secret key MK are set by

$$\mathsf{PP} = \big(n, \mathbb{G}, \mathbb{G}_T, e, g, g_1 = g^\alpha, g_2, h, u', u_1, \dots, u_l, v', v_1, \dots, v_m, w, H\big),$$
$$\mathsf{MK} = g_2^\alpha.$$

$\mathsf{KeyGen}(\mathsf{ID}, \mathsf{MK}, \mathsf{PP})$: To generate a private key for an identity $\mathsf{ID} = (\kappa_1, \dots, \kappa_l) \in \{0,1\}^l$ using the master secret key MK, the keygen algorithm selects a random exponent $s_1 \in \mathbb{Z}_n$, and outputs

$$\mathsf{SK}_{\mathsf{ID}} = (K_1, K_2, K_3) = \big(g_2^\alpha \cdot (u' \prod_{i=1}^{l} u_i^{\kappa_i})^{s_1},\ g^{s_1},\ h^{s_1}\big) \in \mathbb{G}^3.$$

$\mathsf{Sign}(M, \psi, \mathsf{SK}_{\mathsf{ID}}, \mathsf{PP})$: To generate an individual signature for a message M and a DNF formula ψ using the private key $\mathsf{SK}_{\mathsf{ID}}$, the sign algorithm first computes $(\mu_1, \dots, \mu_m) = H(M, \psi)$ and chooses random exponents $s_2, r \in \mathbb{Z}_n$, then it creates $V = K_1 \cdot (u' \prod_{i=1}^{l} u_i^{\kappa_i})^{s_2} \cdot (v' \prod_{j=1}^{m} v_j^{\mu_j})^r$, $S = K_2 \cdot g^{s_2}$, $T = K_3 \cdot h^{s_2}$, and $R = g^r$. If we let $s = s_1 + s_2$, then we have the individual signature as

$$\theta = \big(V, S, T, R\big) = \big(g_2^\alpha \cdot (u' \prod_{i=1}^{l} u_i^{\kappa_i})^s \cdot (v' \prod_{j=1}^{m} v_j^{\mu_j})^r,\ g^s,\ h^s,\ g^r\big) \in \mathbb{G}^4.$$

$\mathsf{Merge}(M, \psi, \mathsf{SS}, \mathsf{PP})$: The merge algorithm takes a message M, a DNF formula $\psi = \vee_{i=1}^{a} \wedge_{j=1}^{b} \mathsf{ID}_{i,j}$, an individual signature set $\mathsf{SS} = \{(\mathsf{ID}_{i^*,j}, \theta_{i^*,j})\}_{1 \le j \le b}$ where i^* is an index such that $1 \le i^* \le a$ and $\theta_{i^*,j}$ is an individual signature $(V_{i^*,j}, S_{i^*,j}, T_{i^*,j}, R_{i^*,j})$ that was generated by $\mathsf{ID}_{i^*,j}$. Let $\{f_i\}_{1 \le i \le a}$ be such that $f_i = 1$ if $i = i^*$ and $f_i = 0$ if $i \ne i^*$. To generate a signature, it proceeds as follows.

1. First, the set SS is used to create a multi-signature of the message M as $\{(\tilde{V}_j = V_{i^*,j},\ \tilde{S}_j = S_{i^*,j},\ \tilde{T}_j = T_{i^*,j},\ \tilde{R}_j = R_{i^*,j})\}_{1 \le j \le b}$.

2. For all $i \in \{1,\dots,a\}$, it defines $Y_{i,j} = u'\prod_{k=1}^{l} u_k^{\kappa_{i,j,k}}$ where $\mathsf{ID}_{i,j} = (\kappa_{i,j,1},\dots,\kappa_{i,j,l}) \in \{0,1\}^l$, and chooses random $z_{i,1},\dots,z_{i,b} \in \mathbb{Z}_n$, then it computes $\{(C_{i,j} = (Y_{i,j}/w)^{f_i} h^{z_{i,j}},\ \pi^C_{i,j} = ((Y_{i,j}/w)^{2f_i-1} h^{z_{i,j}})^{z_{i,j}})\}_{1\le j\le b}$. Next it constructs $\pi_i^{col} = ((\prod_{j=1}^{b}(Y_{i,j}/w))^{2f_i-1} h^{z_i^{col}})^{z_i^{col}}$ where $z_i^{col} = \sum_{j=1}^{b} z_{i,j}$.
3. For all $j \in \{1,\dots,b\}$, it chooses random $t_{j,1},\dots,t_{j,l-1} \in \mathbb{Z}_n$ and sets $t_{j,l} = \sum_{i=1}^{a} z_{i,j} - \sum_{k=1}^{l-1} t_{j,k}$, then it constructs $\{(D_{j,k} = u_k^{\kappa_{i^*,j,k}} \cdot h^{t_{j,k}},\ \pi^D_{j,k} = (u_k^{2\kappa_{i^*,j,k}-1} \cdot h^{t_{j,k}})^{t_{j,k}})\}_{1\le k\le l}$ for $\mathsf{ID}_{i^*,j} = (\kappa_{i^*,j,1},\dots,\kappa_{i^*,j,l})$.
4. To convert $\{(\tilde{V}_j, \tilde{S}_j, \tilde{T}_j, \tilde{R}_j)\}_{1\le j\le b}$ as a blinded one that is verifiable and anonymous, it computes $\{z_j^{row} = \sum_{i=1}^{a} z_{i,j}\}_{1\le j\le b}$ and sets $\{(\sigma_{1,j} = \tilde{V}_j \cdot \tilde{T}_j^{z_j^{row}},\ \sigma_{2,j} = \tilde{S}_j,\ \sigma_{3,j} = \tilde{R}_j)\}_{1\le j\le b}$.
5. The final signature is output as,

$$\sigma = \big(\{(\sigma_{1,j}, \sigma_{2,j}, \sigma_{3,j}\}_{1\le j\le b},\ \{(\{(C_{i,j}, \pi^C_{i,j})\}_{1\le j\le b},\ \pi_i^{col})\}_{1\le i\le a},$$
$$\{(D_{j,k}, \pi^D_{j,k})\}_{1\le j\le b, 1\le k\le l}\ \big) \in \mathbb{G}^{2ab+a+3b+2lb}.$$

$\mathsf{Verify}(\sigma, M, \psi, \mathsf{PP})$: The verify algorithm takes as input a signature σ, a message M, and a DNF formula $\psi = \vee_{i=1}^{a} \wedge_{j=1}^{b} \mathsf{ID}_{i,j}$, then it proceeds as follows.

1. For all $i \in \{1,\dots,a\}$, it computes $Y_{i,j} = u'\prod_{k=1}^{l} u_k^{\kappa_{i,j,k}}$ where $\mathsf{ID}_{i,j} = (\kappa_{i,j,1},\dots,\kappa_{i,j,l})$, then it checks that $e(C_{i,j}, C_{i,j}/(Y_{i,j}/w)) \overset{?}{=} e(h, \pi^C_{i,j})$ for all $j \in \{1,\dots,b\}$ and $e(\prod_{j=1}^{b} C_{i,j}, \prod_{j=1}^{b}(C_{i,j}/(Y_{i,j}/w))) \overset{?}{=} e(h, \pi_i^{col})$.
2. For all $j \in \{1,\dots,b\}$, it checks that $e(D_{j,k}, D_{j,k}/u_k) \overset{?}{=} e(h, \pi^D_{j,k})$ for all $k \in \{1,\dots,l\}$ and $u'\prod_{k=1}^{l} D_{j,k} \overset{?}{=} w_j \prod_{i=1}^{l} C_{i,j}$.
3. Next, it computes $(\mu_1,\dots,\mu_m) = H(M,\psi)$ and checks that

$$\forall j \in \{1,\dots,b\},\ e(g, \sigma_{1,j}) \overset{?}{=} e(g_1, g_2) \cdot e(\sigma_{2,j}, w\prod_{i=1}^{a} C_{i,j}) \cdot e(\sigma_{3,j}, v'\prod_{i=1}^{m} v_j^{\mu_i}).$$

4. If all tests are successful, then it outputs "accept"; otherwise it outputs "reject".

6.2 Correctness

The correctness of the signature is obtained by the following equation:

$$\begin{aligned} e(g, \sigma_{1,j}) &= e(g, g_2^{\alpha} \cdot (u'\prod_{k=1}^{l} u_k^{\kappa_{i^*,j,k}})^{s_j} \cdot (v'\prod_{k=1}^{m} v_k^{\mu_k})^{r_j} \cdot h^{s_j \cdot z_j^{row}}) \\ &= e(g_1, g_2) \cdot e(\sigma_{2,j}, w\prod_{i=1}^{a} C_{i,j}) \cdot e(\sigma_{3,j}, v'\prod_{k=1}^{m} v_k^{\mu_k}) \end{aligned}$$

where $w\prod_{i=1}^{a} C_{i,j} = (u'\prod_{k=1}^{l} u_k^{\kappa_{i^*,j,k}}) \cdot h^{z_j^{row}}$.

6.3 Security

Theorem 4. *The above ID-based DNF signature scheme satisfies unforgeability under the CDH assumption on $\mathbb{G}_p$ and the collision-resistant hash function H.*

The proof is given in the appendix A.

Theorem 5. *The above ID-based DNF signature scheme satisfies anonymity under the SD assumption in a bilinear group $\mathbb{G}$ of composite order n.*

The proof is omitted due to space constraints.

6.4 Removing the Restriction

The restriction that the number of identities in all conjunctions should be same can be removed by adding dummy private keys of dummy identities to the public parameters. Suppose that ψ is an original DNF formula such that the number of identities in conjunctions are not same, then we define ψ' as the number of identities in conjunctions are same by adding dummy identities to ψ. Note that we should not expand the number of disjunctions by adding dummy identities, because it is trivial to forge the signature of ψ' that contains a conjunction of dummy identities only. Since private keys for dummy identities are known to everyone, the individual signatures for dummy identities can be generated by the merge algorithm. Unforgeability and anonymity are follows from the facts that dummy private keys can be regarded as extracted private keys, dummy private keys alone can't satisfy ψ', and security models considers insider corruption and full key exposure.

Theorem 6. *The modified ID-based DNF signature scheme with dummy identities satisfies unforgeability and anonymity if the original ID-based DNF signature scheme in the section 6.1 satisfies unforgeability and anonymity.*

The proof is omitted due to space constraints.

7 Extensions

In this section, we present two extensions of our ID-based DNF signature with random oracles.

Multiple KGCs. One drawback of ID-based system is that the master secret key is only kept in the Key Generation Center (KGC). This lags the scalability of the system, thus multiple KGCs will be needed to overcome the scalability problem. Our construction with random oracles can be modified to support multiple KGCs. The idea is extending ID-based multi-signature to support multiple KGCs and using our extended GOS proof for zero or one bit-strings to guarantee that the hidden identities come from the same signers group.

Different Messages. In ID-based DNF signatures, all actual signers should generate individual signatures on a same message. However it is natural to allow each signer to generate an individual signature for it's own message. Recently, Boyen proposed similar signature scheme in public key system [6]. The idea to construct an ID-based DNF signature for different messages is using Gentry-Ramzan's ID-based aggregate signature scheme [11] as a building block. In the ID-based aggregate signature scheme, given n signatures on n distinct messages from n distinct users, all these signatures can be aggregated into a single short signature.

8 Conclusion

We presented the first non-interactive ID-based DNF signatures that are secure under the CDH and subgroup decision assumptions. Our first construction uses random oracles, but it is efficient and the size of signature is compact. Our second construction does not use random oracles, but the size of signature is not compact. We note that the second construction directly yields the first ID-based ring signature to achieve signer anonymity against full key exposure without random oracles, because an ID-based ring signature scheme is a special case of an ID-based DNF signature scheme. Additionally we presented extensions of our scheme that support multiple KGCs and different messages. One interesting open problem is to construct a compact ID-based DNF signature without random oracles.

References

1. Beimel, A.: Secure schemes for secret sharing and key distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
2. Bellare, M., Namprempre, C., Neven, G.: Security proofs for identity-based identification and signature schemes. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 268–286. Springer, Heidelberg (2004)
3. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg (2007)
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
5. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–342. Springer, Heidelberg (2005)
6. Boyen, X.: Mesh signatures How to leak a secret with unwitting and unwilling participants. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 210–227. Springer, Heidelberg (2007)
7. Bresson, E., Stern, J., Szydlo, M.: Threshold ring signatures and applications to ad-hoc groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 465–480. Springer, Heidelberg (2002)
8. 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 (2003)

9. Chow, S.S.M., Yiu, S.-M., Hui, L.C.K.: Efficient identity based ring signature. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 499–512. Springer, Heidelberg (2005)
10. Diffe, W., Hellman, M.E.: New directions in cryptgraphy. IEEE Transactions on Information Theory IT-22(6), 644–654 (1976)
11. Gentry, C., Ramzan, Z.: Identity-based aggregate signatures. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 257–273. Springer, Heidelberg (2006)
12. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute based encryption for fine-graned access control of encrypted data. In: ACM conference on Computer and Communications Security (ACM CCS), pp. 89–98 (2006)
13. 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)
14. Herranz, J., Sáez, G.: New identity-based ring signature schemes. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 27–39. Springer, Heidelberg (2004)
15. 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)
16. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
17. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: ACM conference on Computer and Communications Security (ACM CCS), pp. 195–203 (2007)
18. Rivest, R., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
19. Sahai, A., Waters, B.: Fuzzy identity based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
20. Sandhu, R.S., Coyne, E.J., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
21. Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 166–180. Springer, Heidelberg (2007)
22. Shamir, A.: Identity-based cryptosystems and signaure shcemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
23. Waters, B.: Efficient identity-based encryption without random oralces. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
24. 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)

A Proof of Theorem 4

Proof. In this proof, we suppose that the adversary does not cause hash collision. That is, it does not issue two message pair (M, ψ) and (M', ψ') such that $(M, \psi) \neq (M', \psi')$ but $H(M, \psi) = H(M', \psi')$. Note that if the adversary causes hash collision, it can be converted to an adversary for collision-resistant hash functions. Thus we can divide the adversary as two types according to their forgery as follows:

1. Type-1 adversary $\mathcal{A}_1$ is one of which forgery is not such that exactly one of the exponents $\{f_i\}$ equals 1, that is, $\sum_{i=1}^{a} f_i \neq 1$.
2. Type-2 adversary $\mathcal{A}_2$ is one of which forgery is such that exactly one of the exponents $\{f_i\}$ equals 1, that is, $\sum_{i=1}^{a} f_i = 1$.

For each type of adversary $\mathcal{A}_1$ and $\mathcal{A}_2$, we will construct algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively. The proof easily follows from following two lemmas and facts that the CDH attacker can be constructed from the discrete logarithm attacker and Waters two-level signature is secure under CDH assumption. □

Lemma 1. *If there exists a type-1 adversary $\mathcal{A}_1$, then there exists an algorithm $\mathcal{B}_1$ that solves the discrete logarithm problem on $\mathbb{G}_p$.*

Proof. Suppose there exists a type-1 adversary $\mathcal{A}_1$ that breaks unforgeability of our ID-based DNF signature scheme. The algorithm $\mathcal{B}_1$ that solves the discrete logarithm problem using $\mathcal{A}_1$ is given: The description of the bilinear group $\mathbb{G}$, the factorization p, q of order n, and the tuple (g_p, g_p^{α}) where g_p is a generator of $\mathbb{G}_p$. Its goal is to compute α. Then $\mathcal{B}_1$ that interacts with $\mathcal{A}_1$ is described as follows.

Setup: The algorithm $\mathcal{B}_1$ selects random generators $(g, g_2, v', v_1, \ldots, v_m) \in \mathbb{G}^{m+3}$, $h \in \mathbb{G}_q$, and random exponents $(\gamma, x', x_1, \ldots, x_l, y) \in \mathbb{Z}_n^{l+3}$ with restriction that $y \mod p \neq 0$. Next it selects a collision-resistant hash function H and sets $\mathsf{PP} = (n, \mathbb{G}, \mathbb{G}_T, e, g, g_1 = g^{\gamma}, g_2, h,\ u' = g^{x'}, u_1 = g^{x_1}, \ldots, u_l = g^{x_l}, v', v_1, \ldots, v_m,\ w = (g_p^{\alpha} h)^y, H)$ and $\mathsf{MK} = g_2^{\gamma}$.

Queries: $\mathcal{B}_1$ can correctly response to $\mathcal{A}_1$'s various queries, since it knows the master secret key.

Output: Finally, $\mathcal{A}_1$ outputs (σ^*, M^*, ψ^*) where $\sigma^* = (\{(\sigma_{1,j}, \sigma_{2,j}, \sigma_{3,j})\}_{1 \leq j \leq b}, \{(\{(C_{i,j}, \pi_{i,j}^{C})\}_{1 \leq j \leq b}, \pi_i^{col})\}_{1 \leq i \leq a}, \{(D_{j,k}, \pi_{j,k}^{D})\}_{1 \leq j \leq b, 1 \leq k \leq l})$ and $\psi^* = \bigvee_{i=1}^{a} \bigwedge_{j=1}^{b} \mathsf{ID}_{i,j}$.

If (1) the corrupted identities set $C = \{\mathsf{ID}_i\}_{1 \leq i \leq q_E}$ by private key extraction queries satisfies the DNF formula ψ^*; or (2) let S be the set of identity that was requested an individual signatures queries for (M^*, ψ^*), then $S \cup C$ satisfies the DNF formula ψ^*; or (3) $\mathcal{A}$ did request a signature for a pair (M^*, ψ^*); or (4) $\mathsf{Verify}(\sigma^*, M^*, \psi^*, \mathsf{PP}) \neq$ "accept", then $\mathcal{B}_1$ stops the simulation because $\mathcal{A}_1$ was not successful. Otherwise, $\mathcal{B}_1$ can solve the given problem as follows: First, it recovers $\{f_i\}_{1 \leq i \leq a}$ by setting $f_i = 0$ if $C_{i,j}^{\delta_p} = 1$ or $f_i = 1$ otherwise. Let $f = \sum_{i=1}^{a} f_i$, then $f \neq 1$, because $\mathcal{A}_1$ is a type-1 adversary. Let I be a set of index i such that $f_i = 1$ and $Y_{i,j} = u' \prod_{k=1}^{l} u_k^{\kappa_{i,j,k}}$. We obtain

$$w \prod_{i=1}^{a} C_{i,j} = w \cdot \prod_{i \in I} (Y_{i,j}/w) \cdot h^{z_j^{row}} = w^{1-f} \cdot \prod_{i \in I} Y_{i,j} \cdot h^{z_j^{row}}.$$

Since the signature is valid one, it should satisfy the verification equation by unknown identity. That is, there exists $\mathsf{ID}_j^+ = (\kappa_{j,1}^+, \ldots, \kappa_{j,l}^+)$ such that $w \prod_{i=1}^a C_{i,j} = Y_j^+ \cdot h^{z_j^{row}}$ where $Y_j^+ = u' \prod_{k=1}^l u_k^{\kappa_{j,k}^+}$, because of the equation

$$w \prod_{i=1}^{a} C_{i,j} = u' \prod_{k=1}^{l} D_{j,k} = u' \prod_{k=1}^{l} u_k^{\kappa_{j,k}^+} \cdot h^{z_j^{row}} = Y_j^+ \cdot h^{z_j^{row}}.$$

Then $\mathcal{B}_1$ recovers the identity ID_j^+ from $\{(D_{j,k}, \pi_{j,k}^D)\}$ by setting $\kappa_{j,k}^+ = 0$ if $D_{j,k}^{\delta_p} = 1$ or $\kappa_{j,k}^+ = 1$ otherwise. Let $F(\mathsf{ID}_{i,j}) = x' + \sum_{k=1}^l \kappa_{i,j,k} \cdot x_k$ where $\mathsf{ID}_{i,j} = (\kappa_{i,j,1}, \ldots, \kappa_{i,j,1})$. We obtain the following equation from the above two equations by raising δ_p to both sides,

$$\big(w^{(1-f)}\big)^{\delta_p} = \big(Y_j^+ / \prod_{i \in I} Y_{i,j}\big)^{\delta_p} = \big(g^{F(\mathsf{ID}_j^+) - \Sigma_{i \in I} F(\mathsf{ID}_{i,j})}\big)^{\delta_p}.$$

Additionally, we have $w = (g_p^\alpha h)^y$, $y \mod p \neq 0$, and $f \neq 1$. Therefore it solves the given discrete logarithm problem as follows:

$$\alpha = (F(\mathsf{ID}_j^+) - \sum_{i \in I} F(\mathsf{ID}_{i,j})) \cdot y^{-1} \cdot (1-f)^{-1} \mod p.$$

Analysis. Let $\mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{DL}}$ be the advantage of $\mathcal{B}_1$ that breaks the discrete logarithm problem. Since $\mathcal{B}_1$ succeeds whenever $\mathcal{A}_1$ does, we have $\mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{DL}} \geq \mathsf{Adv}_{\mathcal{A}_1}^{\mathsf{IBDNF\text{-}UF}}$. This completes our proof. □

Lemma 2. *If there exists a type-2 adversary $\mathcal{A}_2$, then there exists an algorithm $\mathcal{B}_2$ that breaks the unforgeability of Waters two-level signature scheme.*

Proof. Suppose there exists a type-2 adversary $\mathcal{A}_2$ that breaks unforgeability of our ID-based DNF signature scheme. The algorithm $\mathcal{B}_2$ that forges Waters two-level signature using $\mathcal{A}_2$ is given: The description of the bilinear group $\mathbb{G}$, the factorization p, q of order n, and the public parameter of Waters two-level signature as $\tilde{\mathsf{PP}} = (p, \mathbb{G}_p, \mathbb{G}_{T_p}, e, \tilde{g}, \tilde{g}_1, \tilde{g}_2, \tilde{u}', \tilde{u}_1, \ldots, \tilde{u}_l, \tilde{v}', \tilde{v}_1, \ldots, \tilde{v}_m, H)$ where all is in subgroups of order p. Then $\mathcal{B}_2$ that interacts with $\mathcal{A}_2$ is described as follows.

Setup: The algorithm $\mathcal{B}_2$ selects random $(f, f_2, h, \gamma', \gamma_1, \ldots, \gamma_l, \nu', \nu_1, \ldots, \nu_m) \in \mathbb{G}_q^{l+m+5}$, $w \in \mathbb{G}$, and a random exponent $\beta \in \mathbb{Z}_q^*$. Next it sets $\mathsf{PP} = (n, \mathbb{G}, \mathbb{G}_T, e, g = \tilde{g}f, g_1 = \tilde{g}_1 f^\beta, g_2 = \tilde{g}_2 f_2, h, u' = \tilde{u}'\gamma', u_1 = \tilde{u}_1\gamma_1, \ldots, u_l = \tilde{u}_l\gamma_l, v' = \tilde{v}'\nu', v_1 = \tilde{v}_1\nu_1, \ldots, v_m = \tilde{v}_m\nu_m, w, H)$ and gives PP to $\mathcal{A}_2$. The PP are correctly distributed.

Queries: For a private key extraction query on ID, $\mathcal{B}_2$ first asks the private key of Waters two-level signature and receives $\tilde{\mathsf{SK}}_{\mathsf{ID}} = (\tilde{K}_1, \tilde{K}_2) \in \mathbb{G}_p^2$, then it chooses a random $s \in \mathbb{Z}_q$ and constructs the private key as $\mathsf{SK}_{\mathsf{ID}} = (K_1 = \tilde{K}_1 \cdot f^\beta \cdot (\gamma' \prod_{i=1}^l \gamma_i^{\kappa_i})^s,\ K_2 = \tilde{K}_2 \cdot f^s,\ K_3 = h^s)$.

For an individual signature query on (M, ψ, ID), $\mathcal{B}_2$ first asks the signature of Waters two-level signature and receives $\tilde{\theta} = (\tilde{\theta}_1, \tilde{\theta}_2, \tilde{\theta}_3)$, then it chooses random $s, r \in \mathbb{Z}_q$ and constructs the signature as $\theta = (V = \tilde{\theta}_1 \cdot f^{\beta} \cdot (\gamma' \prod_{i=1}^{l} \gamma_i^{\kappa_i})^s \cdot (\nu' \prod_{i=1}^{m} \nu_i^{\mu_i})^r,\ S = \tilde{\theta}_2 \cdot f^s,\ T = h^s,\ R = \tilde{\theta}_3 \cdot f^r)$.

For a signature query on (M, ψ) where $\psi = \vee_{i=1}^{a} \wedge_{j=1}^{b_i} \mathsf{ID}_{i,j}$, $\mathcal{B}_2$ first selects an arbitrary index i^* and constructs individual signatures of $\mathsf{ID}_{i^*,j}$ for all j, then it creates the final signature using Merge algorithm.

Output: Finally, $\mathcal{A}_2$ outputs (σ^*, M^*, ψ^*) where $\sigma^* = (\{(\sigma_{1,j}, \sigma_{2,j}, \sigma_{3,j})\}_{1 \le j \le b}, \{(\{(C_{i,j}, \pi_{i,j}^{C})\}_{1 \le j \le b}, \pi_i^{col})\}_{1 \le i \le a}, \{(D_{j,k}, \pi_{j,k}^{D})\}_{1 \le j \le b, 1 \le k \le l})$ and $\psi^* = \vee_{i=1}^{a} \wedge_{j=1}^{b} \mathsf{ID}_{i,j}$.

If (1) the corrupted identities set $C = \{\mathsf{ID}_i\}_{1 \le i \le q_E}$ by private key extraction queries satisfies the DNF formula ψ^*; or (2) let S be the set of identity that was requested an individual signatures queries for (M^*, ψ^*), then $S \cup C$ satisfies the DNF formula ψ^*; or (3) $\mathcal{A}$ did request a signature for a pair (M^*, ψ^*); or (4) $\mathsf{Verify}(\sigma^*, M^*, \psi^*, \mathsf{PP}) \neq$ "accept", then $\mathcal{B}_2$ stops the simulation because $\mathcal{A}_2$ was not successful. Otherwise, $\mathcal{B}_2$ can convert the signature to Waters two-level signature as follows: First, it recovers $\{f_i\}_{1 \le i \le a}$ by setting $f_i = 0$ if $C_{i,j}^{\delta_p} = 1$ or $f_i = 1$ otherwise. Since $\mathcal{A}_2$ is a type-2 adversary, there is exactly one index i^* such that $f_{i^*} = 1$. Using the index i^*, the signers identities $\{\mathsf{ID}_{i^*,j}\}_{1 \le j \le b}$ can be reconstructed from ψ. Let the index j^* be such that neither the private key extraction for ID_{i^*,j^*} and an individual signature on (M^*, ψ^*) by ID_{i^*,j^*} was queried by $\mathcal{A}_2$. By the conditions of $\mathcal{A}_2$'s valid forgery, the index j^* always exists. We obtain from the verification equation by raising δ_p,

$$\begin{aligned} e(\tilde{g}, \sigma_{1,j^*}^{\delta_p}) &= e(\tilde{g}_1, \tilde{g}_2) \cdot e(\sigma_{2,j^*}^{\delta_p}, (w \prod_{i=1}^{a} C_{i,j^*})^{\delta_p}) \cdot e(\sigma_{3,j^*}^{\delta_p}, (v' \prod_{i=1}^{m} v_{j^*}^{\mu_i})^{\delta_p}) \\ &= e(\tilde{g}_1, \tilde{g}_2) \cdot e(\sigma_{2,j^*}^{\delta_p}, \tilde{u}' \prod_{i=1}^{l} \tilde{u}_{j^*}^{\kappa_i}) \cdot e(\sigma_{3,j^*}^{\delta_p}, \tilde{v}' \prod_{i=1}^{m} \tilde{v}_{j^*}^{\mu_i}). \end{aligned}$$

Thus $(\sigma_{1,j^*}^{\delta_p}, \sigma_{2,j^*}^{\delta_p}, \sigma_{3,j^*}^{\delta_p})$ is a valid Waters two-level signature on (M^*, ψ^*) by the identity ID_{i^*,j^*}. Then $\mathcal{B}_2$ outputs it and halts.

Analysis. Let $\mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{W\text{-}IBS}}$ be the advantage of $\mathcal{B}_2$ that breaks Waters two-level signature scheme. Since $\mathcal{B}_2$ succeeds whenever $\mathcal{A}_2$ does, we have $\mathsf{Adv}_{\mathcal{B}_2}^{\mathsf{W\text{-}IBS}} \ge \mathsf{Adv}_{\mathcal{A}_2}^{\mathsf{IBDNF\text{-}UF}}$. This completes our proof. □

How to Balance Privacy with Authenticity[*]

Pairat Thorncharoensri, Willy Susilo, and Yi Mu

Centre for Computer and Information Security
School of Computer Science & Software Engineering
University of Wollongong, Australia
{pt78,wsusilo,ymu}@uow.edu.au

Abstract. In several occasions, it is important to consider the privacy of an individual together with the authenticity of the message produced by that individual or hold by that individual. In the latter scenario, the authenticity of the message enables one to prove that the message that he/she holds is authentic to other people. Nonetheless, this will normally incur that the privacy of the signer will be exposed at the same time. In this paper, we consider a situation where the authenticity of the message will be ensured together with the privacy of the signature holder, if and only if the signature is designated *once* to a third party. However, as soon as there is more than one designation occurs, then the privacy of the signer (and the signature holder) will cease. We consider real scenarios where this type of notion is required. We formalize this notion as a *one-time universal designated verifier signature*, and for the first time in the literature, we provide a concrete scheme to realize this primitive.

Keywords: privacy, authenticity, universal designated verifier signature schemes, one time, non-transferability.

1 Introduction

Consider a real life scenario where a patient Henry has been identified with some sensitive disease, such as AIDS, by the doctor Susan. Susan will provide a statement for Henry so that Henry can be referred to a specialist, who will be able to help Henry. Nonetheless, due to the medical restriction, Henry is only permitted to see *one* specialist who will conduct further test on him. In this case, we can observe that there are *two* important properties attached to this scenario, namely the *privacy* of the patient together with the *authenticity* provided by Susan. Firstly, Susan needs to provide an authenticated statement to Henry so that Henry can convince a specialist of his choice, but with limitation that this statement can only be used *once*. Therefore, there must be a mechanism to stop Henry to use this authentication statement for more than once. We envisage that this scenario can be realized if there is a primitive that will behave as follows. Susan can issue a message m and sign the message using her secret key $\mathcal{SK}_S$,

[*] This work is partially supported by ARC Linkage Project Grant LP0667899.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 184–201, 2009.

to produce a signature σ. Note that σ can be verified publicly using Susan's public key $\mathcal{PK}_S$, but this signature will only be provided to Henry via a secure and authenticated channel, and hence, nobody can verify the authenticity of this signature. When Henry wants to convince a specialist, say Charlie, then Henry will *designate* Susan's signature, i.e. to convert σ to $\widetilde{\sigma}$, to Charlie, such that only Charlie will be convinced with the authenticity of $\widetilde{\sigma}$. This process should be guaranteed such that $\widetilde{\sigma}$ will not allow anyone other than Charlie to be convinced about the authenticity of the message m, issued originally by Susan. Nonetheless, we have to also restrict Henry so that he can only designate the signature *once*. That means, if Henry issues another designated signature $\bar{\sigma}$ on the same σ, then the knowledge of $\bar{\sigma}$ and $\widetilde{\sigma}$ should be able to be used to derive σ. This way, Henry's privacy will not be provided any longer, if he misbehaves. This problem seems quite intriguing but unfortunately, there is *no* existing cryptographic primitive that can be used to realize this scenario. We will demonstrate that there exist a few primitives in the literature that is quite close to this scenario, but unfortunately they cannot be used adequately to solve this problem.

The second motivating example is derived from the online game known as Ragnarok[1]. This online game provides a way to allow the users to trade their items during its gameplay. We note that this feature is not unique to Ragnarok, but it also appears in some other online games, such as Gaiaonline[2], etc. The normal requirement in the trading in these games is there is no information of the traders in the game itself. The scenario is as follows. Suppose a player, Hugh, wants to convince another player, Vicky, that he indeed has some money (or points) or some rare items in the online game and he would like to trade it with her. The online game administrator is the one who provides a signature on Hugh's items. Here, Hugh has to convince Vicky (and only Vicky) when the trade happens. If Hugh wants to cheat, then he will try to convince another person but this will indeed reveal his *privacy* on the administrator's signature.

Previous Works

As mentioned earlier, there is *no* primitive that provides what is required in the above scenario. Nonetheless, there are some related primitives in the literature that are closely related to what is required above.

The notion of designated verifier signatures was introduced by Jakobsson, Sako and Impagliazzo in [4]. In this notion, a signature does not provide only authentication of a message but it also provides the deniability property that allows the signer to deny the signature (since the verifier can also generate such a signature). Only the designated verifier will be convinced with the authenticity of the signature on the message, since the verifier can always construct this signature by himself. The topics on designated verifier signatures areas have been widely studied, for example [8,7,10,11,1,13].

In Asiacrypt 2003, Steinfeld, Bull, Wang and Pieprzyk [12] introduced an interesting cryptographic primitive called "Universal Designated-Verifier

[1] http://ragnarok-online-2.com/game/guide/items

[2] http://www.gaiaonline.com

Signature (UDVS) Scheme". A UDVS scheme is an ordinary signature scheme with an additional functionality that provides a privilege to *a signature holder* to designate the signature to any verifier that is chosen by him. To protect the privacy of the original signer, a designated verifier signature generated by the signature holder is designed to convince only the designated verifier. UDVS provides a very close feature that is required in the above application.

The first UDVS scheme without random oracle model was proposed by Zhang, Furukawa and Imai in [15]. In SCN 2006, Laguillaumie, Libert and Quisquater proposed two efficient UDVS scheme in the standard model [6]. Introduced by Huang et al. in [3], the restricted UDVS scheme is a UDVS scheme with additional functionality to limit a power of signature holder in generating a UDVS signature for only k times. If the signature holder generated more than k times then k UDVS signatures are used to deduce a standard signature. In this setting, a signature holder is the one who controls the limited number(k) of designated verifier signature and coefficients in polynomial used to limit the number of designations. In ISC 2007, Laguillaumie and Vergnaud argued that the restricted UDVS scheme in [3] has failed to achieve the *restriction* property [9]. Nonetheless, there is *no way* to limit the ability of the signature holder to only convince *one* single verifier.

Our Contributions
In this paper, we introduce the notion of One-Time Universal Designated Verifier Signature (OT-UDVS) scheme to capture the above requirements. We provide a model to capture this notion and present a concrete scheme and its security proof to show that our scheme is secure in our model. Additionally, our security model also captures a collusion between a malicious signer and a malicious verifier.

Organization of The Paper
The paper is organized as follows. In the next Section, we will review some preliminaries that will be used throughout this paper. In Section 3 and 4, the definition of OT-UDVS and its security model will be presented. In Section 5, we will present the overview of the building blocks required for constructing our concrete OT-UDVS scheme. We present our OT-UDVS scheme in Section 6. Then, the security proof of our concrete scheme is presented in Section 7. Finally, we conclude the paper.

2 Preliminaries

2.1 Notation

The following notations will be used throughout the paper. We denote a probabilistic polynomial-time algorithm by PPT. When a PPT algorithm F privately accesses and executes another PPT algorithm E, we denote it by $F^{E(.)}(.)$. Let $poly(.)$ be a deterministic polynomial function. We say that q is polynomial-time in k if $q \leq poly(1^k)$ for all polynomials $poly(k)$ for all sufficiently large k. If a function $f : \mathbb{N} \rightarrow \mathbb{R}$ is said to be *negligible*, then $f(n) < \frac{1}{n^c}$ holds for all constant $c > 0$ and for all sufficiently large n. The operation of picking l at random from a (finite) set L is denoted by $l \xleftarrow{\$} L$. We said a collision of a function $h(.)$ to refer to the case where there are message pair m, n of distinct points in its message space such that $h(m) = h(n)$. The symbol $||$ denotes the concatenation of two strings (or integers).

2.2 Bilinear Pairing

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic multiplicative groups generated by g_1 and g_2, respectively. The order of both generators is a prime p. Let $\mathbb{G}_T$ be a cyclic multiplicative group with the same order p. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear mapping with the following properties:

1. *Bilinearity:* $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$ for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2\ , a, b \in \mathbb{Z}_p$.
2. *Non-degeneracy:* There exist $g_1 \in \mathbb{G}_1$ *and* $g_2 \in \mathbb{G}_2$ such that $\hat{e}(g_1, g_2) \neq 1$.
3. *Computability:* There exists an efficient algorithm to compute $\hat{e}(g_1, g_2)$ for all $g_1 \in \mathbb{G}_1,\ g_2 \in \mathbb{G}_2$.

Note that there exists $\varphi(.)$ function which maps $\mathbb{G}_1$ to $\mathbb{G}_2$ or vice versa in one time unit.

2.3 Complexity Assumptions

Definition 1 (Computation Diffie-Hellman (CDH) Problem). *Given a* 3*-tuple* (g, g^x, g^y) *as input, output* $g^{x \cdot y}$. *An algorithm* $\mathcal{A}$ *has advantage* ϵ *in solving the CDH problem if*

$$\Pr\left[\mathcal{A}(g, g^x, g^y) = g^{x \cdot y}\right] \geq \epsilon$$

where the probability is over the random choice of $x, y \in \mathbb{Z}_q^*$ *and the random bits consumed by* $\mathcal{A}$.

Assumption 1. (t, ϵ)**-Computation Diffie-Hellman Assumption.** We say that the (t, ϵ)-CDH assumption holds if no PPT algorithm with time complexity $t(.)$ has advantage at least ϵ in solving the CDH problem.

3 One-Time Universal Designated Verifier Signature Schemes (OT-UDVS)

We assume that all parties must comply a registration protocol with a certificate authority CA to obtain a certificate on their public parameters. We give a definition of one time universal designated verifier signature scheme as follows.

Definition 2. *A one time universal designated verifier signature scheme* Σ *is an* 8*-tuple(SKeyGen, Sign,* $Verify$, $VKeyGen$, *Delegate*, $DVerify$, *DSimulate*, *Open) such that*

Signature Scheme Setup: *A signature scheme comprises of three PPT algorithms (SKeyGen, Sign,* $Verify$*).*

- *Signer's Public Parameters and Secret Key Generator ($\Sigma.SKeyGen$):* $\Sigma.SKeyGen$ *is a PPT algorithm that, on input a security parameter* $\mathcal{K}$, *outputs the secret key* (sk_S) *and the public parameter* (pk_S) *of signer. That is* $\{pk_S, sk_S\} \leftarrow \Sigma.SKeyGen(1^{\mathcal{K}})$.
- *Signature Signing ($\Sigma.Sign$):* *On input a signing secret key* sk_S, *public parameters* pk_S, *a message* m, $\Sigma.Sign$ *outputs signer's signature* σ. *That is* $\sigma \leftarrow \Sigma.Sign(m, sk_S, pk_S)$.
- *Signature Verification ($\Sigma.Verify$):* *On input signer's public parameters* pk_S, *a message* m *and a signature* σ, *output a verification decision* $d \in \{Accept, Reject\}$. *That is* $d \leftarrow \Sigma.Verify(m, \sigma, pk_S)$.

Verifier's Public Parameters and Key Generator ($\Sigma.VKeyGen$): $\Sigma.VKeyGen$ *is a PPT algorithm that, on input a security parameter* $\mathcal{K}$, *outputs strings* (sk_V, pk_V) *where they denote the secret key and the public parameter of signer, respectively. That is* $\{pk_V, sk_V\} \leftarrow \Sigma.VKeyGen(1^{\mathcal{K}})$.

Signature Delegation ($\Sigma.Delegate$) : *On input verifier's public parameters* pk_V, *signer's public parameters* pk_S, *a signer's signature* σ, *and a message* m, $\Sigma.Delegate$ *outputs a designated verifier signature* $\hat{\sigma}$. *That is* $\hat{\sigma} \leftarrow \Sigma.Delegate(m, \sigma, pk_V, pk_S)$.

Delegated Signature Verification ($\Sigma.DVerify$) : *On input verifier's public parameters* pk_V, *verifier's secret key* sk_V, *signer's public parameters* pk_S, *a message* m *and a designated verifier signature* $\hat{\sigma}$, $\Sigma.DVerify$ *outputs a verification decision* $d \in \{Accept, Reject\}$. *That is*

$$d \leftarrow \Sigma.DVerify(m, \hat{\sigma}, pk_V, sk_V, pk_S).$$

Simulation of a Delegated Signature ($\Sigma.DSimulate$): *On input verifier's public parameters* pk_V, *verifier's secret key* sk_V, *signer's public parameters* pk_S, *and a message* m, $\Sigma.DSimulate$ *outputs a designated verifier signature* $\bar{\sigma}$ *such that* $Valid \leftarrow \Sigma.DVerify(m, \bar{\sigma}, pk_V, sk_V, pk_S)$. *That is* $\bar{\sigma} \leftarrow \Sigma.DSimulate(m, pk_V, sk_V, pk_S)$.

Opening a Delegated Signature ($\Sigma.Open$) : $\Sigma.Open$ *is a PPT algorithm that takes two designated verifier signatures* $\dot{\sigma}, \ddot{\sigma}$ *and their designated verifier's public parameters* $pk_{\dot{V}}, pk_{\ddot{V}}$, *signer's public parameters* pk_S, *and a message* m *as inputs. It outputs a signer signature* σ. *That is*

$$\sigma \leftarrow \Sigma.Open(\dot{\sigma}, \ddot{\sigma}, pk_{\dot{V}}, pk_{\ddot{V}}, pk_S, m).$$

For all $\mathcal{K} \in \mathbb{N}$, *all* $(pk_S, sk_S) \in \Sigma.SKeyGen(1^{\mathcal{K}})$, *all* $(pk_V, sk_V) \in \Sigma.VKeyGen(1^{\mathcal{K}})$ *and all messages* m, Σ *must satisfy the following properties:*

Completeness of a Signature:

$$\forall \sigma \in \Sigma.Sign(m, sk_S, pk_S), \Pr[\Sigma.Verify(m, \sigma, pk_S) = Valid] = 1. \quad (1)$$

Completeness of a Designated Verifier Signature:

$$\begin{aligned} &\forall \hat{\sigma} \in \Sigma.Delegate(m, \sigma, pk_V, pk_S), \\ &\Pr[\Sigma.DVerify(m, \hat{\sigma}, pk_V, sk_V, pk_S) = Valid] = 1. \end{aligned} \quad (2)$$

Completeness of a Simulated Designated Verifier Signature:

$$\begin{aligned}&\forall\bar{\sigma} \in \Sigma.DSimulate(m, pk_V, sk_V, pk_S),\\&\Pr[\Sigma.DVerify(m, \bar{\sigma}, pk_V, sk_V, pk_S) = Valid] = 1.\end{aligned} \tag{3}$$

Completeness of an Opened Signature:

$$\begin{aligned}&\forall\dot{\sigma} \in \Sigma.Delegate(m, \sigma, pk_{\dot{V}}, pk_S); \forall\ddot{\sigma} \in \Sigma.Delegate(m, \sigma, pk_{\ddot{V}}, pk_S):\\&\Sigma.DVerify(m, \dot{\sigma}, pk_V, sk_V, pk_S) = Valid;\\&\Sigma.DVerify(m, \ddot{\sigma}, pk_V, sk_V, pk_S) = Valid;\\&\sigma \leftarrow \Sigma.Open(\dot{\sigma}, \ddot{\sigma}, pk_{\dot{V}}, pk_{\ddot{V}}, pk_S, m),\\&\Pr[\Sigma.Verify(m, \sigma, pk_S) = Valid] = 1.\end{aligned} \tag{4}$$

We refer to a signature to be 'open' if there are two valid designated signatures dedicated to two different designated verifiers issued by the signature holder.

Source Hiding:

$$\begin{aligned}&\forall\hat{\sigma} \in \Sigma.Delegate(m, \sigma, pk_V, pk_S); \forall\bar{\sigma} \in \Sigma.DSimulate(m, pk_V, sk_V, pk_S):\\&\Sigma.DVerify(m, \hat{\sigma}, pk_V, sk_V, pk_S) = Valid;\\&\Sigma.DVerify(m, \bar{\sigma}, pk_V, sk_V, pk_S) = Valid;\\&\sigma \xleftarrow{\$} \{\hat{\sigma}, \bar{\sigma}\}, |\Pr[\sigma = \hat{\sigma}] - \Pr[\sigma = \bar{\sigma}]| \text{ is } \textit{negligible.}\end{aligned} \tag{5}$$

In the next section, we will provide the details of formal definitions of the security models for OT-UDVS scheme. These include the unforgeability, the single designatability and the non-transferability privacy (which implies the source hiding property). The completeness of signature, designated verifier signature, simulated designated verifier signature and opened signature are straightforward as the above properties, and hence, it will be omitted.

4 Security Models of OT-UDVS Schemes

The following subsection describes the formal definitions of the unforgeability, the non-transferability privacy and the single designatability. The unforgeability and the non-transferability privacy are introduced in [12]. For the single designatability, this property is firstly introduced in this paper to capture the requirement of the OT-UDVS scheme. The single designatability property captures the sense that no signature holder can convince more than one designated verifier or produce more than one designated verifier signature. If he/she did it, then the original signature will be revealed, as the proof of this misbehaviour.

4.1 Unforgeability

It is mentioned in [12,2] that there are actually two types of unforgeability properties to consider which are "standard signature unforgeability" and "designated verifier unforgeability". It is also concluded that "designated verifier unforgeability" always implies "standard signature unforgeability" [2]. Hence, for the rest of this paper, when we discuss about unforgeability property, it indeed refers to "designated verifier unforgeability". To capture a collusion attack that is launched by a malicious signature holder, malicious signers and malicious verifiers, chosen public key attacks play a significant role in this security model. This attack simulates a situation that in addition to the target signer and the target verifier, an adversary possesses the knowledge of secret keys of other signers or verifiers, and designated verifier signatures prior to the attack. This attack reflects the same situation where a collusion happens among a malicious signature holder, malicious signers and malicious verifiers.

The unforgeability property provides a security against existential unforgeability under adaptive chosen message and chosen public key attack. It intentionally prevents an attacker corrupted with signature holder to generate an (OT-U)DVS signature $\hat{\sigma_*}$ on a new message M^*. Formally, the unforgeability provides an assurance that given the signer public parameters pk_S and an access to signing oracle, delegation oracle, and (designated verifier) verification oracle, one, accessing oracles with its arbitrarily choice of verifier's public parameters pk_V and choice of message m as inputs, should be unable to produce a designated verifier signature on a new message. We denote by *CM-CPK-A* the adaptively chosen message and chosen public key attack, and by *EUF-OT-UDVS* the existential unforgeability of OT-UDVS scheme. Let $\mathcal{A}^{CM-CPK-A}_{EUF-OT-UDVS}$ be the adaptively chosen message and chosen public key adversary and let $\mathcal{F}$ be a simulator. The following game between $\mathcal{F}$ and $\mathcal{A}$ is defined to describe the existential unforgeability of OT-UDVS scheme:

SKeyGen **queries :** At most q_{SKG}, $\mathcal{A}$ can make a query for a public key of signer. In response, $\mathcal{F}$ runs the $\Sigma.SKeyGen$ algorithm to generate a secret key sk_S and public parameters pk_S of signer. $\mathcal{F}$ replies $\mathcal{A}$ with pk_S.

Sign **queries :** At most q_S, $\mathcal{A}$ can make a query for a signature σ on its choice of message m under its choice of signer public parameters pk_S. In response, $\mathcal{F}$ runs the $\Sigma.Sign$ algorithm to generate a signature σ on a message m corresponding with pk_S. $\mathcal{F}$ then returns σ, m to $\mathcal{A}$.

$Verify$ **queries :** At most q_V, $\mathcal{A}$ can make a query for the verification of a signature σ on a message m with its corresponding public parameters pk_S. In response, $\mathcal{F}$ runs the $\Sigma.Verify$ algorithm to verify a signature σ on a message m corresponding with pk_S $\mathcal{F}$ outputs *Accept* if a signature σ on a message m is valid regrading to pk_S, otherwise, it outputs $Reject$.

VKeyGen **queries :** At most q_{VKG}, $\mathcal{A}$ can make a query for public parameters pk_V of verifier In response, $\mathcal{F}$ runs the $\Sigma.VKeyGen$ algorithm to generate a secret key sk_V and public parameters pk_V of a verifier. $\mathcal{F}$ replies $\mathcal{A}$ with pk_V.

Delegate **queries :** At most q_D, $\mathcal{A}$ can make a query for a designated verifier signature $\hat{\sigma}$ on its choice of message m under its choice of signer public parameters pk_S and verifier public parameters pk_V. In response, $\mathcal{F}$ runs the $\Sigma.Delegate$ algorithm to generate a designated verifier signature $\hat{\sigma}$ on a message m corresponding with pk_S, pk_V. $\mathcal{F}$ then returns $\hat{\sigma}, m$ to $\mathcal{A}$.

DVerify **queries :** At most q_{DV}, $\mathcal{A}$ can make a query for the verification of a designated verifier signature $\hat{\sigma}$ (or $\bar{\sigma}$) on its chosen message m with its corresponding public parameters pk_S, pk_V. In response, $\mathcal{F}$ runs the $\Sigma.DVerify$ algorithm to verify a designated verifier signature $\hat{\sigma}$ (or $\bar{\sigma}$) on a message m corresponding with pk_S, pk_V $\mathcal{F}$ outputs *Accept* if a designated verifier signature $\hat{\sigma}$ (or $\bar{\sigma}$) on a message m is valid regrading to pk_S, pk_V, otherwise, it outputs *Reject*.

DSimulate **queries :** At most q_{DS}, under its choice of signer public parameters pk_S and verifier public parameters pk_V, $\mathcal{A}$ can make a query for a (simulated) designated verifier signature $\bar{\sigma}$ on its choice of message m which $\bar{\sigma}$ must indeed generated by verifier. In response, $\mathcal{F}$ runs the $\Sigma.DSimulate$ algorithm to generate a (simulated) designated verifier signature $\bar{\sigma}$ on a message m corresponding with pk_S, pk_V. $\mathcal{F}$ then returns $\bar{\sigma}, m$ to $\mathcal{A}$.

Open **queries :** At most q_O, under its choice of signer public parameters pk_S and verifier public parameters $pk_{\dot{V}}, pk_{\ddot{V}}$, $\mathcal{A}$ can make a query to open a signature σ on a message m from two designated verifier signature $\dot{\sigma}, \ddot{\sigma}$. In response, $\mathcal{F}$ runs the $\Sigma.Open$ algorithm to open a signature σ on a message m from $\dot{\sigma}, \ddot{\sigma}$ and its corresponding $pk_S, pk_{\dot{V}}, pk_{\ddot{V}}$. $\mathcal{F}$ outputs σ if $\dot{\sigma}, \ddot{\sigma}$ signatures on a message m is valid regrading to $pk_S, pk_{\dot{V}}, pk_{\ddot{V}}$, otherwise, it outputs *Reject*.

Key **queries :** At most q_K, $\mathcal{A}$ can make a query for secret key sk_S (or sk_V) corresponding to public parameters pk_S (or pk_V) of signer (or verifier). In response, $\mathcal{F}$ replies $\mathcal{A}$ with a corresponding secret key sk_S (or sk_V).

At the end of the above queries, we assume that $\mathcal{A}$ outputs a forged signature $\hat{\sigma}_*$ on a message M^* with respect to the public parameters pk_S^*, pk_V^*. We say that $\mathcal{A}$ win the game if

1. $Accept \leftarrow \Sigma.DVerify(M^*, \hat{\sigma}_*, pk_V^*, sk_V^*, pk_S^*)$.
2. Neither pk_S^* nor pk_V^* has been submitted as input of a query for a secret key to *Key* queries.
3. $\mathcal{A}$ never made a request for a signature on input M^*, pk_S^* to *Sign* queries.
4. $\mathcal{A}$ never made any request for a designated verifier signature on input M^*, pk_S^* to *Delegate* queries.
5. $\mathcal{A}$ never made any request for a designated verifier signature on input M^*, pk_V^* to *DSimulate* queries.

Let $Succ_{EUF-OT-UDVS}^{CM-CPK-A}(.)$ be the success probability function of that $\mathcal{A}_{EUF-OT-UDVS}^{CM-CPK-A}$ wins the above game.

Definition 3. *We say that OT-UDVS scheme is* $(t, q_H, q_{SKG}, q_S, q_V, q_{VKG}, q_D, q_{DV}, q_{DS}, q_O, q_K, \epsilon)$*-secure existentially unforgeable under a chosen message and chosen public key attack if there are no PPT CM-CPK-A adversary* $\mathcal{A}_{EUF-OT-UDVS}^{CM-CPK-A}$ *such that the success probability* $Succ_{EUF-OT-UDVS}^{CM-CPK-A}(k) = \epsilon$ *is negligible in* k*, where* $\mathcal{A}_{EUF-OT-UDVS}^{CM-CPK-A}$ *runs in time at most* t*, make at most* q_H*,* q_{SKG}*,* q_S*,* q_V*,* q_{VKG}*,* q_D*,* q_{DV}*,* q_{DS}*,* q_O*, and* q_K *queries to the random oracle, SKeyGen queries, Sign queries, Verify* *queries, $VKeyGen$ queries, Delegate queries, $DVerify$ queries, DSimulate queries, Open queries, and Key queries, respectively.*

4.2 Non-transferability Privacy

The source hiding property claims that, given verifier's public parameters pk_V, verifier's secret key sk_V, signer's public parameters pk_S, and a message m, one can compute a (simulated) designated verifier signature indistinguishable from a designated verifier signature generated by a signature holder. In addition, this property does not imply the non-transferability privacy property which is introduced in [12] and clarified in [2].

In addition to the requirement of the source hiding property, the non-transferability privacy property is required even one can obtain or review many designated verifier signatures $\hat{\sigma_1}, ..., \hat{\sigma_q}$ on the same message m designated to both same or different verifiers, where $\hat{\sigma_1}, ..., \hat{\sigma_q}$ are generated by the same signature holder using the same signature σ. However, in our OT-UDVS, the non-transferability privacy property is different from above description since signature holder can generated only one designated verifier signature $\hat{\sigma}$ per message per verifier. The original signature σ is otherwise revealed when one obtains two or more designated verifier signatures generated from the same signature σ. Hence, the non-transferability privacy property for OT-UDVS scheme is provided that (1) it is implied the source hiding property and, (2) even one can obtain or review many designated verifier signatures $\hat{\sigma}_1, ..., \hat{\sigma}_q$ on its choices of message $m_1, ..., m_q$ designated to both same or different verifiers, it is hard to convince other party that a signer indeed generated a signature $\hat{\sigma} \in \{\hat{\sigma}_1, ..., \hat{\sigma}_q\}$ on a message $m \in \{m_1, ..., m_q\}$.

We denote by *ENT-OT-UDVS* the existential non-transferable privacy of OT-UDVS scheme. Let $\mathcal{A}_{ENT-OT-UDVS}^{CM-CPK-A}$ be the adaptively chosen message and chosen public key distinguisher and let $\mathcal{F}$ be a simulator. The following game between $\mathcal{F}$ and $\mathcal{A}$ is defined to describe the existential non-transferable privacy of OT-UDVS scheme:

The game is divided into two phases. $\mathcal{F}$ first defines *SKeyGen*, *Sign*, $Verify$, $VKeyGen$, *Delegate*, $DVerify$, *DSimulate*, *Open*, Key queries and their responses in the same way as defined in the model of *unforgeability.* Then the game is run as follows:

1. **Phase** 1 **:** $\mathcal{A}$ can submit a request to *SKeyGen*, *Sign*, $Verify$, $VKeyGen$, *Delegate*, $DVerify$, *DSimulate*, *Open*, Key queries. The queries response as their design.
2. **Challenge :** When $\mathcal{A}$ decides to challenge $\mathcal{F}$, it puts forward M^*, pk_S^*, pk_V^* with the constraints that
   a. $\mathcal{A}$ never made a request for a signature on input M^*, pk_S^* to *Sign* queries.
   b. $\mathcal{A}$ never submitted a request for a designated verifier signature on input M^*, pk_S^* to *Delegate* queries.

- **c.** $\mathcal{A}$ never submitted a request for a designated verifier signature on input M^*,pk_V^* to *DSimulate* queries.
- **d.** pk_S^* has never been submitted as input of a query for a secret key to *Key* queries.

In return, $\mathcal{F}$ chooses bit $b \xleftarrow{\$} \{0,1\}$. If $b = 1$ then $\mathcal{F}$ returns $\hat{\sigma} \leftarrow \Sigma.Delegate(M^*, \sigma, pk_V^*, pk_S^*)$ from *Delegate* queries. Otherwise, $\mathcal{F}$ returns $\bar{\sigma} \leftarrow \Sigma.DSimulate(M^*, pk_V^*, sk_V^*, pk_S^*)$ from *DSimulate* queries.

3. **Phase** 2 **:** $\mathcal{A}$ can arbitrarily return to *Phase* 1 or *Challenge* as many as it want. The only constraint is that $\mathcal{A}$ must have at least one set of challenge M^*,pk_S^*,pk_V^* such that
   - **a.** $\mathcal{A}$ never submitted a request for a signature on input M^*,pk_S^* to *Sign* queries.
   - **b.** $\mathcal{A}$ never submitted a request for a designated verifier signature on input M^*,pk_S^* to *Delegate* queries.
   - **c.** $\mathcal{A}$ never submitted a request for a designated verifier signature on input M^*,pk_V^* to *DSimulate* queries.
   - **d.** $\mathcal{A}$ never submitted a request for a secret key sk_S^* corresponding with pk_S^* to *Key* queries.
4. **Guessing :** On the challenge M^*,pk_S^*,pk_V^*, $\mathcal{A}$ finally outputs a guess b'. The distinguisher wins the game if $b = b'$.

Let $Succ_{ENT-OT-UDVS}^{CM-CPK-A}(.)$ be the success probability function of that $\mathcal{A}_{ENT-OT-UDVS}^{CM-CPK-A}$ wins the above game.

Definition 4. *We say that OT-UDVS scheme is (t,q_H,q_{SKG},q_S,q_V,q_{VKG},q_D,q_{DV},q_{DS},q_O,q_K,ϵ)-secure existentially non-transferable privacy under a chosen message and chosen public key attack if there are no PPT CM-CPK-A distinguisher $\mathcal{A}_{ENT-OT-UDVS}^{CM-CPK-A}$ such that the success probability $Succ_{ENT-OT-UDVS}^{CM-CPK-A}(k) = |\Pr[b = b'] - \Pr[b \neq b']| = \epsilon$ is negligible in k, where $\mathcal{A}_{ENT-OT-UDVS}^{CM-CPK-A}$ runs in time at most t, make at most q_H, q_{SKG}, q_S, q_V, q_{VKG}, q_D, q_{DV}, q_{DS}, q_O, and q_K queries to the random oracle, SKeyGen queries, Sign queries, Verify queries, VKeyGen queries, Delegate queries, DVerify queries, DSimulate queries, Open queries, and Key queries, respectively.*

4.3 Single Designatability

A similar property is "Opening" which is introduced in [2] and analyzed by Laguillaumie and Vergnaud in [9]. They concentrated on the multi-time restricted delegation. However, Laguillaumie and Vergnaud argued there is always a proof for the restricted UDVS scheme in [2] that a signature holder can generate a designated verifier signature from a signature without getting a penalty for over spending. We argue that, for our OT-UDVS scheme, the signature holder can be restricted such that he/she can only convince one verifier with one designated verifier signature without getting a penalty.

The single designatability property provides security against existential single designatability under adaptive chosen message and chosen public key attack. Institutively, it prevents an attacker corrupted with signature holder to generate two (one-time universal) designated verifier signatures $\dot{\sigma}, \ddot{\sigma}$ on a message m such that both signatures are valid on the same message generated by the same signature holder, however, they could not be opened to reveal a original signature σ generated by the signer.

We denote by *ESD-OT-UDVS* the existential single designatablility of OT-UDVS scheme. Let $\mathcal{A}^{CM-CPK-A}_{ESD-OT-UDVS}$ be the adaptively chosen message and chosen public key adversary and let $\mathcal{F}$ be a simulator. The following game between $\mathcal{F}$ and $\mathcal{A}$ is defined to describe the existential single designatability of OT-UDVS scheme:

To initiate a simulation, $\mathcal{F}$ first defines $SKeyGen$, $Sign$, $Verify$, $VKeyGen$, *Delegate*, $DVerify$, *DSimulate*, *Open*, Key queries and their responses in the same way as defined in the model of *unforgeability.* $\mathcal{A}$ can access arbitrarily to an random oracle and the above queries. At the end of the above queries, we assume that $\mathcal{A}$ outputs two designated verifier signatures $\dot{\sigma}, \ddot{\sigma}$ on a message M^* regarding to public parameters $pk^*_S, pk_{\dot{V}}, pk_{\ddot{V}}$. Let σ be a signature produced by *Sign* queries on input M^*, pk^*_S. We say that $\mathcal{A}$ wins the above game if:

1. $(Accept \leftarrow \Sigma.DVerify(M^*, \dot{\sigma}, pk_{\dot{V}}, sk_{\dot{V}}, pk^*_S))$
   $\bigwedge$ $(Accept \leftarrow \Sigma.DVerify(M^*, \ddot{\sigma}, pk_{\ddot{V}}, sk_{\ddot{V}}, pk^*_S))$
   $\bigwedge$ $\sigma \not\leftarrow \Sigma.Open(\dot{\sigma}, \ddot{\sigma}, pk_{\dot{V}}, pk_{\ddot{V}}, pk^*_S, M^*)$.
2. pk^*_S has never been submitted as input of a query for a secret key corresponding to signer public parameters to Key queries.
3. $\mathcal{A}$ could make just one request for a designated verifier signature on input $M^*, pk^*_S, pk_{\dot{V}}$ or on input $M^*, pk^*_S, pk_{\ddot{V}}$ to *Delegate* queries or *DSimulate* queries, respectively.

Let $Succ^{CM-CPK-A}_{ESD-OT-UDVS}(.)$ be the success probability function of that $\mathcal{A}^{CM-CPK-A}_{ESD-OT-UDVS}$ wins the above game.

Definition 5. *We say that OT-UDVS scheme is (t,q_H,q_{SKG},q_S,q_V,q_{VKG},q_D, q_{DV}, q_{DS},q_O,q_K,ϵ)-secure existentially single designatable under a chosen message and chosen public key attack if there are no PPT $CM-CPK-A$ adversary $\mathcal{A}^{CM-CPK-A}_{ESD-OT-UDVS}$ such that the success probability $Succ^{CM-CPK-A}_{ESD-OT-UDVS}(k) = \epsilon$ is negligible in k, where $\mathcal{A}^{CM-CPK-A}_{ESD-OT-UDVS}$ runs in time at most t, make at most q_H, q_{SKG}, q_S, q_V, q_{VKG}, q_D, q_{DV}, q_{DS}, q_O, and q_K queries to the random oracle, $SKeyGen$ queries, $Sign$ queries, $Verify$ queries, $VKeyGen$ queries, Delegate queries, $DVerify$ queries, DSimulate queries, Open queries, and Key queries, respectively.*

5 Primitive Tools

5.1 Trapdoor Commitment Scheme

A trapdoor commitment scheme TC comprises of three PPT algorithms which are *Setup*, $Tcom$ and $Topen$.

$Setup(1^{\mathcal{K}})$ is a algorithm that takes a security parameter $\mathcal{K}$ as an input and generates public parameters pk and a trapdoor (secret) key sk.

$Tcom(pk, m, r)$ is a algorithm that takes pk, m, r as inputs and outputs a commitment value T.

$Topen(sk, pk, m, m', r)$ is a algorithm that takes pk, m, r as inputs and reveals r' such that $T = Tcom(pk, m, r) = Tcom(pk, m', r')$.

The detail of the idea in transforming an identification scheme into a trapdoor commitment scheme can be found in [5]. For clarification, we provide the detail of the Schnorr trapdoor commitment scheme transformed from the Schnorr identification scheme as follows.

$Setup:$ On input a security parameter $\mathcal{K}$, $Setup$ randomly selects a prime l such that $l \approx poly(1^{\mathcal{K}})$. After that, a generator $g_l \in \mathbb{G}_l$, where $\mathbb{G}_l$ be a group of prime order l and a number $y \in \mathbb{Z}_l^*$ are randomly selected. Let $\mathsf{param} = (l, g_l)$ denote system parameters and let $Y = g_l^y$ denote a public key. $Setup$, thereafter, outputs public parameters $pk = (\mathsf{param}, Y)$ and a secret trapdoor key $sk = y$.

$Tcom:$ On input public parameters pk and two integers $m, r \in \mathbb{Z}_l^*$, $Tcom$ computes an output $T = g_l^r Y^m$ and returns T.

$Topen:$ On input public parameters pk, a secret key sk and three integers $m', m, r \in \mathbb{Z}_l^*$, $Topen$ responses with r' such that $T = g_l^r Y^m = g_l^{r'} Y^{m'}$.

Considering that a trapdoor commitment scheme is used as a component of our one-time universal designated verifier signature scheme in the next section. Hence, we supply the security of the above trapdoor commitment scheme as follows.

Definition 6. *We say that a trapdoor commitment scheme TC is secure if ,on input pk, it is computationally infeasible to compute* (m, r) *and* (m', r') *such that* $Tcom(pk, m, r) = Tcom(pk, m', r')$ *where* $m \neq m'$. *[5]*

Theorem 1. *The above trapdoor commitment scheme is secure if the discrete logarithm assumption is hold.*

Proof. The proof can be found in [5].

6 OT-UDVS Scheme

6.1 Concrete Scheme

Let $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ be a random one-way function that maps any string to group $\mathbb{G}_1$ and let $h : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ be a collision-resistant hash function. We denote by G_1, G_2 groups of prime order p. Assume that there exists an efficient computationally bilinear mapping function $\hat{e}$ map $\mathbb{G}_1$ to $\mathbb{G}_2$. The above mapping function is defined as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The scheme is as follows.

Σ.SKeyGen **:** On input a security parameter $\mathcal{K}$, a signer S randomly chooses a prime $p \approx poly(1^{\mathcal{K}})$. Select a random generator $g \in \mathbb{G}_1$. We denote by $\mathsf{param} = (p, \hat{e}, g, H, h)$ the system parameters. Now, a private key and public parameters are generated as follows. Select a random integer $x \in \mathbb{Z}_p$. Let $X = g^x$ denote a public key. Therefore, $SKeyGen$ returns $sk_S = x$ as a private key of signer and $pk_S = (\mathsf{param}, X)$ as public parameters of signer.

Σ.VKeyGen **:** On input a security parameter $\mathcal{K}$, a verifier V runs a trapdoor commitment scheme's setup function $Setup(1^{\mathcal{K}})$ to obtain $l, g_l, Y = g_l^y, sk_V = y$. Let $\bar{h} : \{0,1\}^* \rightarrow \mathbb{Z}_l^*$ be a collision-resistant hash function selected by V. V then publishes $pk_V = (\mathsf{param} = (l, g_l, \bar{h}), Y)$ as public parameters of verifier and keeps sk_V as a private key of verifier.

Σ.Sign **:** Given sk_S, pk_S and a message M, S computes a signature σ on message M as follows.

$$\begin{aligned} r_1 &\overset{\$}{\leftarrow} \mathbb{Z}_p, \\ \sigma_1 &= g^{r_1}, \\ M' &= M||\sigma_1||pk_S, \\ \sigma_2 &= H(M')^x, \\ \sigma_3 &= H(M')^{r_1}. \end{aligned}$$

The signature on message M is $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.

Σ.Verify **:** Given pk_S, σ and a message M, a signature holder SH first computes $M' = M||\sigma_1||pk_S$ and then checks whether

$$\hat{e}(\sigma_2, g) \overset{?}{=} \hat{e}(H(M'), X) \bigwedge$$
$$\hat{e}(\sigma_3, g) \overset{?}{=} \hat{e}(H(M'), \sigma_1)$$

holds or not. If not, then output `reject`. Otherwise, output `accept`.

Delegate **:** Choose a random integer $r_2 \in \mathbb{Z}_p$. Given pk_V, σ and a message M, SH computes a designate verifier signature $\hat{\sigma}$ on message M as follows.

$$\begin{aligned} M' &= M||\sigma_1||pk_S, \\ T_V &= h(g_l^{r_2} Y^{\bar{h}(M')}), \\ h_V &= h(pk_S||pk_V||M||T_V), \\ R' &= \sigma_2 \cdot \sigma_3^{h_V}, \\ \hat{\sigma}_1 &= \sigma_1, \\ \hat{\sigma}_2 &= \sigma_2^{T_V} \cdot R'. \end{aligned}$$

Output the designated verifier signature on message M as $\hat{\sigma} = (\hat{\sigma}_1, \hat{\sigma}_2, r_2)$.

DVerify **:** Given pk_S, pk_V, sk_V, $\hat{\sigma}$ and a message M, the designated verifier V first computes $M' = M||\hat{\sigma}_1||pk_S$, $T_V = h(g_l^{r_2} Y^{\bar{h}(M')})$, $h_V = h(pk_S||pk_V||M||T_V)$, and $R = X \cdot \hat{\sigma}_1^{h_V}$. Then V checks whether

$$\hat{e}(\hat{\sigma}_2, g) \overset{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), R)$$

holds or not. If not, then output `reject`. Output `accept`, otherwise.

DSimulate **:** On input sk_V, pk_V, pk_S and a message M, V first randomly chooses a generater $K \in \mathbb{G}_1$ and integers $k, \bar{r}_2 \in \mathbb{Z}_p$. V then computes as follows.

$$\begin{aligned}
\bar{M} &= M||K||pk_S,\\
T_V &= h(g_l^{\bar{r}_2} Y^{\bar{h}(\bar{M})}),\\
h_V &= h(pk_S||pk_V||M||T_V),\\
\hat{\sigma}_1 &= (g^k \cdot X^{-T_V} \cdot X^{-1})^{\frac{1}{h_V}},\\
M' &= M||\hat{\sigma}_1||pk_S,\\
r_2 &= \bar{r}_2 + y \cdot \bar{h}(\bar{M}) - y \cdot \bar{h}(M'),\\
\hat{\sigma}_2 &= H(M')^k.
\end{aligned}$$

A designated verifier signature on message M is $\hat{\sigma} = (\hat{\sigma}_1, \hat{\sigma}_2, r_2)$.

Open **:** On input $pk_{\dot{V}}$, $pk_{\ddot{V}}$, pk_S and two valid designated verifier signatures where the first signature $\dot{\sigma} = (\dot{\sigma}_1, \dot{\sigma}_2, \dot{r}_2)$ is designated to a verifier $\dot{V}$ and the other signature $\ddot{\sigma} = (\ddot{\sigma}_1, \ddot{\sigma}_2, \ddot{r}_2)$ is designated to another verifier $\ddot{V}$, *Open* computes the necessary parameters as follows.

$$\begin{aligned}
M' &= M||\dot{\sigma}_1||pk_S = M||\ddot{\sigma}_1||pk_S,\\
T_{\dot{V}} &= h(\dot{g}_{\dot{l}}^{\dot{r}_2} \dot{Y}^{\dot{h}(M')}),\\
h_{\dot{V}} &= h(pk_S||pk_{\dot{V}}||M||T_{\dot{V}}),\\
T_{\ddot{V}} &= h(\ddot{g}_{\ddot{l}}^{\ddot{r}_2} \ddot{Y}^{\ddot{h}(M')}),\\
h_{\ddot{V}} &= h(pk_S||pk_{\ddot{V}}||M||T_{\ddot{V}}).
\end{aligned}$$

We denote by $(\dot{g}_{\dot{l}}, \dot{l}, \dot{Y}, \dot{h})$ and $(\ddot{g}_{\ddot{l}}, \ddot{l}, \ddot{Y}, \ddot{h})$ as the public parameters of $\dot{V}$ and $\ddot{V}$, respectively. Note that $\dot{\sigma}_1 = \ddot{\sigma}_1$ always holds, if $\dot{\sigma}$ and $\ddot{\sigma}$ are generated from the same signature. Therefore, for two (or more) simulated designated verifier signatures or a pair of both simulated designated verifier signature and valid designated verifier signature, a designated verifier signature is highly unlikely to share the first part of signature with other designated verifier signatures. From the Lagrange interpolating polynomial, we then calculate the Lagrange coefficient from $h_{\dot{V}}$ and $h_{\ddot{V}}$ as follows.

$$\begin{aligned}
c_{\dot{V}} &= \frac{-h_{\ddot{V}}}{h_{\dot{V}} - h_{\ddot{V}}},\\
c_{\ddot{V}} &= \frac{-h_{\dot{V}}}{h_{\ddot{V}} - h_{\dot{V}}}.
\end{aligned}$$

Now, *Open* computes a signature from two valid UDVS signatures as follows.

$$\begin{aligned}
\sigma_1 &= \dot{\sigma}_1 = \ddot{\sigma}_1 = g^{r_1},\\
\sigma_2 &= (\dot{\sigma}_2^{c_{\dot{V}}} \cdot \ddot{\sigma}_2^{c_{\ddot{V}}})^{\frac{1}{T_{\dot{V}} \cdot c_{\dot{V}} + T_{\ddot{V}} \cdot c_{\ddot{V}} + 1}} = H(M')^x,\\
\sigma_3 &= \dot{\sigma}_2 \cdot \sigma_2^{T_{\dot{V}}} = \ddot{\sigma}_2 \cdot \sigma_2^{T_{\ddot{V}}} = H(M')^{r_1}.
\end{aligned}$$

Hence, a signature on M is $\sigma = (\sigma_1, \sigma_2, \sigma_3)$. Therefore, *Open* outputs σ.

7 Security Proof

7.1 Completeness

Completeness of a Signature and a Designated Verifier Signature: They are straightforward, and hence, it will be omitted.

Completeness of a Simulated Signature: Given public parameters of signer pk_S, public parameters of designated verifier pk_V, a secret key of designated verifier sk_V, a message M and a designate verifier signature $\hat{\sigma}$, First compute $M' = M||\hat{\sigma}_1||pk_S$, $T_V = h(g_l^{r_2} Y^{\bar{h}(M')})$, $h_V = h(pk_S||pk_V||M||T_V)$, and $R = X \cdot \hat{\sigma}_1^{h_V}$. Then check

$$\begin{aligned}
\hat{e}(\hat{\sigma}_2, g) &\stackrel{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), R) \\
\hat{e}(H(M')^k, g) &\stackrel{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), X \cdot \hat{\sigma}_1^{h_V}) \\
\hat{e}(H(M')^k, g) &\stackrel{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), X \cdot ((g^k \cdot X^{-T_V} \cdot X^{-1})^{\frac{1}{h_V}})^{h_V}) \\
\hat{e}(H(M')^k, g) &\stackrel{?}{=} \hat{e}(H(M'), X^{T_V})\hat{e}(H(M'), g^k \cdot X^{-T_V}) \\
\hat{e}(H(M')^k, g) &\stackrel{?}{=} \hat{e}(H(M'), g^k).
\end{aligned}$$

Therefore, the above statements show that the simulated signature indeed holds.

Completeness of an Opened Signature:

$$\begin{aligned}
\hat{e}(H(M'), X) &\stackrel{?}{=} \hat{e}(\sigma_2, g) \\
\hat{e}(H(M'), X) &\stackrel{?}{=} \hat{e}((\dot{\sigma}_2^{c_{\dot{V}}} \cdot \ddot{\sigma}_2^{c_{\ddot{V}}})^{\frac{1}{T_{\dot{V}} \cdot c_{\dot{V}} + T_{\ddot{V}} \cdot c_{\ddot{V}} + 1}}, g) \\
\hat{e}(H(M'), X) &\stackrel{?}{=} \hat{e}((H(M')^{T_{\dot{V}} \cdot x} \cdot H(M')^x \cdot H(M')^{r_1 \cdot h_{\dot{V}}})^{\frac{-h_{\ddot{V}}}{h_{\dot{V}} - h_{\ddot{V}}}} \\
&\quad \cdot (H(M')^{T_{\ddot{V}} \cdot x} \cdot H(M')^x \cdot H(M')^{r_1 \cdot h_{\ddot{V}}})^{\frac{-h_{\dot{V}}}{h_{\ddot{V}} - h_{\dot{V}}}}, g)^{\frac{1}{T_{\dot{V}} \cdot c_{\dot{V}} + T_{\ddot{V}} \cdot c_{\ddot{V}} + 1}} \\
\hat{e}(H(M'), X) &\stackrel{?}{=} \hat{e}((H(M')^{(T_{\dot{V}} \cdot (\frac{-h_{\ddot{V}}}{h_{\dot{V}} - h_{\ddot{V}}}) + T_{\ddot{V}} \cdot (\frac{-h_{\dot{V}}}{h_{\ddot{V}} - h_{\dot{V}}})) \cdot x}) \cdot \\
&\quad (H(M')^{(\frac{-h_{\ddot{V}}}{h_{\dot{V}} - h_{\ddot{V}}} + \frac{-h_{\dot{V}}}{h_{\ddot{V}} - h_{\dot{V}}}) \cdot x}) \cdot \\
&\quad (H(M')^{r_1 \cdot (h_{\dot{V}} \cdot \frac{-h_{\ddot{V}}}{h_{\dot{V}} - h_{\ddot{V}}} + h_{\ddot{V}} \cdot \frac{-h_{\dot{V}}}{h_{\ddot{V}} - h_{\dot{V}}})}), g)^{\frac{1}{T_{\dot{V}} \cdot (\frac{-h_{\ddot{V}}}{h_{\dot{V}} - h_{\ddot{V}}}) + T_{\ddot{V}} \cdot (\frac{-h_{\dot{V}}}{h_{\ddot{V}} - h_{\dot{V}}}) + 1}} \\
\hat{e}(H(M'), X) &\stackrel{?}{=} \hat{e}(H(M')^{x \cdot (T_{\dot{V}} \cdot (\frac{-h_{\ddot{V}}}{h_{\dot{V}} - h_{\ddot{V}}}) + T_{\ddot{V}} \cdot (\frac{-h_{\dot{V}}}{h_{\ddot{V}} - h_{\dot{V}}}))} \\
&\quad \cdot H(M')^x, g)^{\frac{1}{T_{\dot{V}} \cdot (\frac{-h_{\ddot{V}}}{h_{\dot{V}} - h_{\ddot{V}}}) + T_{\ddot{V}} \cdot (\frac{-h_{\dot{V}}}{h_{\ddot{V}} - h_{\dot{V}}}) + 1}} \\
\hat{e}(H(M'), g^x) &\stackrel{?}{=} \hat{e}(H(M')^x, g).
\end{aligned}$$

The above statements show that a opened signature is complete.

7.2 Unforgeability

Theorem 2. *In the random oracle model, our one-time universal designated verifier scheme is existentially unforgeable under an adaptive chosen message and chosen public key attack if the CDH assumption holds.*

Due to the page limitation, please find the proof for Theorem 2 in the full version of this paper [14].

7.3 Non-transferability Privacy

Theorem 3. *In the random oracle model, the purposed one-time universal designated verifier scheme is existentially non-transferable privacy against adaptively chosen message and chosen public key distinguisher* $\mathcal{A}^{CM-CPK-A}_{ENT-OT-UDVS}$.

Due to the page limitation, please find the proof for Theorem 3 in the full version of this paper [14].

7.4 Single Designatability

Theorem 4. *Our one-time universal designated verifier scheme is existentially single designatable under an adaptive chosen message and chosen public key attack if the hash function is collision resistant.*

Proof. (sketch) Let assume that the hash function h of our OT-UDVS scheme is a collision resistent hash function. We denote by $\mathcal{A}$ a forger algorithm and let $\mathcal{F}$ denote an adversary searching for a collision message-pair for hash function h through $\mathcal{A}$. Due to the completeness of an opened signature, the only designated verifier signatures pair $(\dot{\sigma}, \ddot{\sigma}\ )$ that can open is when $\dot{\sigma_1} = \ddot{\sigma_2}$, $\dot{\sigma_2} = \ddot{\sigma_2}$ and $\dot{r}_2 \neq \ddot{r}_2$ or $pk_{\dot{V}} \neq pk_{\ddot{V}}$. The collision of hash function h will occur if such an event happens.

From the above statement, we construct the simulation as follows. First, since $\mathcal{F}$ can arbitrarily generate a public-secret key pair for $\mathcal{A}$, $\mathcal{F}$ constructs straightforwardly queries as described in Section 6.1. $\mathcal{A}$ is given access to those queries. At the end of the above queries, we assume that $\mathcal{A}$ outputs two designated verifier signatures $\dot{\sigma}, \ddot{\sigma}$ on a message M^* regrading to public parameters $pk^*_S, pk_{\dot{V}}, pk_{\ddot{V}}$. $\mathcal{F}$ pronounces that $\mathcal{A}$ wins the game if both signatures are accepted by $DVerify$ queries, and a secret key corresponding to pk^*_S has never been revealed by Key queries and only one designated verifier signature on message M^* can be queried. $\mathcal{F}$ then computes $T_{\dot{V}} = h(\dot{g}_l^{\dot{r}_2}\dot{Y}^{\dot{h}(M^*||\dot{\sigma_1}||pk^*_S)})$; $T_{\ddot{V}} = h(\ddot{g}_l^{\ddot{r}_2}\ddot{Y}^{\ddot{h}(M^*||\ddot{\sigma_1}||pk^*_S)})$. Next, $\mathcal{F}$ sets $\dot{M} = pk^*_S||pk_{\dot{V}}||M^*||T_{\dot{V}}$; $\ddot{M} = pk^*_S||pk_{\ddot{V}}||M^*||T_{\ddot{V}}$ and computes $h_{\dot{V}} = h_{\ddot{V}} = h(\dot{M}) = h(\ddot{M})$.

We note that both $T_{\dot{V}}$ and $h_{\dot{V}}$ are required to be equal to $T_{\ddot{V}}$ and $h_{\ddot{V}}$, respectively. This is because of $\dot{\sigma_1} = \ddot{\sigma_1}$ and $\dot{\sigma_2} = \ddot{\sigma_2}$ as we mentioned earlier. However, $T_{\dot{V}} = T_{\ddot{V}}$ is easy to achieve since $\mathcal{A}$ possessed the verifier secret keys and it can arbitrarily compute $h(\dot{g}_l^{\dot{r}_2}\dot{Y}^{\dot{h}(M^*||\dot{\sigma_1}||pk^*_S)}) = h(\ddot{g}_l^{\ddot{r}_2}\ddot{Y}^{\ddot{h}(M^*||\ddot{\sigma_1}||pk^*_S)})$. On the other hand, $h_{\dot{V}}$ is hard to make itself equivalent to $h_{\ddot{V}}$ since $\dot{M} \neq \ddot{M}$ in every cases.

Hence, $\mathcal{F}$ outputs $\dot{M}$ and $\ddot{M}$ as messages that lead to a collision of hash value $h(\dot{M}) = h(\ddot{M})$. This completes the proof. □

8 Conclusion

We introduced a one time universal designated verifier signature (OT-UDVS) scheme that allows a signature holder to designate a legitimate signature that he holds *once* to any designated verifier of his choice. Nonetheless, if the signature holder designates this signature for more than once, this will enable any third party to "revoke" the original signature from the designated signatures, and hence, reveal the privacy provided by the original signature that was issued by the signer only to the signature holder. We formally defined the security notion for a OT-UDVS scheme. We also presented a concrete OT-UDVS scheme and provided a security analysis of this scheme including the unforgeability, the single designatability and the non-transferability privacy. An interesting open problem is to extend our primitive to a k-times universal designated verifier signature which constraints a signature holder to generate within only k-designated verifier signatures. This problem seems very interesting in theory but we are also to find a good and practical problem that will require this primitive.

References

1. Huang, X., Mu, Y., Susilo, W., Zhang, F.: Short designated verifier proxy signature from pairings. In: Enokido, T., Yan, L., Xiao, B., Kim, D.Y., Dai, Y.-S., Yang, L.T. (eds.) EUC-WS 2005. LNCS, vol. 3823, pp. 835–844. Springer, Heidelberg (2005)
2. Huang, X., Susilo, W., Mu, Y., Wu, W.: Universal designated verifier signature without delegatability. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 479–498. Springer, Heidelberg (2006)
3. Huang, X., Susilo, W., Mu, Y., Zhang, F.: Restricted universal designated verifier signature. In: Ma, J., Jin, H., Yang, L.T., Tsai, J.J.-P. (eds.) UIC 2006. LNCS, vol. 4159, pp. 874–882. Springer, Heidelberg (2006)
4. 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)
5. Kurosawa, K., Heng, S.-H.: The power of identification schemes. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 364–377. Springer, Heidelberg (2006)
6. Laguillaumie, F., Libert, B., Quisquater, J.-J.: Universal designated verifier signatures without random oracles or non-black box assumptions. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 63–77. Springer, Heidelberg (2006)
7. Laguillaumie, F., Vergnaud, D.: Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 105–119. Springer, Heidelberg (2005)
8. Laguillaumie, F., Vergnaud, D.: Multi-designated verifiers signatures: anonymity without encryption. Inf. Process. Lett. 102(2-3), 127–132 (2007)
9. Laguillaumie, F., Vergnaud, D.: On the soundness of restricted universal designated verifier signatures and dedicated signatures. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 175–188. Springer, Heidelberg (2007)
10. Li, Y., Lipmaa, H., Pei, D.: On delegatability of four designated verifier signatures. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 61–71. Springer, Heidelberg (2005)

11. Lipmaa, H., Wang, G., Bao, F.: Designated verifier signature schemes: Attacks, new security notions and a new construction. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 459–471. Springer, Heidelberg (2005)
12. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal designated-verifier signatures. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 523–542. Springer, Heidelberg (2003)
13. Susilo, W., Zhang, F., Mu, Y.: Identity-based strong designated verifier signature schemes. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 313–324. Springer, Heidelberg (2004)
14. Thorncharoensri, P., Susilo, W., Mu, Y.: How to balance privacy with authenticity (full version) (2008); Can be obtained from the first author
15. Zhang, R., Furukawa, J., Imai, H.: Short signature and universal designated verifier signature without random oracles. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 483–498. Springer, Heidelberg (2005)

Efficient Vote Validity Check in Homomorphic Electronic Voting

Kun Peng and Feng Bao

Institute for Infocomm Research, Singapore
dr.kun.peng@gmail.com

Abstract. Homomorphic electronic voting is very popular but has an efficiency bottleneck: vote validity check, which limits application of e-voting, especially in large-scale elections. In this paper two efficient vote validity check mechanisms are designed for homomorphic e-voting, both of which are much more efficient than the existing vote validity check procedures. Especially, the second vote validity check mechanism is flexible, honest-verifier zero knowledge and highly efficient. With the technique, efficiency of homomorphic e-voting is greatly improved and can be employed in election applications with more critical requirement on performance.

1 Introduction

Electronic voting is a popular application of cryptographic and network techniques to e-government. An e-voting scheme should satisfy the following properties.

- Correctness: all the valid votes are counted without being tampered with.
- Privacy: no information about any voter's choice in the election is revealed.
- Robustness: any abnormal situation can be detected and solved without revealing any vote.
- Flexibility: various election rules are supported.

Most of the existing e-voting schemes can be classified into two categories: mix net voting and homomorphic voting. Mix net voting [32,14,1,13,29,30,17,38,33, 20,35,41,16] employs a mix network to shuffle the encrypted votes before they are decrypted (opened) so that the opened votes cannot be traced back to the voters. Homomorphic voting schemes include [3,28,40,19,4,21,10,22,26,27,37,15], which exploit homomorphism of certain encryption algorithms (e.g. Paillier encryption [31]) to decrypt the sum of the votes while no separate vote is decrypted. Tallying in homomorphic voting only costs one single decryption operation for each candidate, so is much more efficient than tallying in mix net voting, which includes a costly mix network. However, correctness of homomorphic voting depends on validity of the votes. An invalid vote can compromise correctness of a homomorphic voting scheme, so must be detected and deleted before the tallying phase. Unfortunately, vote validity check is very costly (both for the voters to

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 202–217, 2009.

prove validity of their votes and for a tallier (and other verifiers) to verify validity of the votes) and becomes an efficiency bottleneck in homomorphic e-voting. Some homomorphic e-voting schemes [26, 27, 15] adjust the vote format and the corresponding vote validity check mechanism such that a large number of checks in small ranges are replaced by a smaller number of checks in larger ranges. However, their improvement in efficiency is not obvious. An attempt [7] is made to improve efficiency of vote validity check in homomorphic e-voting. However, it only reduces the cost in computation and communication by one fourth to one half, so is still not efficient enough for large-scale election applications.

An interesting technique called batched bid validity check [36, 39] is inspiring. Some sealed-bid e-auction schemes [23,25,8,24] exploit homomorphism of certain bid sealing functions to determine the winning bid without opening any separate bid. They are called homomorphic auction schemes. Similar to the situation in homomorphic e-voting, correctness of the homomorphic auction schemes depend on validity of the bids and thus the bids must be proved and verified to be valid. It is noticed in [36, 39] that the previous bid validity check mechanisms are inefficient. A new vote validity check mechanism is designed in [36, 39] to improve efficiency of bid validity check. Using batch verification to improve efficiency is not a new idea. The batch zero knowledge proof and verification technique in [36] is an extension of the traditional batch verification techniques [5, 6, 2, 34]. Although it is interesting to adapt the bid validity check mechanism in [36, 39] into a new vote validity check mechanism, this idea is confined by three drawbacks. Firstly, it employs different sealing (encryption) functions and parameter settings and it is unknown whether it can suit the frequently employed Paillier encryption or its distributed version [12] in homomorphic e-voting schemes. Secondly, it only supports one-candidate Yes/No election so has a very limited application range in e-voting. Thirdly, it is still not efficient enough for large-scale election applications.

Two new vote validity check mechanisms are proposed in this paper. Their improvement on efficiency is more advanced than that in [7]. They not only integrate proof of validity of multiple votes like in [7] (using a different method of course) but also integrate the operations within each proof of validity of vote. They are called Protocol 1 and Protocol 2. Both of them can guarantee validity of vote with an overwhelmingly large probability and are much more efficient than the existing vote validity check mechanisms. They are general solutions to homomorphic e-voting schemes and is not limited to special election rules (e.g. approval voting, divisible voting and Borda voting in [15]). Protocol 1 modifies and extends the batched bid validity check in [36, 39] to homomorphic e-voting applications, which employ different primitives and setting. So proof and verification techniques different from those in [36, 39] are developed in Protocol 1 to suit the new application. It greatly improve efficiency of vote validity check in computation in homomorphic e-voting when only one candidate is selected in a vote. More formal security model than that in [36, 39] is used in Protocol 1 to illustrate its privacy. However, Protocol 1 cannot work when more than one candidate is selected in a vote. So Protocol 1 is not flexible and can only be used in a special case. Moreover, Protocol 1 needs six rounds of communication in vote validity check and may be too interactive for some applications. In addition, Protocol 1 is still not efficient enough for large-scale election applications. Protocol 2 is completely novel and does not inherit the idea of [7] or [36,39]. It does not limit the number of selected candidates in a vote, so is more flexible. Moreover, it is more efficient in computation than Protocol 1. In addition, it needs fewer rounds of communication and is more efficient in communication than Protocol 1. Before Protocol 1 and Protocol 2 are presented, a key cryptographic primitive, batched ZK proof and verification of N^{th} residue, is designed and analysed in Section 3. Then the two protocols are proposed in Section 4 and Section 5 respectively, both of which employ batched ZK proof and verification of N^{th} residue.

Although both Protocols employ Paillier encryption, application of the new vote validity check technique is not limited to votes encrypted with Paillier encryption. Although Paillier encryption is the most popular choice in homomorphic e-voting, other additive homomorphic encryptions (e.g. variants of ElGamal encryption) can also be used in homomorphic e-voting. The new vote validity check technique can be slightly adjusted to fit those encryption algorithms. Due to space limit, application to those encryption algorithms is not described in this paper and left to interested readers.

2 Preliminary and Background

The following symbols and denotations will be used in this paper.

$\|x\|$	the bit length of integer x
$KN(x)$	knowledge of x
V	a challenger in a zero knowledge proof protocol (an independent third party, cooperating multiparties or a random oracle, which may be implemented through a hash function)
$P(E)$	the probability that event E happens
$x \in_R S$	an integer x randomly chosen from a set S

Usually, homomorphic e-voting schemes employ an additive homomorphic encryption algorithm with a distributed decryption function. An encryption algorithm with decryption function $D()$ is additive homomorphic if $D(c_1) + D(c_2) = D(c_1c_2)$ for any ciphertexts c_1 and c_2. A typical additive homomorphic encryption algorithm with a distributed decryption function is distributed Paillier encryption proposed by Fouque *et al* [12], which is recalled as follows.

1. Key generation:
   $N = pq$, $p = 2\acute{p} + 1$ and $q = 2\acute{q} + 1$ where p and q are primes and $gcd(N, \varphi(N)) = 1$. Integers a, b are randomly chosen from Z_N^* and $g = (1+N)^a + b^N \bmod N^2$. The private key is $\beta\acute{p}\acute{q}$ where β is randomly chosen from Z_N^*. The public key consists of N, g and $\theta = a\beta\acute{p}\acute{q}$. $P_1, P_2, \ldots, P_m$ are the private key holders. Let $F(x) = \sum_{k=0}^{t-1} f_k x^k$ where $f_0 = \beta\acute{p}\acute{q}$ and

f_1, f_2, …, f_{t-1} are random integers in $Z_{\acute{p}\acute{q}}$. The share $d_j = F(j) \bmod \acute{p}\acute{q}N$ is distributed to P_j for $j = 1, 2, \ldots, m$. G is the cyclic subgroup containing all the quadratic residues in $Z^*_{N^2}$. Random integer v is a generator of G and $v_j = v^{\varDelta d_j} \bmod N^2$ for $j = 1, 2, \ldots, m$ where $\varDelta = m!$. v and v_j for $j = 1, 2, \ldots, m$ are published.

2. Encryption:
   A message $s \in Z_N$ is encrypted to $c = g^s r^N \bmod N^2$ where r is randomly chosen from Z^*_N.
3. Partial decryptions of ciphertext c:
   For $j = 1, 2, \ldots, m$, P_j provides his part of decryption $s_j = c^{2\triangle d_j}$ and proves $\log_{c^{4\triangle}} s_j^2 = \log_{v^\triangle} v_j$.
4. Combination of partial decryptions:
   The final decryption result can be recovered as $s = L(\prod_{j \in S} s_j^{2u_i}) \times \frac{1}{4\triangle^2\theta}$ where set S contains the indices of t correct partial decryptions and $u_j = \varDelta \prod_{1 \le j' \le n, j' \ne j} \frac{j'}{j'-j}$.

The complete decryption function in the distributed Paillier encryption including the partial decryptions and the combination is denoted as $D()$. An integer y in Z_{N^2} is an N^{th} residue if there exist an integer x such that $x^N = y \bmod N^2$. With this encryption algorithm to seal the votes, most of the existing homomorphic e-voting schemes can be abstracted into the following protocol.

1. Suppose there are n voters and each voter has to choose μ parties from w candidates in his vote. It is required that $w < q$, which is satisfied in any practical election.
2. Each voter V_i chooses his voting vector $(m_{i,1}, m_{i,2}, \ldots, m_{i,w})$ where $m_{i,l} = 0$ or $m_{i,l} = 1$ for $l = 1, 2, \ldots, w$. A rule is followed: $m_{i,l} = 1$ iff V_i chooses the l^{th} candidate.
3. The distributed Paillier encryption recalled above is employed to encrypt the votes where the private key is shared among talliers $A_1, A_2, \ldots, A_m$. Each vote $(m_{i,1}, m_{i,2}, \ldots, m_{i,w})$ is encrypted into $(c_{i,1}, c_{i,2}, \ldots, c_{i,w})$ where $c_{i,l} = g^{m_{i,l}} s_{i,l}^N \bmod N^2$ and $s_{i,l}$ is randomly chosen from Z^*_N for $l = 1, 2, \ldots, w$.
4. As $w < q$, each V_i illustrates validity of his vote through proof of
   $$KN\ [\ c_{i,l}^{1/N}\ ]\ \vee\ KN\ [\ (c_{i,l}/g)^{1/N}\ ] \text{ for } l = 1, 2, \ldots, w \tag{1}$$
   and
   $$KN\ [\ ((\prod_{l=1}^{w} c_{i,l})/g^{\mu})^{1/N}\ ] \tag{2}$$
   (1) is implemented by running the proof protocol Figure 1 (which is a combination of ZK proof of knowledge of root [18] and ZK proof of partial knowledge [9]) for $l = 1, 2, \ldots, w$. (2) is a proof of knowledge of N^{th} root and can be easily and efficiently implemented using ZK proof of knowledge of root in [18].
5. t honest talliers among $A_1, A_2, \ldots, A_m$ cooperate to decrypt $\prod_{i=1}^{n} c_{i,l}$ for $l = 1, 2, \ldots, w$ into the number of votes for the l^{th} candidate. After the decryption, each A_j proves that his partial decryption is correct.

Correctness and soundness of ZK proof of knowledge of root [18] and ZK proof of partial knowledge [9] guarantee that with a overwhelmingly large probability a voter's vote is valid iff his proof of (1) and (2) can pass public verification. Honest verifier zero knowledge property of ZK proof of knowledge of root [18] and ZK proof of partial knowledge [9] guarantee that no vote is revealed in vote validity check. In this prototype, tallying is very efficient and only cost each tallier $3w$ full-length exponentiations. The cost of vote encryption is acceptable while each voter cost $2w$ full-length exponentiations. However, vote validity check is too inefficient. Although implementation of (2) is efficient, repeating the the proof and verification protocol in Figure 1 for w times to prove and verify (1) is inefficient. Vote validity check costs each voter much higher a cost than vote encryption and a verifier at least $O(wn)$ full-length exponentiations. So vote validity check is the efficiency bottleneck of homomorphic e-voting.

1. V_i publishes $a_{m_{i,l}} = r^N \bmod N^2$

$$a_{1-m_{i,l}} = u_{1-m_{i,l}}^N / (c_{i,l}/g^{1-m_{i,l}})^{v_{1-m_{i,l}}} \bmod N^2$$

   where $r \in Z_N^*$, $v_{1-m_{i,l}} \in Z_N$, and $u_{1-m_{i,l}} \in Z_N^*$ are randomly chosen.
2. V publishes an integer v randomly chosen from Z_N.
3. V_i publishes u_0, u_1, v_0 and v_1 where

$$u_{m_{i,l}} = r s_{i,l}^{v_{m_{i,l}}} \bmod N^2$$
$$v_{m_{i,l}} = v - v_{1-m_{i,l}} \bmod Z_N$$

Public verification:

$$u_0^N = a_0 c_{i,l}^{v_0} \bmod N^2$$
$$u_1^N = a_1 (c_{i,l}/g)^{v_1} \bmod N^2$$
$$v = v_0 + v_1 \bmod Z_N$$

Fig. 1. Repeated w times to implement proof and verification of (1)

3 Batched ZK Proof and Verification of N^{th} Residue

A novel batched zero knowledge proof and verification technique is proposed in this section as a cryptographic primitive to be used in optimisation of vote validity check later in this paper. In batched zero knowledge proof and verification, a security parameter L is used, which is smaller than p and q. L will be used in the rest of this paper. Theorem 1 illustrates that multiple integers can be proved and verified to be N^{th} residues in a batch.

Theorem 1. *Suppose $y_1, y_2, \ldots, y_n$ are in Z_{N^2}. If $\prod_{i=1}^n y_i^{t_i}$ is an N^{th} residue with a probability larger than 2^{-L} where integers $t_1, t_2, \ldots, t_n$ are randomly chosen from $\{0, 1, \ldots, 2^L - 1\}$, then $y_1, y_2, \ldots, y_n$ are N^{th} residues.*

Proof: $\prod_{i=1}^{n} y_i^{t_i}$ is an N^{th} residue with a probability larger than 2^{-L} implies that for any given integer v in $\{1, 2, \ldots, n\}$ there must exist integers $t_1, t_2, \ldots, t_n, t'_v \in \{0, 1, \ldots, 2^L - 1\}$, x and x' such that

$$\prod_{i=1}^{n} y_i^{t_i} = x^N \bmod N^2 \tag{3}$$

$$(\prod_{i=1}^{v-1} y_i^{t_i}) y_v^{t'_v} \prod_{i=v+1}^{n} y_i^{t_i} = x'^N \bmod N^2 \tag{4}$$

Otherwise, for any combination of $t_1, t_2, \ldots, t_{v-1}, t_{v+1}, \ldots, t_n$, all in $\{0, 1, \ldots, 2^L - 1\}$, there is at most one t_v in $\{0, 1, \ldots, 2^L - 1\}$ such that $\prod_{i=1}^{n} y_i^{t_i}$ is an N^{th} residue. This implies that among the 2^{nL} possible choices of $t_1, t_2, \ldots, t_n$ (combination of $2^{(n-1)L}$ possible choices of $t_1, t_2, \ldots, t_{v-1}, t_{v+1}, \ldots, t_n$ and 2^L possible choices of t_v) there is at most $2^{(n-1)L}$ choices to construct N^{th} residue $\prod_{i=1}^{n} y_i^{t_i}$, which is a contradiction to the assumption that $\prod_{i=1}^{n} y_i^{t_i}$ is an N^{th} residue with a probability larger than 2^{-L}. Thus, a contradiction is found when (3) and (4) are not satisfied, which implies they must be satisfied.

(3) and (4) imply that

$$y_v^{t_v - t'_v} = (x/x')^N \bmod N^2$$

According to Euclidean algorithm there exist integers α and β to satisfy $\beta(t_v - t'_v) = \alpha N + GCD(N, t_v - t'_v)$. $GCD(N, t_v - t'_v) = 1$ as $t_v < 2^L$, $t'_v < 2^L$ and $2^L < min(p, q)$. So $y_v^{\beta(t_v - t'_v)} = y_v^{\alpha N} y_v$. Namely,

$$y_v = y_v^{\beta(t_v - t'_v)} / y_v^{\alpha N} = (y_v^{t_v - t'_v})^{\beta} / y_v^{\alpha N} = (x/x')^{N\beta} / (y_v^{\alpha})^N = ((x/x')^{\beta} / y_v^{\alpha})^N \bmod N^2$$

So y_v is an N^{th} residue. Therefore, $y_1, y_2, \ldots, y_n$ are N^{th} residues as v can be any integer in $\{1, 2, \ldots, n\}$. □

According to Theorem 1, proof and verification that y_i for $i = 1, 2, \ldots, n$ are N^{th} residues can be batched to proof and verification that $(\prod_{i=1}^{n} y_i^{t_i})$ is an N^{th} residue when $t_1, t_2, \ldots, t_n$ are randomly chosen. If any integer in $\{y_1, y_2, \ldots, y_n\}$ is not an N^{th} residue, $\prod_{i=1}^{n} y_i^{t_i}$ is an N^{th} residue with only negligible probability.

4 Efficiency Optimisation: Protocol 1

When only one candidate is selected in each vote (or more precisely $\mu = 1$), (1) in the prototype can be implemented using batch proof-verification techniques. For simplicity suppose $m_{i,\delta} = 1$ and $m_{i,l} = 0$ for $l = 1, 2, \ldots, \delta - 1, \delta + 1, \ldots, w$. (1) is implemented by the proof and verification protocol in Figure 2. The first three steps in Figure 2 can be regarded as an oblivious transfer [11]. The three steps transfer t_δ from V to V_i while δ is not revealed to V and $t_1, t_2, \ldots, t_{\delta-1}, t_{\delta+1}, \ldots, t_w$ are not revealed to V_i. V_i needs t_δ to calculate its commitment in Step 4. The last three steps implement a batched proof of knowledge of N^{th} root. The protocol in Figure 2 proves

$$c_{i,l} \text{ is an } N^{th} \text{ residue for } w - 1 \text{ instances of } l \text{ where } l \in \{1, 2, \ldots, w\}$$

without revealing the vote. This proof and verification is much more efficient than repeating the proof and verification in Figure 1 w times, and is effective in vote validity check when $\mu = 1$. In addition to

$$KN\ [\ ((\prod_{l=1}^{w} c_{i,l})/g)^{1/N}\ ] \tag{5}$$

the proof in Figure 2 guarantees that

- $D(c_{i,l}) = 0$ for $w-1$ cases of $l \in \{1, 2, \ldots, w\}$;
- $D(c_{i,l}) = 1$ for 1 case of $l \in \{1, 2, \ldots, w\}$.

Correctness of the optimised vote validity check in Figure 2 is straightforward: when V_i is honest his proof can successfully pass the verification. Other

1. $V \rightarrow V_i$:

$$N'$$
$$s_l = t_l - (\textstyle\prod_{k=1}^{|w|} e_{k,b_{l,k}}) \bmod N' \text{ for } l = 1, 2, \ldots w$$
$$c_k \in_R Z_{N'} \text{ for } k = 1, 2, \ldots |w|$$
$$e = d^{-1} \bmod \phi(N')$$

where $N' = p'q'$, $N' < N$, $d \in_R Z^*_{N'}$, $t_l \in_R \{0, 1, \ldots, 2^L - 1\}$ for $l = 1, 2, \ldots, w$, $e_{k,j} \in_R Z_{N'}$ for $k = 1, 2, \ldots, |w|$ and $j = 0, 1$, $b_{l,k}$ denotes the k^{th} bit of l and p', q' are large primes.

2. $V_i \rightarrow V$: $\quad y_{k,j}$ for $k = 1, 2, \ldots, |w|$ and $j = 0, 1$
where

$$y_{k,b_{\delta,k}} = \tau_k^e \bmod N' \text{ for } k = 1, 2, \ldots, |w|$$
$$y_{k,1\oplus b_{\delta,k}} = y_{k,b_{\delta,k}} c_k \bmod N' \text{ if } b_{\delta,k} = 0 \text{ for } k = 1, 2, \ldots, |w|$$
$$y_{k,1\oplus b_{\delta,k}} = y_{k,b_{\delta,k}} / c_k \bmod N' \text{ if } b_{\delta,i} = 1 \text{ for } k = 1, 2, \ldots, |w|$$

$\tau_k \in_R N'$ for $k = 1, 2, \ldots |w|$ and $\oplus$ stands for XOR.

3. $V \rightarrow V_i$: $\quad E_{k,j} = y_{k,j}^d e_{k,j} \bmod N'$ for $k = 1, 2, \ldots, |w|$ and $j = 0, 1$

4. $V_i \rightarrow V$: $\quad a = r^N g^{s_\delta + (\prod_{k=1}^{|w|} E_{k,b_{\delta,k}})/\prod_{k=1}^{|w|} \tau_k} \bmod N^2$
where r is randomly chosen from Z^*_N.

5. $V \rightarrow V_i$: $\quad t_1, t_2, \ldots, t_w$.

6. $V_i \rightarrow V$: $\quad z = \prod_{l=1}^{w} s_{i,l}^{t_l} / r \bmod N^2$

Public verification:

$$N' < N$$
$$y_{k,1} = c_k y_{k,0} \bmod N' \text{ for } k = 1, 2, \ldots, |w|$$
$$\textstyle\prod_{l=1}^{w} c_{i,l}^{t_l} = a z^N \bmod N^2$$

Fig. 2. Proof and verification in Protocol 1

more complicated security properties of this protocol like soundness and zero knowledge are proved in the following theorems.

Theorem 2. *The proof protocol in Figure 2 is honest-verifier zero knowledge.*

Proof: The proof transcript in Figure 2 includes integers N'; s_l for $l = 1, 2, \ldots, w$; c_k for $k = 1, 2, \ldots, |w|$; e; $y_{k,j}$ for $k = 1, 2, \ldots, |w|$ and $j = 0, 1$; $E_{k,j}$ for $k = 1, 2, \ldots, |w|$ and $j = 0, 1$; a; $t_1, t_2, \ldots, t_w$; z. Any party without any secret knowledge can simulate this proof transcript as follows.

1. Randomly choose large primes p' and q' such that their product is smaller than N. Calculate and publish $N' = p'q'$.
2. Randomly choose $e_{k,j} \in_R Z_{N'}$ for $k = 1, 2, \ldots, |w|$ and $j = 0, 1$ and $t_l \in_R \{0, 1, \ldots, 2^L - 1\}$ for $l = 1, 2, \ldots, w$. Publish t_l and $s_l = t_l - (\prod_{k=1}^{|w|} e_{k,b_{l,k}}) \bmod N'$ for $l = 1, 2, \ldots, w$.
3. Randomly choose $y_{k,0}$ and c_k from $Z_{N'}$ for $k = 1, 2, \ldots, |w|$. Publish $y_{k,0}$, c_k and $y_{k,1} = c_k y_{k,0} \bmod N'$ for $k = 1, 2, \ldots, |w|$.
4. Randomly choose e from $Z^*_{N'}$. Calculate $d = e^{-1} \bmod \phi(N')$. Publish e and $E_{k,j} = y^d_{k,j} e_{k,j} \bmod N'$ for $k = 1, 2, \ldots, |w|$ and $j = 0, 1$.
5. Randomly choose z from Z^*_N. Publish z and $a = (\prod_{l=1}^{w} c_{i,l}^{t_l})/z^N \bmod N^2$.

So if the verifier is honest and chooses the challenges randomly in the proof protocol in Figure 2, both the simulated transcript and the proof transcript in Figure 2 have the same distribution (as detailed in Table 1) and satisfy the following conditions.

- $N' < N$.
- $y_{k,1} = c_k y_{k,0} \bmod N'$ for $k = 1, 2, \ldots, |w|$.
- $\prod_{l=1}^{w} c_{i,l}^{t_l} = az^N \bmod N^2$.
- $\exists$ d uniformly distributed in $Z^*_{N'}$ and $e_{k,j}$ uniformly distributed in $Z_{N'}$ for $k = 1, 2, \ldots, |w|$ and $j = 0, 1$ such that $s_l = t_l - (\prod_{k=1}^{|w|} e_{k,b_{l,k}}) \bmod N'$ for $l = 1, 2, \ldots w$ and $E_{k,j} = y^d_{k,j} e_{k,j} \bmod N'$ for $k = 1, 2, \ldots, |w|$ and $j = 0, 1$.

Therefore, the two transcripts have the same distribution and are indistinguishable if the verifier is honest. □

Theorem 3. *The proof protocol in Figure 2 is sound. More precisely, if V' has only polynomial computation capability and his proof passes the verification in Figure 2 with a probability no smaller than $2^{-wL} + 2^{-L}$, the proof in Figure 2 together with (5) guarantees that $(1, 0, 0 \ldots, 0)$ is permuted and encrypted in $c_{i,1}, c_{i,2}, \ldots, c_{i,w}$.*

To prove Theorem 3, Lemma 1 is proved first.

Lemma 1. *In the first three steps of the protocol in Figure 2, a polynomial V_i can calculate at most one of the w integers $t_1, t_2, \ldots, t_w$.*

Table 1. Distribution of the variables in both transcripts

Variable	Distribution
N'	uniformly in $\{x \mid x = p'q',\ p'$ and q' are large primes, $p'q' < N\}$
e	uniformly in $Z_{N'}^*$
t_l	uniformly in $\{0, 1, \ldots, 2^L - 1\}$
s_l	uniformly in $Z_{N'}$
c_k	uniformly in $Z_{N'}$
$y_{k,0}$	uniformly in $Z_{N'}$
$y_{k,1}$	uniformly in $Z_{N'}$
$E_{k,0}$	uniformly in $Z_{N'}$
$E_{k,1}$	uniformly in $Z_{N'}$
z	uniformly in Z_N^*
a	uniformly in $\{x \mid D(x) = t_\delta\}$

Proof: In the protocol in Figure 2

$$E_{k,j} = y_{k,j}^d e_{k,j} \bmod N' \text{ for } k = 1, 2, \ldots, |w| \text{ and } j = 0, 1 \tag{6}$$

$$s_l = t_l - (\textstyle\prod_{k=1}^{|w|} e_{k,b_{l,k}}) \bmod N' \text{ for } l = 1, 2, \ldots w \tag{7}$$

(6) implies

$$e_{k,j} = E_{k,j}/y_{k,j}^d \bmod N' \text{ for } k = 1, 2, \ldots, |w| \text{ and } j = 0, 1 \tag{8}$$

(7) and (8) imply that

$$t_l = s_l + (\textstyle\prod_{k=1}^{|w|} E_{k,b_{l,k}})/\prod_{k=1}^{|w|} y_{k,b_{l,k}}^d \bmod N' \text{ for } l = 1, 2, \ldots w$$

If in the first three steps of the protocol in Figure 2, a polynomial V_i can calculate more than one of the w integers $t_1, t_2, \ldots, t_w$, then there exists k' in $\{1, 2, \ldots, |w|\}$, such that the polynomial V_i can calculate both $y_{k',0}^d$ and $y_{k',1}^d$. Public verification of $y_{k,1} = c_k y_{k,0} \bmod N'$ for $k = 1, 2, \ldots, |w|$ guarantees that $y_{k',1} = c_{k'} y_{k',0} \bmod N'$. Thus the polynomial V_i can calculate $c_{k'}^d$. So a contradiction to the RSA assumption (when the factorization of N' is unknown, it is impossible to calculate the e^{th} root in polynomial time) is found. Therefore, a polynomial V_i can calculate at most one of the w integers $t_1, t_2, \ldots, t_w$. □

Proof of Theorem 3

As Lemma 1 guarantees that V_i can calculate at most one of the w integers $t_1, t_2, \ldots, t_w$, suppose V_i cannot calculate $t_1, t_2, \ldots, t_{\rho-1}, t_{\rho+1}, \ldots, t_w$. Namely, when V_i calculates a, it does not know $t_1, t_2, \ldots, t_{\rho-1}, t_{\rho+1}, \ldots, t_w$. As V_i can pass the verification in the protocol in Figure 2 with a probability larger than $2^{-wL} + 2^{-L}$, there must exist two different sets of challenges $t_1, t_2, \ldots, t_w$ and $t'_1, t'_2, \ldots, t'_{\rho-1}, t_\rho, t'_{\rho+1}, \ldots, t'_w$ to the same commitment a, such that V_i can give two responses z and z' to satisfy

$$\textstyle\prod_{l=1}^{w} c_{i,l}^{t_l} = az^N \bmod N^2 \tag{9}$$

$$(\textstyle\prod_{l=1}^{\rho-1} c_{i,l}^{t'_l}) c_{i,\rho}^{t_\rho} \prod_{l=\rho+1}^{w} c_{i,l}^{t'_l} = az'^N \bmod N^2 \tag{10}$$

with a probability larger than 2^{-L}. Otherwise, with commitment a the prover can give correct response to at most one challenge with a probability larger than 2^{-L}. This deduction implies that when a random challenge is raised the probability that the prover can pass the verification is no more than

$$0 \times P(E_0) + p_1 P(E_1) + p_2 P(E_2)$$

where p_1 is larger than 2^{-L}, p_2 is no larger than 2^{-L}, E_0 denotes the event that the prover can give correct response to no challenge with a probability larger than 2^{-L}, E_1 denotes the event that the prover can give correct response to just one challenge with a probability larger than 2^{-L} and that challenge happens to be chosen, E_2 denotes the event that the prover can give correct response to just one challenge with a probability larger than 2^{-L} and that challenge is not chosen. As

$$0 \times P(E_0) + p_1 P(E_1) + p_2 P(E_2) = p_1 2^{-wL} + p_2(1 - 2^{-wL}) < 2^{-wL} + 2^{-L},$$

there is a contradiction to the assumption that the prover can pass the verification in the protocol in Figure 2 with a probability no smaller than $2^{-wL} + 2^{-L}$. Thus, a contradiction is found when (9) and (10) are not satisfied, which implies they must be satisfied.

(9) divided by (10) yields

$$\prod_{1 \le l \le w, l \ne \rho} c_{i,l}^{t_l - t'_l} = (z/z')^N \bmod N^2,$$

which is correct with a probability larger than 2^{-L}. So $(\prod_{1 \le l \le w, l \ne \rho} c_{i,l}^{t_l - t'_l})$ is an N^{th} residue with a probability larger than 2^{-L}.

As the challenges $t_1, t_2, \ldots, t_{\rho-1}, t_{\rho+1}, \ldots, t_w$ and $t'_1, t'_2, \ldots, t'_{\rho-1}, t'_{\rho+1}, \ldots, t'_w$ are randomly chosen, $t_1 - t'_1, t_2 - t'_2, \ldots, t_{\rho-1} - t'_{\rho-1}, t_{\rho+1} - t'_{\rho+1}, \ldots, t_w - t'_w$ are random. So according to Theorem 1, $c_{i,1}, c_{i,2}, \ldots, c_{i,\rho-1}, c_{i,\rho+1}, \ldots, c_{i,w}$ are N^{th} residues. Note that (5) guarantees that $\sum_{l=1}^{w} D(c_{i,l}) = 1$. Therefore, $(1, 0, 0 \ldots, 0)$ is encrypted in $c_{i,1}, c_{i,2}, \ldots, c_{i,w}$ after being permuted. □

This new proof and verification of vote validity costs $O(|w|)$ full-length exponentiations, which is much more efficient than the proof and verification of vote validity in the prototype and in [7]. Although it is required that each voter can only select one candidate in Protocol 1, this requirement is satisfied in some election applications.

5 Advanced Efficiency Optimisation: Protocol 2

Although Protocol 1 improves efficiency of vote validity check in homomorphic e-voting, it has several drawbacks. Firstly, it limits that each voter can only select one candidate and thus has only a limited range of applications. Secondly, its vote validity check requires too many rounds (six rounds) of communication, which is a drawback in many circumstances. Thirdly, vote validity check in Protocol

1 is still not efficient enough in computation. Especially, verification of validity of all the votes is still a costly operation. So any verifier without a powerful computation capability still feels difficult to verify validity of the election. To solve these problems, Protocol 2 is proposed, which employs a more advanced vote validity check mechanism. In Protocol 2, to illustrate validity of his vote V_i has to use the proof protocol in Figure 3 to prove

$$\wedge_{l=1}^{w} \left(D(c_{i,l}) = 0 \vee D(c_{i,l}) = 1\right) \tag{11}$$

where (2) is a proof of knowledge of N^{th} root and can be easily and efficiently implemented using ZK proof of knowledge of root in [18].

1. V publishes random integers $t_{l,0}, t_{l,1} \in_R \{0, 1, \ldots, 2^L - 1\}$ for $l = 1, 2, \ldots, w$.
2. V_i publishes

$$a = r^N \prod_{l=1}^{w} (c_{i,l} g^{m_{i,l}-1})^{t_{l,1-m_{i,l}} v_{l,1-m_{i,l}}} \bmod N^2$$

where $v_{l,1-m_{i,l}} \in \{0, 1, \ldots, 2^L - 1\}$ for $l = 1, 2, \ldots, w$ and $r \in Z_N^*$ are randomly chosen.

3. V publishes random integer $v \in \{0, 1, \ldots, 2^L - 1\}$.
4. V_i publishes $v_{l,0},\ v_{l,1}$ for $l = 1, 2, \ldots, w$ and u where

$$v_{l,m_{i,l}} = v - v_{l,1-m_{i,l}} \bmod 2^L \text{ for } l = 1, 2, \ldots, w$$
$$u = r \prod_{l=1}^{w} s_{i,l}^{t_{l,m_{i,l}} v_{l,m_{i,l}}} \bmod N^2$$

Public verification:

$$u^N = a \prod_{l=1}^{w} c_{i,l}^{t_{l,0} v_{l,0}} (c_{i,l}/g)^{t_{l,1} v_{l,1}} \bmod N^2$$
$$v = v_{l,0} + v_{l,1} \bmod 2^L \text{ for } l = 1, 2, \ldots, w$$

Fig. 3. Proof and verification in Protocol 2

Correctness of Protocol 2 is straightforward. Other more complicated security properties of this protocol like soundness and zero knowledge are proved in the following theorems.

Theorem 4. *The protocol in Figure 3 is honest-verifier zero knowledge.*

Proof: A party without any secret knowledge can generate a proof transcript containing $v_{l,0}$, $v_{l,1}$, $t_{l,0}$, $t_{l,1}$ for $l = 1, 2, \ldots, w$; u; v and a as follows.

1. Randomly choose $v_{l,0} \in \{0, 1, \ldots, 2^L - 1\}$, $t_{l,0} \in \{0, 1, \ldots, 2^L - 1\}$, $t_{l,1} \in \{0, 1, \ldots, 2^L - 1\}$ for $l = 1, 2, \ldots, w$, $v \in \{0, 1, \ldots, 2^L - 1\}$ and $u \in Z_N^*$.
2. Calculate $v_{l,1} = v - v_{l,0} \bmod 2^L$ for $l = 1, 2, \ldots, w$.
3. Calculate $a = u^N / (\prod_{l=1}^{w} c_{i,l}^{t_{l,0} v_{l,0}} (c_{i,l}/g)^{t_{l,1} v_{l,1}}) \bmod N^2$.

If the verifier is honest and randomly chooses the challenges in the proof protocol in Figure 3, in both the simulated transcript and the proof transcript in Figure 3, $v_{l,0}$, $v_{l,1}$, $t_{l,0}$, $t_{l,1}$ for $l = 1, 2, \ldots, w$ and v are uniformly distributed in $\{0, 1, \ldots, 2^L - 1\}$, u is uniformly distributed in Z_N^* and a is uniformly distributed in the ciphertext space of the employed Paillier encryption. Therefore, the two transcripts have the same distribution and are indistinguishable. □

Theorem 5. *The proof protocol in Figure 3 is sound. More precisely, if the challenges are random and the proof passes the verification in the protocol in Figure 3 with a probability no smaller than* 2^{1-L}*, then* $\wedge_{l=1}^{w}(D(c_{i,l}) = 0 \vee D(c_{i,l}) = 1)$ *is true.*

Proof: As the prover can pass the verification in the protocol in Figure 3 with a probability larger than 2^{1-L}, the prover must be able to give two responses ($v_{l,0}$ for $l = 1, 2, \ldots, w$; $v_{l,1}$ for $l = 1, 2, \ldots, w$; u) and ($v'_{l,0}$ for $l = 1, 2, \ldots, w$; $v'_{l,1}$ for $l = 1, 2, \ldots, w$; u') to two different challenges v and v' to the same commitment a and same integers $t_{l,0}$, $t_{l,1}$ for $l = 1, 2, \ldots, w$, such that

$$u^N = a \prod_{l=1}^{w} c_{i,l}^{t_{l,0}v_{l,0}} (c_{i,l}/g)^{t_{l,1}v_{l,1}} \bmod N^2 \tag{12}$$

$$v = v_{l,0} + v_{l,1} \bmod 2^L \text{ for } l = 1, 2, \ldots, w \tag{13}$$

$$u'^N = a \prod_{l=1}^{w} c_{i,l}^{t_{l,0}v'_{l,0}} (c_{i,l}/g)^{t_{l,1}v'_{l,1}} \bmod N^2 \tag{14}$$

$$v' = v'_{l,0} + v'_{l,1} \bmod 2^L \text{ for } l = 1, 2, \ldots, w \tag{15}$$

with a probability larger than 2^{-L}. Otherwise, the prover can give a correct response to at most one challenge with a probability larger than 2^{-L}. This deduction implies when a random challenge is raised the probability that the prover can pass the verification is no more than

$$0 \times P(E_0) + p_1 P(E_1) + p_2 P(E_2)$$

where p_1 is larger than 2^{-L}, p_2 is no larger than 2^{-L}, E_0 denotes the event that the prover can give correct response to no challenge with a probability larger than 2^{-L}, E_1 denotes the event that the prover can give correct response to just one challenge with a probability larger than 2^{-L} and that challenge happens to be chosen, E_2 denotes the event that the prover can give correct response to just one challenge with a probability larger than 2^{-L} and that challenge is not chosen. As

$$0 \times P(E_0) + p_1 P(E_1) + p_2 P(E_2) = p_1 2^{-L} + p_2(1 - 2^{-L}) < 2^{-L} + 2^{-L} = 2^{1-L}$$

there is a contradiction to the assumption that the prover can pass the verification in the protocol in Figure 3 with a probability no smaller than 2^{1-L}. Thus, a contradiction is found when (12), (13), (14) and (15) are not satisfied, which implies that they must be satisfied.

(12) divided by (14) yields

$$(u/u')^N = \prod_{l=1}^{w} c_{i,l}^{t_{l,0}(v_{l,0} - v'_{l,0})} (c_{i,l}/g)^{t_{l,1}(v_{l,1} - v'_{l,1})} \bmod N^2 \tag{16}$$

Subtracting (15) from (13) yields

$$v - v' = (v_{l,0} - v'_{l,0}) + (v_{l,1} - v'_{l,1}) \bmod 2^L \text{ for } l = 1, 2, \ldots, w \tag{17}$$

Equation (16) implies that $\prod_{l=1}^{w} c_{i,l}^{t_{l,0}(v_{l,0}-v'_{l,0})}(c_{i,l}/g)^{t_{l,1}(v_{l,1}-v'_{l,1})}$ is an N^{th} residue with a probability larger than 2^{-L}. Note that $t_{l,0}$ and $t_{l,1}$ are randomly chosen for $l = 1, 2, \ldots, w$. So according to Theorem 1, $c_{i,l}^{v_{l,0}-v'_{l,0}}$ and $(c_{i,l}/g)^{v_{l,1}-v'_{l,1}}$ for $l = 1, 2, \ldots, w$ are N^{th} residues.

As v and v' are different integers in $\{0, 1, \ldots, 2^L - 1\}$ and thus $v \neq v' \bmod 2^L$, Equation (17) implies that for any l in $\{1, 2, \ldots, w\}$, $(v_{l,0} - v'_{l,0}) \bmod 2^L$ and $(v_{l,1} - v'_{l,1}) \bmod 2^L$ cannot be zeros at the same time. As $v_{l,0}$, $v'_{l,0}$, $v_{l,1}$, $v'_{l,1}$ are smaller than 2^L, for any l in $\{1, 2, \ldots, w\}$, $v_{l,0} - v'_{l,0}$ and $v_{l,1} - v'_{l,1}$ cannot be zeros at the same time.

Suppose $v_{l,\pi(l)} - v'_{l,\pi(l)} \neq 0$ and $(c_{i,l}/g^{\pi(l)})^{v_{l,\pi(l)}-v'_{l,\pi(l)}} = x_l^N \bmod N^2$ for $l = 1, 2, \ldots, w$ where $\pi()$ is a function from $\{1, 2, \ldots, w\}$ to $\{0, 1\}$. According to Euclidean algorithm there exist integers α_l and β_l to satisfy $\beta_l(v_{l,\pi(l)} - v'_{l,\pi(l)}) = \alpha_l N + GCD(N, v_{l,\pi(l)} - v'_{l,\pi(l)})$ for $l = 1, 2, \ldots, w$. $GCD(N, v_{l,\pi(l)} - v'_{l,\pi(l)}) = 1$ for $l = 1, 2, \ldots, w$ as $v_{l,\pi(l)}$, $v'_{l,\pi(l)} < 2^L < min(p, q)$. So $(c_{i,l}/g^{\pi(l)})^{\beta_l(v_{l,\pi(l)}-v'_{l,\pi(l)})} = (c_{i,l}/g^{\pi(l)})^{\alpha_l N}(c_{i,l}/g^{\pi(l)}) \bmod N^2$. Namely,

$$\begin{aligned} c_{i,l}/g^{\pi(l)} &= (c_{i,l}/g^{\pi(l)})^{\beta_l(v_{l,\pi(l)}-v'_{l,\pi(l)})}/(c_{i,l}/g^{\pi(l)})^{\alpha_l N} \\ &= ((c_{i,l}/g^{\pi(l)})^{(v_{l,\pi(l)}-v'_{l,\pi(l)})})^{\beta_l}/(c_{i,l}/g^{\pi(l)})^{\alpha_l N} \\ &= (x_l^N)^{\beta_l}/(c_{i,l}/g^{\pi(l)})^{\alpha_l N} \\ &= (x_l^{\beta_l}/(c_{i,l}/g^{\pi(l)})^{\alpha_l})^N \bmod N^2 \text{ for } l = 1, 2, \ldots, w \end{aligned}$$

So, $c_{i,l}/g^{\pi(l)}$, namely $c_{i,l}$ or $c_{i,l}/g$, is an N^{th} residue for $l = 1, 2, \ldots, w$. Therefore, $\wedge_{l=1}^{w}(D(c_{i,l}) = 0 \vee D(c_{i,l}) = 1)$ is true. □

Theorem 5 together with (2) guarantees that in each vote there are μ ones and $w - \mu$ zeros. The proof and verification in Protocol 2 only costs $O(1)$ full-length exponentiations, which is much more efficient than the proof and verification of vote validity check in the existing homomorphic e-voting schemes and Protocol 1. With the advanced optimised vote validity check in Figure 3. Moreover, Protocol 2 does not limit the number of selected candidates in a vote and is flexible. So Protocol 2 is more advanced than Protocol 1.

6 Comparison and Conclusion

A comparison is made between the existing e-voting schemes (mix net voting and homomorphic voting) and homomorphic voting with optimised vote validity check (Protocol 1 and Protocol 2) in this paper. In mix net voting, suppose the talliers cooperate to shuffle the votes, decrypt them and prove validity of their operations. The number of multiplications is counted when computational

Table 2. Comparison of e-voting schemes

Scheme	ZK	Flexible	Cost of a voter		Cost of a tal-	Cost of verification of	
			encryption	vote validity proof	lier in tallying	vote validity	tallying
Mix net voting	Some	Yes	$\geq 1.5K+1$ $= 1537$	unnecessary	$\geq 7nK$ $= 71680000$	unnecessary	$\geq 6tnK$ $= 184320000$
Existing homomorphic voting	Some	Yes	$\geq$ $(1.5K+1)w$ $= 24592$	$\geq w(4.5K+3L+2)$ $+1.5K+1.5L+2$ $= 77098$	$\geq w$ $(4.5K+1)$ $= 73744$	$\geq nw(3K+3L+3)$ $+1.5K+1.5L+2$ $= 525380000$	$\geq tw$ $(3K+3L+2)$ $= 153312$
[7]	Yes	Yes	$(1.5K+1)w$ $= 24592$	$w(3K+3L+2)$ $+1.5K+1.5L+2$ $= 52522$	$w(4.5K+1)$ $= 73744$	$nw(1.5K+3L+3)$ $+1.5(w+1)K+1.5L$ $+2 = 279644576$	$tw\times$ (3K+3L+2) $= 153312$
Protocol 1	No	No	$(1.5K+1)w$ $= 24592$	$w(0.5L+2)+$ $(0.75K+1.5)\vert w\vert$ $+7.5K+3$ $= 11113$	$w(4.5K+1)$ $= 73744$	$nw(0.5L+\vert w\vert+$ $1)+(3K+2)\vert w\vert$ $+4.5K+2$ $= 173060000$	$tw\times$ (3K+3L+2) $= 153312$
Protocol 2	Yes	Yes	$(1.5K+1)w$ $= 24592$	$1.5K+6wL+2w$ $= 5424$	$w(4.5K+1)$ $= 73744$	$n(1.5K+6wL+3w)$ $= 54240000$	$tw(3K+3L$ $+2) = 153312$

cost is analysed. K is the length of a full-length integer (e.g. 1024 bits). L is the length of challenges used in ZK proofs. t is the number of honest talliers needed in tallying. As suggested in [5], $1.5x$ modulo multiplications are needed to calculate a modulo exponentiation with an x-bit exponent and $y+0.5yx$ modulo multiplications are needed to calculate the product of y modulo exponentiations with x-bit exponents. An example is given in Table 2, where $K = 1024$, $L = 40$, $n = 10000$, $t = 3$ and $w = 16$. It is clearly illustrated in Table 2

- Protocol 1 improves efficiency, but is not flexible and is still not efficient enough;
- Protocol 2 is flexible and highly efficient.

In summary, vote validity check in homomorphic e-voting is optimised in this paper. Two new vote validity check procedures, Protocol 1 and Protocol 2, are designed. Both of them are more efficient than the existing vote validity check mechanisms. Both of them achieve correctness, privacy and robustness. Protocol 2 is especially advanced. It is correct, zero knowledge private, robust, flexible and highly efficient.

References

1. Abe, M., Hoshino, F.: Remarks on mix-network based on permutation networks. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 317–324. Springer, Heidelberg (2001)
2. Aditya, R., Peng, K., Boyd, C., Dawson, E.: Batch Verification for Equality of Discrete Logarithms and Threshold Decryptions. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 494–508. Springer, Heidelberg (2004)
3. Adler, J.M., Dai, W., Green, R.L., Neff, C.A.: Computational details of the votehere homomorphic election system. Technical report, VoteHere Inc. (2000) (last accessed June 22, 2002), `http://www.votehere.net/technicaldocs/hom.pdf`

4. Baudron, O., Fouque, P.-A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: Twentieth Annual ACM Symposium on Principles of Distributed Computing, pp. 274–283 (2001)
5. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
6. Boyd, C., Pavlovski, C.: Attacking and repairing batch verification schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 58–71. Springer, Heidelberg (2000)
7. Chida, K., Yamamoto, G.: Batch processing for proofs of partial knowledge and its applications. IEICE Trans. Fundamentals E91CA(1), 150–159 (2008)
8. Chida, K., Kobayashi, K., Morita, H.: Efficient sealed-bid auctions for massive numbers of bidders with lump comparison. In: Davida, G.I., Frankel, Y. (eds.) ISC 2001. LNCS, vol. 2200, pp. 408–419. Springer, Heidelberg (2001)
9. 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)
10. Damgaård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
11. Fischer, M.J., Micali, S., Rackoff, C.: A secure protocol for the oblivious transfer (extended abstract). Journal of Cryptology 9(3), 191–195 (1996)
12. Fouque, P.-A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001)
13. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
14. Golle, P., Zhong, S., Boneh, D., Jakobsson, M., Juels, A.: Optimistic mixing for exit-polls. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 451–465. Springer, Heidelberg (2002)
15. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (2005)
16. Groth, J., Lu, S.: Verifiable shuffle of large size ciphertexts. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 377–392. Springer, Heidelberg (2007)
17. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2003)
18. Guillou, L.C., Quisquater, J.J.: A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 216–231. Springer, Heidelberg (1990)
19. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
20. Furukawa, J.: Efficient and verifiable shuffling and shuffle-decryption. IEICE Transactions 88-A(1), 172–188 (2005)
21. Katz, J., Myers, S., Ostrovsky, R.: Cryptographic counters and applications to electronic voting. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 78–92. Springer, Heidelberg (2001)
22. Kiayias, A., Yung, M.: Self-tallying elections and perfect ballot secrecy. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 141–158. Springer, Heidelberg (2002)

23. Kikuchi, H., Harkavy, M., Tygar, J.D.: Multi-round anonymous auction. In: Proceedings of the First IEEE Workshop on Dependable and Real-Time E-Commerce Systems, pp. 62–69 (June 1998)
24. Kikuchi, H.: (m+1)st-price auction protocol. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 291–298. Springer, Heidelberg (2002)
25. Kikuchi, H., Hotta, S., Abe, K., Nakanishi, S.: Distributed auction servers resolving winner and winning bid without revealing privacy of bids. In: Proc. of International Workshop on Next Generation Internet (NGITA 2000), pp. 307–312. IEEE, Los Alamitos (2000)
26. Lee, B., Kim, K.: Receipt-free electronic voting through collaboration of voter and honest verifier. In: JW-ISC 2000, pp. 101–108 (2000)
27. Lee, B., Kim, K.: Receipt-free electronic voting scheme with a tamper-resistant randomizer. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003)
28. Andrew Neff, C.: Conducting a universally verifiable electronic election using homomorphic encryption. White paper, VoteHere Inc. (2000)
29. Andrew Neff, C.: A verifiable secret shuffle and its application to e-voting. In: ACM Conference on Computer and Communications Security 2001, pp. 116–125 (2001)
30. Andrew Neff, C.: Verifiable mixing (shuffling) of elgamal pairs (2004), `http://theory.lcs.mit.edu/ rivest/voting/papers/ Neff-2004-04-21-ElGamalShuffles.pdf`
31. Paillier, P.: Public key cryptosystem based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
32. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
33. Peng, K., Aditya, R., Boyd, C., Dawson, E., Lee, B.: A secure and efficient mix-network using extended binary mixing gate. In: Cryptographic Algorithms and their Uses 2004, pp. 57–71 (2004)
34. Peng, K., Boyd, C.: Batch zero knowledge proof and verification and its applications. ACM TISSEC 10(2), Article No. 6 (May 2007)
35. Peng, K., Boyd, C., Dawson, E.: Simple and efficient shuffling with provable correctness and ZK privacy. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 188–204. Springer, Heidelberg (2004)
36. Peng, K., Boyd, C., Dawson, E.: Batch verification of validity of bids in homomorphic e-auction. Computer Communications 29, 2798–2805 (2006)
37. Peng, K., Boyd, C., Dawson, E., Lee, B.: Multiplicative homomorphic e-voting. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 61–72. Springer, Heidelberg (2004)
38. Peng, K., Boyd, C., Dawson, E., Viswanathan, K.: A correct, private and efficient mix network. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 439–454. Springer, Heidelberg (2004)
39. Peng, K., Dawson, E.: Efficient bid validity check in elgamal-based sealed-bid e-auction. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 209–224. Springer, Heidelberg (2007)
40. Schoenmakers, B.: Fully auditable electronic secret-ballot elections. XOOTIC Magazine (July 2000)
41. Wikstrom, D.: A sender verifiable mix-net and a new proof of a shuffle. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 273–292. Springer, Heidelberg (2005)

Secure Hardware Implementation of Non-linear Functions in the Presence of Glitches*

Svetla Nikova[1], Vincent Rijmen[1,2], and Martin Schläffer[2]

[1] Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC and IBBT,
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
[2] Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
martin.schlaeffer@iaik.tugraz.at

Abstract. Hardware implementations of cryptographic algorithms are still vulnerable to side-channel attacks. Side-channel attacks that are based on multiple measurements of the same operation can be countered by employing masking techniques. In the presence of glitches, most of the currently known masking techniques still leak information during the computation of non-linear functions. We discuss a recently introduced masking method which is based on secret sharing and results in implementations that are provable resistant against first-order side-channel attacks, even in the presence of glitches. We reduce the hardware requirements of this method and show how to derive provable secure implementations of some non-linear building blocks for cryptographic algorithms. Finally, we provide a provable secure implementation of the block cipher Noekeon and verify the results.

Keywords: DPA, masking, glitches, sharing, non-linear functions, S-box, Noekeon.

1 Introduction

Side-channel analysis exploits the information leaked during the computation of a cryptographic algorithm. The most common technique is to analyze the power consumption of a cryptographic device using differential power analysis (DPA) [13]. This side-channel attack exploits the correlation between the instantaneous power consumption of a device and the intermediate results of a cryptographic algorithm. A years-long sequence of increasingly secure designs and increasingly sophisticated attack methods breaking again these designs suggests that the problem won't be solved easily. Therefore, securing hardware implementations against advanced DPA attacks is still an active field of research.

In order to counteract DPA attacks several different approaches have been proposed. The general approach is to make the intermediate results of the cryptographic algorithm independent of the secret key. Circuit design approaches

* The work in this paper has been supported in part by the Austrian Government (BMVIT) through the research program FIT-IT Trust (Project ARTEUS) and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 218–234, 2009.

[26,27] try to remove the root of the side-channel leakage by balancing the power consumption of different data values. However, even small remaining asymmetries make a DPA possible. Another method is to randomize the intermediate values of an algorithm by masking them. This can be done at the algorithm level [2,4,9,20], at the gate level [10,28] or even in combination with circuit design approaches [21].

However, recent attacks have shown that masked hardware implementations (contrary to software implementations [24,23]) can still be attacked using even first-order DPA. The problem of most masking approaches is that they were designed and proven secure in the assumption that the output of each gate switches only once per clock cycle. Instead, glitches [22] occur in combinational CMOS circuits and each signal switches several times. Due to these glitches, these circuits are vulnerable to DPA attacks [15,16]. Furthermore, the amount of information leaked cannot be easily determined from the mathematical description of a masked function. It depends too much on the used hardware technology and the way the circuit is actually placed on a chip.

All these approaches start from compact but rather insecure implementations. Subsequently the designers try to solve the known security issues by adding as little hardware as possible. In [19] and in this paper, a different type of approach is followed. The idea is to first start from a very secure implementation and then, make this approach more practical by minimizing the hardware requirements while still maintaining the security level. The approach is based on only a single assumption about the implementation technology, namely: the existence of memory cells that completely isolate the switching characteristics of their inputs from their outputs (e.g. registers). Therefore, it holds for both, FPGAs and ASICs. Secret sharing schemes and techniques from multiparty computation are used to construct combinational logic which is completely independent of the unmasked values.

In this approach the hardware requirements increase with the number of shares and for each non-linear part of a circuit at least three shares (or masks) are needed. Constructing secure implementations of arbitrary functions using only a small number of shares, is a difficult task. In [19] the inversion in GF(16) has been implemented using 5 shares. In this paper we analyze which basic non-linear functions can be securely implemented using only three shares and present a method how to construct such shared Boolean functions. We show that the multiplication in GF(4) and the block cipher Noekeon can be implemented using three shares as well and present the first verification of this method based on computer simulations.

In the next section we give a short overview on simulation based DPA attacks to motivate sharing. Section 3 reformulates and extends the sharing approach of [19]. In Sect. 4 we show which non-linear functions can be securely implemented using three shares. In Sect. 5 we give a provable secure implementation of the S-box of the block cipher Noekeon using three shares. Computer simulations confirm that this shared S-box is not vulnerable to DPA, even in the presence of glitches.

2 DPA Attacks on Masking

Masking is a side-channel countermeasure which tries to randomize the intermediate values of a cryptographic algorithm [18]. Then, the (randomized) power consumption does not correlate with the intermediate values anymore. The most common masking scheme is Boolean or linear masking where the mask is added by an XOR operation. However, one problem of masking is that cryptographic algorithms like AES [7] or ARIA [14] combine linear and non-linear functions. Thus, many different hardware masking schemes and masked gates have been proposed [2,4,20,29] but all of them have been broken already [1,9,16,30]. Even though no wire carries an unmasked value, the power consumption correlates with the unmasked intermediate results of the algorithm.

2.1 Glitches

The problem of these hardware masking schemes is that the effect of glitches has not been considered. Glitches have first been analyzed in [15] and a technique to model glitches has been presented in [25]. Glitches occur because the signals of a combinational circuit can switch more than once if an input changes. The amount of glitches depends on the specific hardware implementation and on the input values of a combinational logic. Since the power consumption of CMOS circuits strongly depends on the amount of glitches, it depends on all inputs as well. The reason why most masking schemes can be attacked is that they combine masks and masked values into the same combinational logic. Since they are not processed independently, it depends on the actual hardware implementation whether a design is secure and cannot be proven during the design process.

2.2 Simulation Based Attacks and Gate Delays

Although it is difficult to verify whether a design or a masking scheme is secure, different simulation techniques have been developed to verify the security of a design. A simple method to analyze a design is by using the assumption that there is no delay at the inputs and inside of a combinational logic. In this case, each signal and output switches at most once and even simple masking schemes are secure using this model. However, in [17] it has been shown by means of computer simulations, that most masked gates can be attacked if the input signals of the combinational logic arrive at different moments in time.

In [8] a model is used where each of the n input signals of a gate can arrive at a different time. Thus, the output can switch up to n times. Although the model does not allow delays inside the gate it takes glitches into account. In their paper a gate is defined to be G-equivalent, if there is no correlation between the number of output transitions and the unmasked value. Since it is not possible to build any non-linear gate which is G-equivalent using standard masking, the weakened requirement of semi-G-equivalence has been defined. Using this notation it is possible to define non-linear masked gates which can be used to build arbitrary circuits. However, the big disadvantage of this method is that semi-G-equivalent circuits still have routing constraints and it depends on the implementation whether a circuit is secure.

Another disadvantage of the previous model is that it does not take delays inside the gate into account. Therefore, a more detailed power consumption model is to count all transitions which occur in a combinational logic. A common method is to use unit delay for all gates and an even more accurate method is to derive the delay of a circuit by back-annotated netlists [12]. In this case, different timing information for different gates and wire lengths are considered. Most secure masking schemes can be broken by performing attacks based on these simulations.

However, none of these methods can *prove* that a circuit is secure in the presence of glitches because each method takes only special cases into account. Therefore, these methods can only be used to *attack* masking schemes. In the following sections we examine a masking scheme based on secret sharing which is provable secure in the presence of glitches.

3 Sharing

In this section we recall the most important elements of the approach in [19]. The idea is to build combinational blocks which are completely independent of the unmasked values. This can be achieved by avoiding that masked values and masks are used as an input to the same combinational circuit. Each function is shared into combinational blocks which are independent of at least one share. Hence, also the number of glitches and the power consumption is independent of any unmasked value. This method can be extended to counter even higher-order attacks by increasing the number of shares [19].

3.1 Terminology

We denote a vector of s shares x_i by $\overline{x} = (x_1, x_2, \ldots, x_s)$ and split a variable x into s additive shares x_i with $x = \bigoplus_i x_i$. Since we are only using linear or XOR masking, the addition $x + y$ is always computed over $\mathrm{GF}(2^n)$ in the remainder of this paper. Further, we will only use (s, s) secret sharing schemes, hence all s shares are needed in order to determine x uniquely. More precisely, we use secret sharing schemes where the conditional probability distribution $\Pr(\overline{x}|x)$ is uniform for every possible sharing vector $\overline{X} = (X_1, X_2, \ldots, X_s)$:

$$\Pr(\overline{x} = \overline{X}) = c \Pr(x = \bigoplus_i X_i) \tag{1}$$

with c a normalization constant, which ensures that $\sum_{\overline{X}} \Pr(\overline{x} = \overline{X}) = 1$. In words, any bias present in the joint distribution of the shares $\overline{x}$ is only due to a bias in the distribution of the unshared variable x. This means, that for each unshared value of x, all possible sharing vectors $\overline{X} = (X_1, X_2, \ldots, X_s)$ with $x = \bigoplus_i X_i$ need to occur equally likely.

3.2 Realization

Assume we want to implement a vectorial Boolean function $z = \mathrm{f}(x)$ with the n-bit input x and the m-bit output z. Then we need a set of functions f_i which together compute the output of f. We call this a *realization* and get the following property:

Property 1 (Correctness [19]). Let $z = \mathrm{f}(x)$ be a vectorial Boolean function. Then the set of functions $\mathrm{f}_i(\overline{x})$ is a *realization* of f if and only if

$$z = \mathrm{f}(x) = \bigoplus_{i=1}^{s} \mathrm{f}_i(\overline{x}) \tag{2}$$

for all shares $\overline{x}$ satisfying $\bigoplus_{i=1}^{s} x_i = x$.

Alternatively, we will denote the n components of x by $(a, b, \dots)$ and the m components of z by $(e, f, \dots)$. We define the vectorial Boolean function of $z = \mathrm{f}(x)$ by:

$$(e, f, \dots) = \mathrm{f}(a, b, \dots) \tag{3}$$

and its m Boolean component functions $\mathrm{f}^j(x)$ of $\mathrm{f}(x)$ as follows:

$$\begin{aligned} e &= \mathrm{f}^1(x) = \mathrm{f}^1(a, b, \dots) \\ f &= \mathrm{f}^2(x) = \mathrm{f}^2(a, b, \dots) \end{aligned} \tag{4}$$

To construct a shared implementation of the function f, each element of the input x and of the result z is divided into s shares. To divide the function f, we need to split each component function f^j into s shared functions with:

$$\begin{aligned} e &= e_1 + \dots + e_s = \mathrm{f}^1((a_1 + \dots + a_s), (b_1 + \dots + b_s), \dots) \\ f &= f_1 + \dots + f_s = \mathrm{f}^2((a_1 + \dots + a_s), (b_1 + \dots + b_s), \dots) \end{aligned} \tag{5}$$

3.3 Non-completeness

The next property is important to prove the security of a realization of a function. We denote the reduced vector $(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_s)$ by $\overline{x}_i$.

Property 2 (Non-completeness [19]). Every function is independent of at least one share of the input variable x and consequently, independent of at least one share of each component. Without loss of generality, we require that z_i is independent of x_i:

$$\begin{aligned} z_1 &= \mathrm{f}_1(x_2, x_3, \dots, x_s) &&= \mathrm{f}_1(\overline{a}_1, \overline{b}_1, \dots) \\ z_2 &= \mathrm{f}_2(x_1, x_3, \dots, x_s) &&= \mathrm{f}_2(\overline{a}_2, \overline{b}_2, \dots) \\ &\dots \\ z_s &= \mathrm{f}_s(x_1, x_2, \dots, x_{s-1}) &&= \mathrm{f}_s(\overline{a}_s, \overline{b}_s, \dots) \end{aligned} \tag{6}$$

Theorem 2 and Theorem 3 of [19] are essential. These theorems state that the shared components f_i of a function $z = \mathrm{f}(x)$ are independent of the input x and output z, as long as the shared realization satisfies Property 2 and the input vectors satisfy (1). Therefore, also the mean power consumption, or any other characteristic of an implementation of each component f_i is independent of the unmasked values x and z, even in the presence of glitches or the delayed arrival time of some inputs. More general, to counter DPA attacks of order r, each shared component function needs to be independent of at least r shares.

When partitioning each Boolean component function f^j, we need to ensure that Property 2 is satisfied for each component. As we have defined in Property 2, each output share with index i needs to be independent of all input shares with the same index i, namely independent of a_i, b_i, ...

$$e_i = \mathrm{f}_i^1(\overline{a}_i, \overline{b}_i, \dots)$$
$$f_i = \mathrm{f}_i^2(\overline{a}_i, \overline{b}_i, \dots)$$

3.4 Uniform

In principle, Property 2 is sufficient to construct secure implementations of arbitrary (vectorial) Boolean functions. However, the number of required shares, and thereby the size of the circuit, grows rapidly with the algebraic degree of the function. This can be reduced by splitting the function into stages. The only requirement is that the switching characteristics between each stage are isolated. This can be achieved by a pipelined implementation, where the different stages are separated by registers or latches. In order to design the different stages separately, we need to ensure (1) for the input shares $\overline{x}$ of each stage. Since every output of a stage is used as the input in the next stage we need to make assumptions about the probability distribution of the output shares $\overline{z}$ of a shared function as well. The following property ensures that if the input-share distribution satisfies (1), also the output-share distribution does:

Property 3 (Uniform [19]). [1] A realization of $z = \mathrm{f}(x)$ is *uniform*, if for all distributions of the inputs x and for all input share distributions satisfying (1), the conditional probability

$$\Pr(\overline{z} = \overline{Z} | z = \bigoplus_i Z_i) \tag{7}$$

is constant.

4 Sharing Non-linear Functions Using 3 Shares

In [19] it has been proven, that at least three shares are needed to fulfill Property 2 for any non-linear function. However, in their paper the best result was a

[1] This property is called *Balance* instead of *Uniform* in [19].

uniform implementation for the inversion in GF(16) using 5 shares. In this section we analyze which basic non-linear functions can be shared using only three shares and present a method to construct them, such that all three properties are fulfilled. Finally, we show how the multiplication in GF(4), which is often used in the implementation of the AES S-box [5], can be successfully shared using three shares.

4.1 Constructing Non-linear Shared Functions

We construct non-linear shared functions by splitting the shared function, such that only Property 1 and 2 are fulfilled first. In [19] it has been proven that this is always possible for any function of algebraic degree two. If we continue with the notation of Sect. 3.3 terms of degree two can only be placed in the share with the missing index. For example, the term $a_1 b_2$ can only be a part of function (or share) f_3 since f_1 has to be independent of a_1 and f_2 of b_2. However, all linear terms and quadratic terms with equal index i can be placed in one of the two shared functions f^j with $i \neq j$.

Usually, Property 3 is not fulfilled after this step. To change the output-share distribution we can add other terms to the non-complete shared functions. These *correction terms* must not violate the first two properties but can be used to fulfill Property 3. Hence, only a special set of correction terms can be added to the individual shares. To maintain Property 1, it is only possible to add the same term to an even number of different shares. This ensures that the correction terms cancel out after adding the shares. To retain Property 2 we can only add terms which are independent of at least two shares. Therefore, only linear terms and terms with equal index i can be used as correction terms.

4.2 Sharing Non-linear Functions with 2 Inputs

In [19] it has been shown that no shared AND gate using three shares exists which fulfills all three properties. Note that using *any* single non-linear gate we would be able to build arbitrary circuits. Hence, we generalize the idea in this section.

Theorem 1. *No non-linear gate or Boolean function with two inputs and one output can be shared using three shares.*

Proof. All non-linear Boolean functions with two inputs and one output can be defined in algebraic normal form (ANF) by the following 8 functions with parameters $k_0, k_1, k_2 \in \{0, 1\}$ and index $i = k_0 \cdot 4 + k_1 \cdot 2 + k_2$:

$$f_i(a, b) = k_0 + k_1 a + k_2 b + ab. \tag{8}$$

To share these non-linear Boolean functions using three shares, we first split the inputs a and b into three shares and get the following functions:

$$\begin{aligned}
e_1 + e_2 + e_3 &= f_i(a_1 + a_2 + a_3, b_1 + b_2 + b_3) \\
&= k_0 + k_1(a_1 + a_2 + a_3) + k_2(b_1 + b_2 + b_3) \\
&\quad + (a_1 + a_2 + a_3) \cdot (b_1 + b_2 + b_3) \\
&= k_0 + k_1(a_1 + a_2 + a_3) + k_2(b_1 + b_2 + b_3) \\
&\quad + a_1b_1 + a_1b_2 + a_1b_3 + a_2b_1 + a_2b_2 + a_2b_3 + a_3b_1 + a_3b_2 + a_3b_3.
\end{aligned}$$

Then, terms with different indices are placed into the share with the missing index and the share for all other terms can be chosen freely.

To satisfy Property 3, the shared-output distribution of (e_1, e_2, e_3) needs to be uniform for each unshared input value (a, b). In other words, each possible shared output value has to occur equally likely.The input of the unshared functions can take the 4 values $(a, b) \in \{00, 01, 10, 11\}$. In the case of the shared multiplication with $f(a, b) = ab$, we get for the input $(a, b) = 00$ the output $e = e_1 + e_2 + e_3 = 0$ and the distribution of its shared output values $(e_1, e_2, e_3) \in \{000, 011, 101, 110\}$ has to be uniform.

For each of the 8 non-linear functions all possible correction terms are the constant term, the 6 linear terms $a_1, a_2, a_3, b_1, b_2, b_3$ and the 3 quadratic terms a_1b_1, a_2b_2, a_3b_3. Due to the small number of correction terms we can evaluate all possibilities and prove that no combinations leads to a uniform shared representation. It follows that a shared non-linear function with 2 inputs, one output and 3 shares does not exist. □

4.3 Sharing Non-linear Functions with 3 Inputs

The result of the previous section leads to the question if there are any non-linear functions that can be shared using three shares. To answer this question we look at the class of non-linear Boolean functions with 3 inputs and one output bit:

$$f_i(a, b, c) = k_0 + k_1a + k_2b + k_3c + k_4ab + k_5ac + k_6bc + k_7abc \qquad (9)$$

with $k_0, \ldots, k_7 \in \{0, 1\}$. As shown in [19, Theorem 1], a Boolean function of algebraic degree 3 can never be shared using three shares. Therefore, we always require $k_7 = 0$. To get a non-linear function at least one of the coefficients with degree two (k_4, k_5, k_6) needs to be non-zero and we get 112 non-linear functions. To share these 112 functions, we split each input and output into three shares and get:

$$\begin{aligned}
e_1 + e_2 + e_3 &= f_i(a_1 + a_2 + a_3, b_1 + b_2 + b_3, c_1 + c_2 + c_3) \\
&= k_0 + k_1(a_1 + a_2 + a_3) + k_2(b_1 + b_2 + b_3) + k_3(c_1 + c_2 + c_3) \\
&\quad + k_4(a_1b_1 + a_1b_2 + a_1b_3 + a_2b_1 + a_2b_2 + a_2b_3 + a_3b_1 + a_3b_2 + a_3b_3) \\
&\quad + k_5(a_1c_1 + a_1c_2 + a_1c_3 + a_2c_1 + a_2c_2 + a_2c_3 + a_3c_1 + a_3c_2 + a_3c_3) \\
&\quad + k_6(b_1c_1 + b_1c_2 + b_1c_3 + b_2c_1 + b_2c_2 + b_2c_3 + b_3c_1 + b_3c_2 + b_3c_3)
\end{aligned}$$

These functions can be shared using the same method as in the previous section but we can now use the following 22 correction terms:

linear:	$1, a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3$
degree 2:	$a_1b_1, a_2b_2, a_3b_3, a_1c_1, a_2c_2, a_3c_3, b_1c_1, b_2c_2, b_3c_3$
degree 3:	$a_1b_1c_1, a_2b_2c_2, a_3b_3c_3$

By adding at least three correction terms, many uniform shared functions for all of the 112 non-linear functions can be found.

4.4 Shared Multiplication in GF(4)

In this section we show that the multiplication in GF(4) can be successfully shared using three shares. We have implemented the multiplication in GF(4) using normal bases. The used normal basis is (v, v^2) and the two elements are represented by $v = 01$ and $v^2 = 10$. The zero element is represented by 00 and the one element by 11. We define the multiplication in GF(4) using this normal basis by:

$$\begin{aligned}(e, f) &= (a, b) \times (c, d)\\ e &= ac + (a + b)(c + d)\\ f &= bd + (a + b)(c + d)\end{aligned}$$

and get the following multiplication tables:

×	0	1	v	v^2
0	0	0	0	0
1	0	1	v	v^2
v	0	v	v^2	1
v^2	0	v^2	1	v

×	00	11	01	10
00	00	00	00	00
11	00	11	01	10
01	00	01	10	11
10	00	10	11	01

To construct a shared multiplication in GF(4), each of the 4 inputs a, b, c and d and the results e and f are divided into three shares:

$$\begin{aligned}(e_1 + e_2 + e_3) = {}&(a_1 + a_2 + a_3)(c_1 + c_2 + c_3)\\ &+ ((a_1 + a_2 + a_3) + (b_1 + b_2 + b_3))((c_1 + c_2 + c_3) + (d_1 + d_2 + d_3))\\ (f_1 + f_2 + f_3) = {}&(b_1 + b_2 + b_3)(d_1 + d_2 + d_3)\\ &+ ((a_1 + a_2 + a_3) + (b_1 + b_2 + b_3))((c_1 + c_2 + c_3) + (d_1 + d_2 + d_3))\end{aligned}$$

After expanding the multiplication formulae, each term of the two component functions is placed into one of the three output shares (see App. A). Since the multiplication in GF(4) consists only of quadratic terms it is always possible to fulfill Property 2.

To fulfill Property 3 we need a uniform output-share distribution for each of the 16 unshared input values (a, b, c, d). For example, the input $(a, b, c, d) = 0111$ results in the output $(e, f) = 01$. The shared result is uniform, if each possible value of $(e_1, e_2, e_3, f_1, f_2, f_3)$ with $e_1 + e_2 + e_3 = 0$ and $f_1 + f_2 + f_3 = 1$ occurs equally likely. We have 2^4 unshared and 2^{12} shared input values and hence, we get $2^{12-4} = 2^8$ values for each unshared output (e, f). Since two bits of the shares $(e_1, e_2, e_3, f_1, f_2, f_3)$ have already been determined, each of the remaining 2^4 shares has to occur $2^{8-4} = 2^4$ times.

The input of the shared multiplication are the 12 variables a_i, b_i, c_i and d_i with $i \in \{1, 2, 3\}$. When searching for uniform functions, we can add only correction terms which have the same index i in all of its elements. We get 1 constant, 4 linear and 6 quadratic terms, 4 terms of degree 3 and 1 term $(a_i b_i c_i d_i)$ of degree 4. This gives 16 possible correction terms for each shared component function of e and f. The search space of finding a uniform representation can be reduced by allowing only a limited number of correction terms. Further, e_i and f_i are rotation symmetric and each Boolean shared function needs to be balanced. Using at most 6 linear or quadratic correction terms, we have found thousands of uniform realizations of the multiplication in GF(4) using three shares. Hence, a hardware designer has still lots of freedom to choose an efficient implementation and we give an example for found correction terms in App. A.

5 Noekeon

Noekeon [6] is a block cipher with a block and key length of 128 bits, which has been designed to counter implementation attacks. It is an iterated cipher consisting of 16 identical rounds. In each round 5 simple round transformations are applied. The cipher is completely linear except for the non-linear S-box Gamma. The linear parts can be protected against first-order DPA using one mask (two shares), whereas for the non-linear part this is not possible. In this section we show how the non-linear S-box Gamma can be successfully shared using 3 shares. Finally, we show that this shared function is secure in the presence of glitches by performing a simulation based on a back-annotated netlist.

5.1 The S-Box Gamma

The non-linear 4-bit S-box Gamma is defined by Table 1) and consists of two equal non-linear layers $NL(x)$, separated by a linear layer $L(x)$:

$$S(x) = NL(L(NL(x)) \tag{10}$$

The non-linear layer $(e, f, g, h) = NL(a, b, c, d)$, which consists of only one AND, one NOR and two XOR operations, and the linear layer $(i, j, k, l) = L(a, b, c, d)$ are defined by:

$$\begin{aligned} h &= d & i &= a \\ g &= c & j &= a + b + c + d \\ f &= b + \neg(c \vee d) = 1 + b + c + d + cd & k &= b \\ e &= a + (b \wedge c) = a + bc & l &= d \end{aligned}$$

Table 1. The substitution table of the 4-bit S-box Gamma of the block cipher Noekeon

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	7	A	2	C	4	8	F	0	5	9	1	E	3	D	B	6

5.2 Sharing the Noekeon S-Box Using 3 Shares

Since the algebraic degree of this function is 3, the whole function cannot be shared using 3 shares. However, if we split Gamma into two stages with algebraic degree two, we can share it using 3 shares again. We split Gamma after the linear layer and combine the first non-linear layer with the linear layer to get $y = L(NL(x))$ and $z = NL(y)$. This results in less complex formulae and we get for the ANF of the resulting 8 Boolean component functions $(i, j, k, l) = L(NL(a, b, c, d))$ and $(e, f, g, h) = NL(i, j, k, l)$:

$$\begin{aligned} i &= d & \qquad e &= i + jk \\ j &= 1 + b + c + d + cd & \qquad f &= 1 + j + k + l + kl \\ k &= 1 + a + b + bc + cd & \qquad g &= k \\ l &= a + bc & \qquad h &= l \end{aligned}$$

To share these functions we need to share the 4 inputs and outputs of each layer and get 24 shared Boolean functions. We place the terms depending on their index into the regarding output share which results already in uniform shared functions for both stages of Gamma. The formulae for the two steps of the shared Noekeon S-box using three shares are shown in App. B. We have implemented both the protected and the unprotected Noekeon S-box using a $0.35\mu m$ standard cell library [3]. A schematic of the shared Noekeon S-box is shown in Fig. 1. In a straight forward implementation using just the ANF of the functions, the protected S-box is approximately 3.5 times larger than the unprotected S-box (188 gate equivalents compared to 54 gate equivalents). Since there is room for further improvements and the linear parts of the Noekeon cipher can be implemented using two shares only, the overall size of the cipher can still be less than 3.5 times larger. This shows that shared implementations can already compete with other hardware countermeasures.

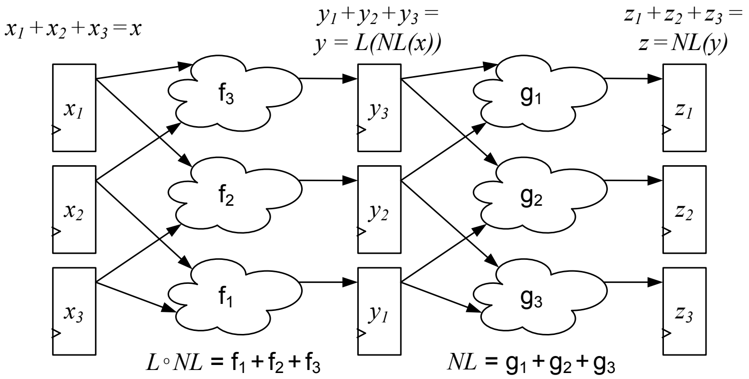


Fig. 1. A schematic of the shared Noekeon S-box using three shares

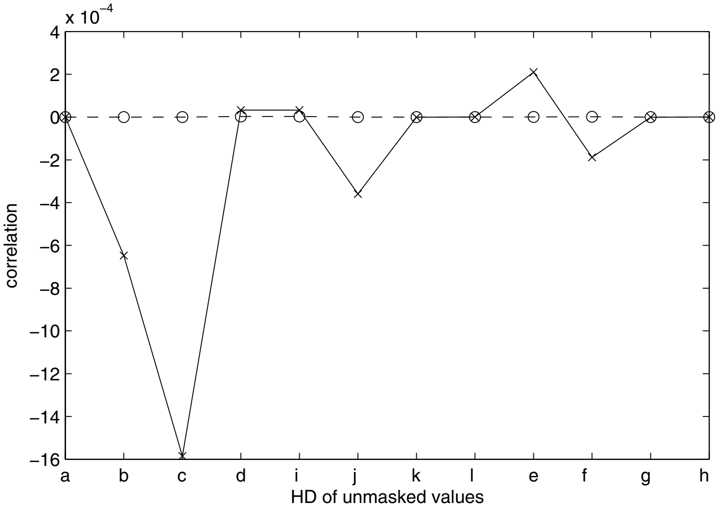


Fig. 2. The correlation of the computer simulated attack on the two implementations of the Noekeon S-box. The solid line shows the result for the S-box without registers, and the dashed line for the S-box with registers in between the two stages.

5.3 Simulation Based on the Transition Count Model

Any shared implementation of these two stages of the S-box with registers in between is secure even in the presence of glitches. Further, we do not have any timing constraints or the need for balanced wires. The resulting shared component functions can be implemented on any hardware and optimized in any form, as long as there is a glitch resistant layer in between and the functionality stays the same. In order to study the resistance of this implementation in the presence of glitches, we have synthesized the circuit and performed a simulation of an attack using the transition count model. We have used a back-annotated netlist to derive the timing delays. Note that this is only one example of an implementation. However, by computing the correlation coefficient between the unmasked values and the number of transitions, we can show that the implementation with registers in between is secure in this model, whereas the implementation without registers between the two stages is not.

The shared S-box has 12 inputs and outputs. To verify if a circuit is secure we count the total number of transitions for each possible input transition. Every input can perform one out of 4 transitions, $0 \rightarrow 0, 0 \rightarrow 1, 1 \rightarrow 0, 1 \rightarrow 1$. Thus, we need to simulate $4^{12} = 16.777.216$ transitions. We do not need to simulate different arrival times since glitches occur due to the internal gate delays. Figure 2 shows the correlation between the number of transitions and the Hamming Distance (HD) of the unmasked values before and after the input transition. The correlation for both implementations with and without registers between the two stages is shown. These results demonstrate the DPA resistance of a shared implementation using three shares in the presence of glitches. However, the final confirmation for the security of this method can only be provided by an on-chip implementation which is clearly out of scope of this paper.

6 Conclusion

In the side-channel resistant implementation method proposed in [19] the number of shares grows with the complexity of the function to be protected. Consequently, also the number of gates required increases. In this paper we have analyzed which basic non-linear functions can be securely implemented using the minimum of three shares and presented a method to construct shared Boolean functions. We have implemented the block cipher Noekeon using only three shares by introducing pipelining stages separated by latches or registers. Finally, we have presented the first verification of this implementation method based on computer simulations.

In this work we have shown that it is possible to implement cryptographic functions using much less hardware requirements than proposed in [19]. By varying the number of shares between non-linear and linear layers, the hardware requirements for a full cipher can even be further reduced. This makes the shared implementation method more competitive with other countermeasures while maintaining its high security level. Future work is to further reduce the hardware requirements and securely implement more complex non-linear functions such as the AES or ARIA S-boxes [11], which is still a mathematically challenging task. A reasonable trade-off between the number of shares and pipelining stages needs to be found. Further analysis is required to investigate the practical resistance of shared implementations against higher-order and template attacks.

References

1. Akkar, M.-L., Bevan, R., Goubin, L.: Two Power Analysis Attacks against One-Mask Methods. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 332–347. Springer, Heidelberg (2004)
2. Akkar, M.-L., Giraud, C.: An Implementation of DES and AES, Secure against Some Attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001)
3. Austria Microsystems. Standard Cell Library 0.35μm CMOS (C35), `http://asic.austriamicrosystems.com/databooks/c35/databook_c35_33`
4. Blömer, J., Guajardo, J., Krummel, V.: Provably Secure Masking of AES. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 69–83. Springer, Heidelberg (2004)
5. Canright, D.: A Very Compact S-Box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer, Heidelberg (2005)
6. Daemen, J., Peeters, M., Assche, G.V., Rijmen, V.: Nessie proposal: NOEKEON. Submitted as an NESSIE Candidate Algorithm (2000), `http://www.cryptonessie.org`
7. Daemen, J., Rijmen, V.: AES proposal: Rijndael. Submitted as an AES Candidate Algorithm. Submitted as an AES Candidate Algorithm (2000), `http://www.nist.gov/aes`

8. Fischer, W., Gammel, B.M.: Masking at Gate Level in the Presence of Glitches. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 187–200. Springer, Heidelberg (2005)
9. Golic, J.D., Tymen, C.: Multiplicative Masking and Power Analysis of AES. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 198–212. Springer, Heidelberg (2003)
10. Ishai, Y., Sahai, A., Wagner, D.: Private Circuits: Securing Hardware against Probing Attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
11. Kim, C., Schläffer, M., Moon, S.: Differential Side Channel Analysis Attacks on FPGA Implementations of ARIA. Electronics and Telecommunications Research Institute (ETRI) in Daejeon, South Korea 30(2), 315–325 (2008)
12. Kirschbaum, M., Popp, T.: Evaluation of Power Estimation Methods Based on Logic Simulations. In: Posch, K.C., Wolkerstorfer, J. (eds.) Proceedings of Austrochip 2007, October 11, 2007, Graz, Austria, pp. 45–51. Verlag der Technischen Universität Graz (October 2007) ISBN 978-3-902465-87-0
13. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
14. Kwon, D., Kim, J., Park, S., Sung, S.H., Sohn, Y., Song, J.H., Yeom, Y., Yoon, E.-J., Lee, S., Lee, J., Chee, S., Han, D., Hong, J.: New Block Cipher: ARIA. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 432–445. Springer, Heidelberg (2004)
15. Mangard, S., Popp, T., Gammel, B.M.: Side-Channel Leakage of Masked CMOS Gates. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 351–365. Springer, Heidelberg (2005)
16. Mangard, S., Pramstaller, N., Oswald, E.: Successfully Attacking Masked AES Hardware Implementations. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 157–171. Springer, Heidelberg (2005)
17. Mangard, S., Schramm, K.: Pinpointing the Side-Channel Leakage of Masked AES Hardware Implementations. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 76–90. Springer, Heidelberg (2006)
18. Messerges, T.S.: Securing the AES Finalists Against Power Analysis Attacks. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 150–164. Springer, Heidelberg (2001)
19. Nikova, S., Rechberger, C., Rijmen, V.: Threshold Implementations Against Side-Channel Attacks and Glitches. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (2006)
20. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A Side-Channel Analysis Resistant Description of the AES S-Box. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 413–423. Springer, Heidelberg (2005)
21. Popp, T., Mangard, S.: Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 172–186. Springer, Heidelberg (2005)
22. Rabaey, J.M.: Digital Integrated Circuits: A Design Perspective. Prentice-Hall, Inc., Upper Saddle River (1996)
23. Rivain, M., Dottax, E., Prouff, E.: Block Ciphers Implementations Provably Secure Against Second Order Side Channel Analysis. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 127–143. Springer, Heidelberg (2008)
24. Schramm, K., Paar, C.: Higher Order Masking of the AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 208–225. Springer, Heidelberg (2006)

25. Suzuki, D., Saeki, M., Ichikawa, T.: DPA Leakage Models for CMOS Logic Circuits. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 366–382. Springer, Heidelberg (2005)
26. Tiri, K., Verbauwhede, I.: Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 125–136. Springer, Heidelberg (2003)
27. Tiri, K., Verbauwhede, I.: A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In: DATE, pp. 246–251. IEEE Computer Society Press, Los Alamitos (2004)
28. Trichina, E., Korkishko, T., Lee, K.-H.: Small Size, Low Power, Side Channel-Immune AES Coprocessor: Design and Synthesis Results. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 113–127. Springer, Heidelberg (2005)
29. Trichina, E., De Seta, D., Germani, L.: Simplified Adaptive Multiplicative Masking for AES. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 187–197. Springer, Heidelberg (2003)
30. Waddle, J., Wagner, D.: Towards efficient second-order power analysis. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 1–15. Springer, Heidelberg (2004)

A Formulas for the Multiplication in GF(4)

An example of the formulae in ANF for the shared Multiplication in GF(4) using 3 shares together with the correction terms e'_1, e'_2, e'_3 and f'_1, f'_2, f'_3:

$$\begin{aligned} e_1 = {} & a_2d_2 + a_2d_3 + a_3d_2 + \\ & b_2c_2 + b_2c_3 + b_3c_2 + \\ & b_2d_2 + b_2d_3 + b_3d_2 \end{aligned}$$

$$\begin{aligned} e_2 = {} & a_1d_3 + a_3d_1 + a_3d_3 + \\ & b_1c_3 + b_3c_1 + b_3c_3 + \\ & b_1d_3 + b_3d_1 + b_3d_3 \end{aligned}$$

$$\begin{aligned} e_3 = {} & a_1d_1 + a_1d_2 + a_2d_1 + \\ & b_1c_1 + b_1c_2 + b_2c_1 + \\ & b_1d_1 + b_1d_2 + b_2d_1 \end{aligned}$$

$$\begin{aligned} f_1 = {} & a_2c_2 + a_2c_3 + a_3c_2 + \\ & a_2d_2 + a_2d_3 + a_3d_2 + \\ & b_2c_2 + b_2c_3 + b_3c_2 \end{aligned}$$

$$\begin{aligned} f_2 = {} & a_1c_3 + a_3c_1 + a_3c_3 + \\ & a_1d_3 + a_3d_1 + a_3d_3 + \\ & b_1c_3 + b_3c_1 + b_3c_3 \end{aligned}$$

$$\begin{aligned} f_3 = {} & a_1c_1 + a_1c_2 + a_2c_1 + \\ & a_1d_1 + a_1d_2 + a_2d_1 + \\ & b_1c_1 + b_1c_2 + b_2c_1 \end{aligned}$$

$$\begin{aligned} e'_1 &= a_3 + b_2c_2 + b_3c_3 + a_2c_2 \\ e'_2 &= a_1 + a_3 + d_1 + b_1c_1 + b_3c_3 + a_1d_1 \\ e'_3 &= a_1 + d_1 + b_1c_1 + b_2c_2 + a_2c_2 + a_1d_1 \end{aligned}$$

$$\begin{aligned} f'_1 &= c_3 + d_3 + a_2c_2 + a_3c_3 + b_2d_2 + b_3d_3 \\ f'_2 &= c_3 + d_1 + d_3 + a_3c_3 + b_1d_1 + b_3d_3 \\ f'_3 &= d_1 + a_2c_2 + b_1d_1 + b_2d_2 \end{aligned}$$

B Formulas for the Noekeon S-Box Using 3 Shares

The formulae in ANF of the shared Noekeon S-box or non-linear function Gamma using 3 shares. The first step combines the first non-linear layer with the linear layer:

$$\begin{aligned}
i_1 &= d_2\\
i_2 &= d_3\\
i_3 &= d_1\\
j_1 &= 1 + b_2 + c_2 + d_2 + c_2d_2 + c_3d_2 + c_2d_3\\
j_2 &= b_3 + c_3 + d_3 + c_3d_1 + c_1d_3 + c_3d_3\\
j_3 &= b_1 + c_1 + d_1 + c_1d_1 + c_2d_1 + c_1d_2\\
k_1 &= 1 + a_2 + b_2 + b_2c_2 + b_3c_2 + b_2c_3 + c_2d_2 + c_3d_2 + c_2d_3\\
k_2 &= a_3 + b_3 + b_3c_1 + b_1c_3 + b_3c_3 + c_3d_1 + c_1d_3 + c_3d_3\\
k_3 &= a_1 + b_1 + b_1c_1 + b_2c_1 + b_1c_2 + c_1d_1 + c_2d_1 + c_1d_2\\
l_1 &= a_2 + b_2c_2 + b_3c_2 + b_2c_3\\
l_2 &= a_3 + b_3c_1 + b_1c_3 + b_3c_3\\
l_3 &= a_1 + b_1c_1 + b_2c_1 + b_1c_2.
\end{aligned}$$

The second step consists only of the second non-linear layer:

$$\begin{aligned}
e_1 &= i_2 + j_2k_2 + j_3k_2 + j_2k_3\\
e_2 &= i_3 + j_3k_1 + j_1k_3 + j_3k_3\\
e_3 &= i_1 + j_1k_1 + j_2k_1 + j_1k_2\\
f_1 &= 1 + j_2 + k_2 + l_2 + k_2l_2 + k_3l_2 + k_2l_3\\
f_2 &= j_3 + k_3 + l_3 + k_3l_1 + k_1l_3 + k_3l_3\\
f_3 &= j_1 + k_1 + l_1 + k_1l_1 + k_2l_1 + k_1l_2\\
g_1 &= k_2\\
g_2 &= k_3\\
g_3 &= k_1\\
h_1 &= l_2\\
h_2 &= l_3\\
h_3 &= l_1.
\end{aligned}$$

Novel PUF-Based Error Detection Methods in Finite State Machines*

Ghaith Hammouri, Kahraman Akdemir, and Berk Sunar

Worcester Polytechnic Institute
100 Institute Road, Worcester, MA 01609-2280
{hammouri,kahraman,sunar}@wpi.edu

Abstract. We propose a number of techniques for securing finite state machines (FSMs) against fault injection attacks. The proposed security mechanisms are based on physically unclonable functions (PUFs), and they address different fault injection threats on various parts of the FSM. The first mechanism targets the protection of state-transitions in a specific class of FSMs. The second mechanism addresses the integrity of secret information. This is of particular interest in cryptographic FSMs which require a secret key. Finally, the last mechanism we propose introduces a new fault-resilient error detection network (EDN). Previous designs for EDNs always assume resilience to fault injection attacks without providing a particular construction. The PUF-based EDN design is suitable for a variety of applications, and is essential for most fault resilient state machines. Due to the usage of PUFs in the proposed architectures, the state machine will enjoy security at the logical level as well as the physical level.

Keywords: Fault-resilience, state-machines, adversarial-faults, PUF.

1 Introduction

Over the last decade, side-channel attacks drew significant attention in the field of cryptography. This class of attacks mainly depend on leaking secret information through implementation specific side-channels. Various attack mechanisms and countermeasures have been published in this domain, yet this is still an active field of research both in academia and industry. Roughly speaking, side-channel attacks can be classified into two main categories which are passive and active attacks.

Passive attacks depend on observing and analyzing the implementation specific characteristics of the chip. Power and timing analysis are the most powerful and common attacks in this branch. In these attack mechanisms, power and timing information of the cryptographic hardware is measured/observed at various steps and a following statistical analysis reveals information about the secret in the system [24,25]. On the other hand, in an active attack the adversary actually

* This material is based upon work supported by the National Science Foundation under NSF Grants No. CNS-0831416 and CNS-0716306.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 235–252, 2009.

changes the state/value of specific data in the chip in order to gain access to the secret information. In other words, the attacker injects faults to specific parts of an integrated circuit (IC) through side-channels and uses the injected flaw to gain access to the secret information such as the cryptographic key. Boneh et al. [8] demonstrated the effectiveness of these attacks on breaking otherwise impenetrable cryptographic systems. Some of the most crucial active fault attack mechanisms discussed in the literature are external voltage variations, external clock transients, temperature variations and bit-flips using highly dense optical laser beams [38,36]. For more information on the details of side-channel attacks the reader is referred to [4,31].

Due to their strength, side-channel attacks create a serious and crucial threat to the existing cryptographic infrastructure. Especially active attacks, even though harder to introduce, are very effective. Various counter-measures have been suggested to counter act this class of attacks and to provide secure/fault-resilient cryptographic hardware designs. Most of the solutions proposed to secure against fault injection attacks utilize some form of a concurrent error detection (CED) mechanism [6,22,12]. Non-linear error detection codes, a version of CED, arose as the most effective of these defense mechanisms. These codes provide a high level of robustness (the maximum error masking probability is minimized over all possible error vectors) with a relatively high hardware cost [20,28,27,19,21].

Another countermeasure proposed against side-channel attacks is the dual-rail encoding. In these schemes, a data value is represented by two wires. In this case, two out of four states represent the data and the two extra states which are “quiet” and “alarm” can be used for security purposes. Consequently, utilizing the dual railing for the cryptographic designs can potentially provide security against active and passive side-channel attacks [39,9,42].

A different interesting approach to protect against active attacks in authentication schemes are Physically Unclonable Functions (PUFs) [10,33,37]. A PUF is a physical pseudo-random function which exploits the small variances in the wire and gate delays inside an integrated circuit (IC). Even when the ICs are designed to be logically identical, the delay variances will be unique to each IC. These variances depend on highly unpredictable factors, which are mainly caused by the inter-chip manufacturing variations. Hence, given the same input, the PUF circuit will return a different output on different chipsets [30]. Additionally, if the PUFs are manufactured with high precision, any major external physical influence will change the PUF function and therefore change the output. These features are indeed very attractive for any low cost scheme hoping to achieve tamper-resilience against active attacks.

In this paper, we focus our attention on active attacks and present a CED design based on PUFs. As we will see in the next section we focus our attention on error detection in the control logic of hardware implementations. The remainder of this paper is organized as follows: Section 2 discusses our motivation and contributions. Section 3 introduces the necessary background on PUF circuits. Our PUF-based approach to secure known-path state machines is discussed in Section 4. Next, the key integrity scheme utilizing PUFs is described in Section 5. In

Section 6, a PUF-based secure error detection network (EDN) is presented and conclusions are summarized in Section 7.

2 Motivation

Almost all of the research related to CED focused on protecting the datapath of the cryptographic algorithms. However, control unit security against active fault attacks was mostly neglected and therefore has led to a significant security breach in many cryptographic hardware implementations. A literature review indicates that there is not much work done about this topic, except some simple single-bit error detection and correction scheme descriptions in the aerospace and communications areas [26,5]. These schemes are proposed against naturally occurring faults (for example due to radioactive radiation and mostly single event upsets) on the control units and are not nearly sufficient when a highly capable adversary is considered. For example, the encryption states in a finite state machine (FSM) can be bypassed and the secret information can be easily revealed by this type of an adversary. Similarly, the state which validates the login information can be skipped and the attacker can directly impersonate a valid user with a limited effort. At the end of the day, the states in a control unit are implemented using flip-flops, which are vulnerable to bit flips and fault injection attacks.

After observing this gap in control unit fault-tolerance, Gaubatz et al. [13] conducted an initial study on this issue. In their paper, the authors describe an attack scenario on the control structure of the Montgomery Ladder Algorithm. In this scenario the adversary can easily access to the secret information by actively attacking to the controller segment of the hardware. As a solution to this problem, the authors propose applying linear error detection codes on the control units of cryptographic applications. Although the presented approach is quite interesting, it can only provide a limited level of security and robustness.

Our Contribution: In this paper, we present a high level description of a technique which uses PUFs to secure a class of finite state machines against fault injection attacks. Our proposal offers a two-layer security approach. In the first layer the PUF's functionality is used to produce a checksum mechanism on the logical level and in the second layer, the intrinsic sensitivity of the PUF construction is used as a protection mechanism on the physical level. An injected error has a high probability of either causing a change in the checksum, or causing a change in the PUF characteristics, therefore signaling an attack. With this dual approach our proposal opens an interesting area of research which explores hardware specific features of state machines. The proposed design integrates with a finite state machine in three different ways: 1) It provides a checksum mechanism for the state transitions 2) It provides key integrity protection for any secrets used by the state machine 3) It provides a novel fault-resilient implementation of the error detection network. Our work here is the first study on utilizing PUFs to secure the control logic in hardware implementations. In addition, the utilization of PUFs proved to be extremely suitable when the hardware resources are limited.

3 Physically Unclonable Functions

A PUF is a challenge-response circuit which takes advantage of the interchip variations. The idea behind a PUF is to have the same logical circuit produce a different output depending on the actual implementation parameters of the circuit. The variations from one circuit implementation to another are not controllable and are directly related to the physical aspects of the fabrication environment. These aspects include temperature, pressure levels, electromagnetic waves and quantum fluctuations. Therefore, two identical logical circuits sitting right next to each other on a die in a fabrication house might still have quite different input-output behavior due to the nature of a PUF.

The reason one might seek to explore such a property is to prevent any ability to clone the system. Additionally, because of the high sensitivity of these interchip variations it becomes virtually impossible for an attacker to accurately reproduce the hardware. Another major advantage of the PUF's sensitivity is to prevent physical attacks on the system. Trying to tap into the circuit will cause a capacitance change therefore changing the output of the PUF circuit. Removing the outer layer of the chip will have a permanent effect on these circuit variables and again, it will change the output of the PUF circuit. Briefly, we can say that a well-built PUF device will be physically tamper-resilient up to its sensitivity level. Multiple designs have been proposed in the literature to realize PUFs. Optical PUFs were first introduced in [34]. Delay-based PUFs or more known as silicon PUFs were introduced in [10]. Coating PUFs were introduced in [40]. More recently, FPGA SRAM PUFs were introduced in [14]. Surface PUFs were proposed in [33,41] and further developed in [37]. In general, so far the usage of PUFs has focused on the fact that they are unclonable. In this paper we focus our attention to the delay-based PUF first introduced in [10].

A delay based PUF [10] is a $\{0,1\}^n \rightarrow \{0,1\}$ mapping, that takes an n-bit challenge (C) and produces a single bit output (R). The delay based PUF circuit consists of n stages of switching circuits as shown in Figure 1. Each switch has two input and two output bits in addition to a control bit. If the control bit of the switch is logical 0, the two inputs are directly passed to the outputs through a straight path. If on the other hand, the control bit to the switch is 1, the two input bits are switched before being passed as output bits. So based on the control bit of every switch, the two inputted signals will take one of two possible paths. In the switch-based delay PUF, there are n switches where the output of each switch is connected to the input of the following switch. At the end, the two output bits of the last switch are connected to a flip-flop, which is called the *arbiter*. The two inputs to the first of these switches are connected to each other, and then connection is sourced by a pulse generator.

The delay PUF can be described using the following linear model [10,15],

$$R = \mathrm{PUF}_Y(C) = \mathrm{sign}\left(\sum_{i=1}^{n}(-1)^{p_i} y_i + y_{n+1}\right) \quad . \tag{1}$$

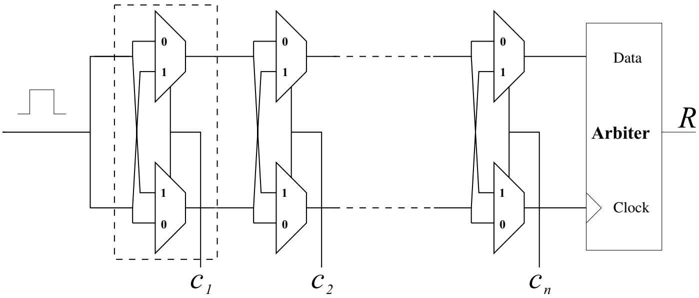


Fig. 1. A basic delay based PUF circuit

where $Y = [y_1 \ldots y_{n+1}]$ with y_i as the delay variation between the two signal paths in the i^{th} stage and y_{n+1} captures the setup time and the mismatch of the arbiter. Sign$(x) = 1$ if $x > 0$, and 0 if $x \leq 0$. $p_i = c_i \oplus c_{i+1} \oplus \ldots \oplus c_n$, where c_i is the i^{th} bit of C. Note that the relation between $P = [p_1 \ldots p_n]$ and $C = [c_1 \ldots c_n]$ can be described using the equation ($P = UC$). The strings C and P are represented as column vectors, U is the upper triangular matrix with all non-zero entries equal to 1 and the matrix multiplication is performed modulo 2. Equation 1 captures the ideal PUF response. However, due to race conditions which will sometimes cause the two signals inside the PUF paths to have very close delays, the output of the PUF will sometimes be random. We refer to these random outputs as *metastable* outputs. This happens with a certain probability depending on the sensitivity of the arbiter. For typical flip-flops a 64-bit PUF will have about 1 metastable output for every 1000 outputs [29].

The variables y_i capture the secret maintained in the hardware, and which cannot be measured directly. These variables are usually assumed to be independent with each following a normal distribution of mean 0 and some variance which can be assumed to be 1 without loss of generality [30]. We note here that the independence of these variables will highly depend on the manufacturing process. For example in an FPGA implementation it is much harder to get almost independent y_i variables. In an ASIC implementation the y_i variables seem to be closer to independence. However, to simulate an independent response one might average over multiple independent challenges. In this study, we will work under the assumption that the y_i variables are independent.

With the independence assumption one can derive the probability distribution of two inputs $C^{(1)}$ and $C^{(2)}$ producing the same PUF output over all possible outputs as follows,

$$\text{Prob}[\text{PUF}_Y(C^{(1)}) = \text{PUF}_Y(C^{(2)})] = 1 - \frac{2}{\pi}\arctan\left(\sqrt{\frac{d}{n+1-d}}\right). \quad (2)$$

where d is the Hamming distance between $P^{(1)} = UC^{(1)}$ and $P^{(2)} = UC^{(2)}$. For the full derivation the reader is referred to [16,15]. This relation carries the important fact that the correlations in the PUF output are solely dependent on the Hamming distance.

It is important to mention that due to the linear nature of the PUF circuit, given a number of challenge-response pairs (C, R), an attacker can use linear programming [35,32,10] to approximate for the unknown vector Y. To solve this problem we have one of two options. First, hide the output R such that it is not accessible to an adversary. Our second option is to use non-linearization techniques such as the feed-forward scheme presented in [10,29]. For simplicity we will assume that the output R is not available to an adversary. This only means that the attacker cannot read R, but he still can inject a fault. We will shortly see from the coming sections that this is quite a reasonable assumption.

Adversarial Fault Model: In this paper, we do not put a limit on the possible fault injection methods that can be used by the attacker. In other words, the adversary can utilize any attack mechanism on the device in order to successfully inject a fault. The injected fault on a specific part of the circuit manifests itself at the output as an erroneous result. Consequently, the error e becomes the difference between the expected output x and the observed output $\tilde{x}=x + e$. This error can be characterized either as logical or arithmetic depending on the functions implemented by the target device. If the target area of the device is mostly dominated by flip-flops, RAM, registers, etc. then using the logical model, where the error is expressed as an XOR operation ($\tilde{x}=x \oplus e$), is more appropriate. If on the other hand, the targeted region of the device is an arithmetic circuit, then it is more useful to use the arithmetic model where the error may be expressed as an addition with carry ($\tilde{x}=x + e \bmod 2^k$, where k is the data width). In this paper, we are mostly concerned with secure FSMs and key storage. As a result, it is more appropriate to use the logical fault model. It is also important to note that since the analysis conducted in this paper does not assume attacker's inability to read the existing data on the circuit before injecting a fault, overwriting and jamming errors can also be modeled as logical error additions. In addition, note that one additional assumption for the fault model of Section 5 is described by Assumption 1.

4 Securing Known-Path State Machines

In this section we address the security of state machines against adversarial fault attacks. We focus our attention to the state machines which are not dependent upon input variables. Such machines are quite common in cryptographic applications which typically contain a limited number of states and perform functions which require a long sequence of states. We now formally define the class of state machines which we address in this section.

Definition 1. *A known-path state machine, is a state machine in which state transitions are not dependent upon the external input. The state-sequence which the state machine goes through is known beforehand, and can be considered a property of the state machine.*

An example of an algorithm that can be implemented with a known-path state machine is the "always multiply right-to-left binary exponentiation" [17] which

Algorithm 1. Always multiply Right-to-Left Binary Exponentiation Algorithm [17]

```
Require: $x$, $e$=$(e_{t-1}, ..., e_0)_2$
Ensure: $y = x^e$
  $R_0 \leftarrow 1$                                        INIT
  $R_1 \leftarrow x$                                        LOAD
  for $i = 0$ upto $t - 1$ do
    $b = 1 - e_i$
    $R_b \leftarrow R_b^2$                                  SQUARE
    $R_b \leftarrow R_b \cdot R_{e_i}$                      MULTIPLY
  end for
  $y \leftarrow R_0$                                        RESULT
```

is shown above. The associated state diagram for this algorithm is shown in Figure 2 with a possible fault injection attack (indicated by the dotted line). As can be observed, once the start signal is received, the transitions will follow a specific pattern and are independent from any kind of input except i which is a predetermined value. Another example of a known-path state machine is the "Montgomery ladder exponentiation" which is commonly used for RSA signature generation [18]. The PUF based security mechanism which we describe in this section defines a generic approach to secure this class of state machines. We now rigorously define our approach and derive the probability of detecting an injected error.

Let F be a known-path state machine with m states. We refer to the known-path of states which the state machine goes through in a full operation as the *state-sequence* and we denote it by S_Ω. Here, Ω represents the encoded states in the state-sequence observed by F. We define f to be the encoding function for our states. The function f is also assumed to produce a binary output. Let n be the bit size of the output of f and k the number of states in Ω.

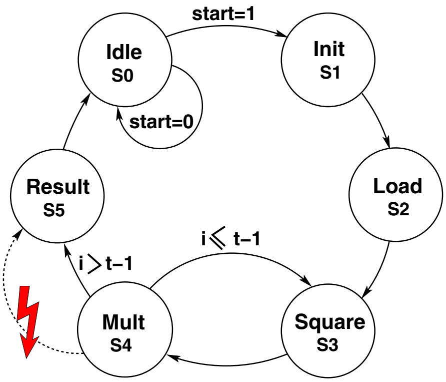


Fig. 2. State Diagram Representation of Left-to-Right Exponentiation Algorithm with Point of Attack

So for example, if F enters the state-sequence $s_1, s_2, s_3, s_4, s_3, s_4, s_5$ then $\Omega = [f(1), f(2), f(3), f(4), f(3), f(4), f(5)]$. If the encoding scheme is a simple binary encoding then $\Omega = [001, 010, 011, 100, 011, 100, 101]$, $n = 3$ and $k = 7$.

With the above definitions our proposal's main goal becomes to finger-print the state-sequence S_Ω. The way we achieve this is by adding a PUF circuit to the state machine logic. The straightforward idea of the scheme works as follows: at initialization time the circuit calculates the PUF output for each of the encodings in Ω. This output which we label ω is then securely stored for future comparisons. The i^{th} bit of ω which we label ω_i is calculated as

$$\omega_i = \mathrm{PUF}_Y\left(\Omega(i)\right)$$

where $\Omega(i)$ is the i^{th} entry in Ω. This equation means that an n-bit PUF needs to be utilized, and that ω will be a k-bit string. This straightforward approach captures the essential idea of the proposal. However, there are problems with the efficiency of this approach. One can imagine a simple state machine going into a 3-state loop for 1000 cycles. This would mean that ω contains at least 3000 bits of a repeating 3-bit sequence. If secure storage is not an issue, or if k is a small number, then the straightforward approach should suffice. However, when a state machine is expected to have large iterations over various cycles a different approach should be explored.

To solve this problem we take a deeper look into the state-sequences that we expect to observe in the known-path state machines. Such state-sequences can be broken into q different sub-sequences each of which contain p_i states and is repeated for c_i cycles. We can now rewrite the overall state-sequence as

$$\Omega = [\{\Omega_1\}^{c_1} \mid \{\Omega_2\}^{c_2} \mid \ldots \mid \{\Omega_q\}^{c_q}] \ ,$$

where $|$ stands for concatenation of sequences, and $\{\Omega_i\}^{c_i}$ indicates the repetition of the sub-sequence Ω_i for c_i times. Note that $k = c_1p_1 + c_2p_2 + \ldots + c_qp_q$. With the new labeling, we can see that the checksum does not need to be of length k. It should suffice for the checking circuit to store the constants c_i and p_i in

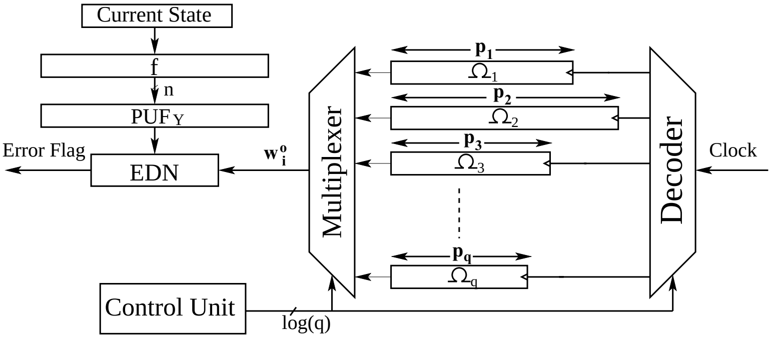


Fig. 3. PUF-based circuit for protecting FSMs

addition to the bits $\omega_{i,j} = \mathrm{PUF}_Y\ (\Omega_i(j))$ for $i = 1 \ldots q$ and $j = 1 \ldots c_i$. We can write

$$\omega = [\ \omega_1 \mid \omega_2 \mid\ \ldots\ \mid \omega_q\ ]\ ,$$

where ω_i will contain p_i bits. With only having to store the ω_i strings, the scheme will be efficient in terms of storage. We next explain how the checking circuit works.

As can be seen in Figure 3, the checking circuit stores each of the ω_i strings in a separate shift register with the output (the most significant bit) connected to the input (the least significant bit) of the register. The outputs of the shift registers are also connected to a multiplexer, whose $\log(q)$ select signals connected to a small control unit. The control unit's main task is to maintain a counter C which will indicate how far along the state-sequence is the state machine, therefore generating the select signals for the multiplexer. When $C \leq c_1p_1$ the first register's output is selected. Once the counter exceeds c_1p_1 the control unit will select the second register, and will maintain the same output until the counter reaches $c_1p_1 + c_2p_2$. The control unit will essentially chose the i^{th} register as long as $c_1p_1 + c_2p_2 + \ldots + c_{i-1}p_{i-1} < C \leq c_1p_1 + c_2p_2 + \ldots + c_ip_i$. The checking circuit continues in this fashion until the counter reaches $k = c_1p_1 + c_2p_2 + \ldots + c_qp_q$ at which point the counter resets to zero since the state machine will be back to its initial state. We will label the output of the multiplexer at i^{th} state of the state-sequence as ω_i^o. For the registers to generate the right output, the select signals produced by the control unit also need to be fed into a decoder which will produce the clock signals of the registers. The input of the decoder will be the master clock signal, and the outputs of the decoder will be connected to the clock inputs of the shift registers. Note that these signals will only be high when the corresponding register is being used, therefore causing the register to shift accordingly.

At every clock cycle the current check bit $x_i = \mathrm{PUF}_Y\ (f(S_i))$ is generated, where S_i is the current state of the state machine. The checking circuit will verify the condition $\omega_i^o = x_i$. Whenever this condition is violated the checking circuit can issue a signal to indicate a fault injection. We have mentioned earlier that the output of a PUF circuit will not be consistent for a certain percentage of the inputs. This percentage will set a tolerance threshold labeled L for the checking circuit. If the number of violations detected by the checking circuit is more than L, the checking circuit can signal an attack, therefore halting the circuits operation.

To calculate the probability of an attack actually being detected, we note that the PUF output is uniform. A fault injected by the attacker will change the current state, and will consequently change the following states. We label the states in the fault-free sequence S_Ω as ideal states, and we label the states which are different as a result of the fault injection as faulty states. The fault-free sequence and the new faulty sequence will have t different states, $t \leq k$. We are interested in calculating the probability of the new faulty states actually yielding a different PUF output than that of the ideal states. Equation 2 shows that when the Hamming distance between the two PUF inputs is about $n/2$, this probability is 0.5. With this in mind, we can choose the encoding function f and the size of its output n such that the encodings of any two states have a Hamming distance of $n/2$. Even more efficiently, if the encoding is assumed to be secret, the Hamming distance between the encoded state vectors would be averaged over all possible encodings, therefore also yielding an effective Hamming distance of $n/2$ between any two state vectors. In either case, the probability of a faulty state generating the same PUF output as an ideal state will effectively be 0.5. With these factors, we can expect the detection probability of an injected fault which causes a total of t state changes to be

$$P_t = 1 - 2^{-(t-L)} \quad .$$

Naturally, t is assumed to be larger than L since otherwise the detection probability would be zero. If the fault injected causes a small number of state changes t, this probability will not be sufficient to secure the system. Although it is expected that an injected fault will cause a large number of state changes, for completeness we next handle the case when t is small.

We propose two approaches to solve this problem. The first is to utilize a number of PUF circuits each of which storing a separate array of checksums. And the second is to use a single PUF but calculate the check bits for different encoding functions of the states. Essentially one can use a single encoding and then apply a permutation to generate a variant encoding. Whether we use d PUFs or we use d different encodings, in either case we will be adding d-levels of check bits. Regardless of which of the two approaches we use the detection probability of a fault causing t state changes will be

$$P_t = 1 - 2^{-(td-L)} \quad .$$

Using error control techniques for the PUF circuit such as majority voting [29], the error rate in the PUF output can be reduced to as low as 1 in 1000. This means that for state machine where $k < 1000$ states, the probability of error detection becomes $P_t = 1 - 2^{1-td}$. Naturally, this probability does not take into account the probability of inducing a change in the internal PUF parameters. Such a change will have an effect on the PUF output and will therefore increase the error detection probability.

In order to estimate the hardware overhead incurred by the proposed error detection mechanism, we carry out the following analysis. The number of flip-flops required for storing the checksums will be equivalent to the number of flip-flops used in the current state register. As mentioned earlier, the counter C constitute the main part of the control unit, which is also the case for the state machine. Therefore, we can argue that the size of the control unit and the checksum storage will be approximately the same as the state machine, implying a 100% overhead.

The encoding function f will typically have an output size which is on the order of the total number of states m. Consequently, we can assert that the function f will on average use about $2m$ combinational gates. The same applies to the PUF circuit which will also require about $2m$ gates. Finally, the size of the decoder and the multiplexer shown in Figure 3 is expected to be on the order of log(q) gates. Although q can be of any size, in a typical state machine q will not be larger than 2^m. Hence, the number of gates associated with the multiplexer and the decoder will be about $2m$. Adding these numbers results in a total gate count of $6m$. This is the same number of gates used by the current state register (Note that m flip-flops are approximately composed of $6m$ universal gates). In general, it is safe to assume that the current state register will consume approximately 50% of the total state machine area, which implies an area overhead of 50%.

As a result, the total area overhead introduced by the proposed error detection scheme will be approximately 150%, with a high error detection rate even against strong adversaries. When compared to the simpler error detection schemes such as Triple Modular Redundancy (TMR) and Quadruple Modular Redundancy (QMR) (which only replicate the existing hardware, implement the same function concurrently, and do a majority voting to check if an error has been injected), the proposed scheme provides a higher level of security even against advanced adversaries because an attacker can simply inject the same error to all replicas of the original hardware and mask the error in these detection schemes. The area overhead associated with these trivial detection mechanisms will be at least 200% for TMR and 300% for QMR which is also higher than the overhead of the proposed mechanism. As a comparison to a more advanced error detection scheme, the study conducted by Gaubatz et al. [13], which utilizes linear codes for error detection, reports an area overhead of more than 200%. However, their fault model assumes weak adversaries and the error detection scheme becomes vulnerable against strong attackers. It is important to note that, finite state machines usually constitute a very small part of the entire circuit. Therefore, although the reported area overhead might appear to be large, the effective increase in the overall area is reasonable. To sum up, PUF-based error detection mechanism discussed in this paper accomplishes a higher level of security with a reduced area overhead.

5 Key Integrity

In [7], Biham and Shamir extended the fault injection attacks to block-ciphers and reported that they can recover the full DES key of a tamper-proof encryptor by examining 50 to 200 cipher-texts. In their paper, they also described a method to break an unknown (unspecified) cryptosystem by utilizing the asymmetric behavior associated with the used memory device. Basically, their fault model assumes that the applied physical stress on the memory device, which contains the key bits, could only cause one to zero transitions[1]. Using this attack, the secret key can be obtained using at most $O(n^2)$ encryptions. Similarly, Kocar [23] reports a method to estimate the key bits of a cryptographic device by employing the charge shifting characteristic of EEPROM cell transistors after

[1] This one-way characteristic can also cause zero to one transitions depending on the asymmetry of technology used to fabricate the memory device.

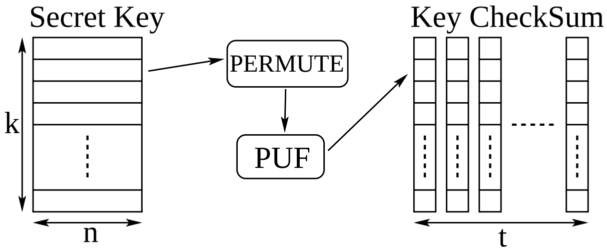


Fig. 4. Key integrity check using PUF

baking. In addition, in [3] authors describe an EEPROM modification attack where they can recover the DES key by overwriting the key bits one by one and checking for parity errors. Same authors, in [2], also discuss example attacks on PIC16C84 microcontroller and DS5000 security processor in which security bits can be reset by modifying the source voltages. The key overwrite attacks also constitute a crucial risk on smart cards where the key is stored inside the EEPROM. To summarize, fault injection attacks on secret keys stored on-chip memory pose a serious threat in many cryptographic hardware implementations.

In this section, we propose utilizing PUFs as a solution to this important problem. The main idea here is similar to that of the previous section. Basically, we generate a secret key fingerprint or checksum for the correct secret key using a PUF circuit. As outlined earlier, the checksums are assumed to be stored secretly while allowing fault injection. Regularly, the checking circuit can check and verify the integrity of the key. If the checksum value for the current key does not match the checksum value for the correct secret key, this can be interpreted as an error injection to the key. As a result, an error message can be issued and the secret data can be flushed or the device can be reseted to prevent any kind of secret leakage. This mechanism is briefly shown in Figure 4. This figure shows part of the memory which contains a secret key of size $k \times n$. Each row of this key block is labeled r_i and is treated as an input to the PUF circuit. If the rows are directly fed to the PUF circuit, an attacker can carefully choose his errors such that the Hamming distance between the actual variables (P) defined in Equation 1 is minimal. Recall that this would mean that the PUF output will not be able to detect the injected error. If the size of the checksum for each key row is a single bit, the error detection probability for an injected error would be 0.5 as the PUF can only provide an output of $\{0, 1\}$. In this case, the success rate for the attacker is considerably high. This is why we utilize a permutation block as shown in Figure 4.

The permutation block will essentially permute each input row r_i by a predetermined permutation ρ_j where $j = (1, \ldots, t)$ and $t < n$. Consequently, $\rho_j(r_i)$ is fed to the PUF in order to generate the (i, j) bit of the checksum. In short, for the secret key array S with size $k \times n$ and rows r_i, we calculate the (i, j) bit of the checksum S_w as

$$S_w(i, j) = \mathrm{PUF}_Y(\rho_j(r_i)) \tag{3}$$

where the ρ_j's are random permutations pre-chosen secretly and S_w is of size $k \times t$.

When this model is applied to secure the cryptographic devices against memory overwrite attacks, the robustness and security measure of the error detection scheme becomes a direct function of t, the number of the permutations used for each row. The probability of an error being detected is essentially the probability of an error changing the PUF output. However, we have seen in the previous section that the PUF output will sometimes be metastable. Therefore, we will again define an acceptable level of errors which will be a property of the system and which will not raise an alarm. Similar to the previous section we define this level as L. Now we can define the event for an error being detected. In particular, an error injected to row r_i will be detected provided that the following equation will not hold for more than L of the row's t checking bits.

$$\mathrm{PUF}_Y(\rho_j(r_i)) = \mathrm{PUF}_Y(\rho_j(r_i + e)) \tag{4}$$

where e indicates an error injected to the i^th row, e.g. bit flip of some memory cells. We now calculate this probability. Equation 4 is essentially the probability calculated in Equation 2. Therefore, we will again have to refer back to the Hamming distance between the ideal and the attacked PUF inputs. Because the permutations ρ_j are taken over all possible permutations, the attacker cannot control the effective location of his injected errors. To simplify the calculation we make the following assumption.

Assumption 1: *We assume an attacker model where the number of faults injected by the attacker is uniform over all possible number of faults.*

With Assumption 1 we can calculate the expected value of the Hamming distance between the P values of the original data and the faulty data when taken over all permutations to be equal to $n/2$. Going back to Equation 2 for this particular Hamming distance the probability for Equation 3 to hold will be 0.5. With this, the detection probability of an error becomes $1 - 2^{-(t-L)}$.

At this point, it is important to note the trade-off between the area overhead and security level of the suggested mechanism. As the number of permutations t for each row increases, the security of the device gets stronger because the error detection probability increases. However, the area overhead also increases linearly with t due to the checksum storage space. The optimal value for t is an application dependent issue and can be adjusted according to the required security level or allowed area overhead.

Note that one can use error correcting codes to address the integrity issue. However, such a solution would require substantially more hardware for decoding the code words. Moreover, the PUF circuit has built-in fault resilience due to its sensitive characteristics. Consequently, any kind of fault injection or perturbation of the hardware will modify the result of the PUFs. This brings an additional level of security to the proposed key integrity and protection scheme. In addition, the solution we present here views the separate memory rows as independent entities. It is an interesting problem to explore combinations of the rows and columns, which might improve the error detection probability. Finally, the model here assumes that the checksum is hidden secretly. If a designer wishes to relax this condition different PUF designs should then be explored.

6 Error Detection Network Security

Concurrent error detection (CED) is one of the most common solutions against active fault attacks. The basic idea in these schemes is to calculate the expected result using predictor circuits in parallel to the main hardware branch, and compare if the predicted value of the result matches the actual value calculated in the main branch. A good overview along with elaborate examples about this mechanism can be found in [11]. An error signal is issued when these two paths do not produce agreeing results. This comparison along with the error signal generation are conducted by the error detection network (EDN) modules.

While the attackers are always assumed to target the main or predictor branch of a cryptographic device, the EDN network which is in fact the weakest link in the design is always assumed to be completely fault resilient. An attacker can deactivate the EDN by preventing an error signal from being issued or by just simply attacking the inputs of the EDN. Therefore, totally disabling/bypassing the error detection mechanism. To tackle this problem, we propose utilizing PUF structures to design secure and fault tolerant EDN blocks.

The suggested PUF based EDN mechanism is shown in Figure 5. Basically, the results coming from the main and predictor branches of the computation are first XOR-ed together. In the absence of an injected fault, the result will be the zero vector. The circuit starts by producing a fingerprint of the PUF response to an all zero bit challenge right before the circuit is deployed. This fingerprint is stored as the checksum bit. Throughout the circuit's operation, the XOR-ed results are continuously permuted and fed into the PUF circuit. This permutation block implements the same functionality as in Section 5. In the absence of an

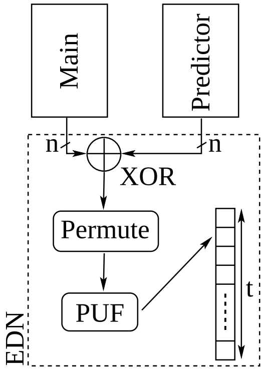


Fig. 5. PUF based EDN

error, the permutation will have no effect on the all zero vector, Therefore the output should always match the stored checksum bit. However, when an error is injected the output of the XOR will not be the all zero vector. This will cause the permutations to generate different challenge vectors which will consequently produce PUF outputs which are different from the checksum bit.

When the circuit detects a mismatch between the output of the PUF and the checksum bit an injected fault is assumed and an error signal can be issued. Similar to the analysis conducted in the previous two sections, the error detection capability of the EDN is dependent upon the number of applied permutations t, and can be formulated as $1 - 2^{-(t-L)}$. As in the previous sections, L here is the acceptable threshold of errors in the PUF response. The trade-off between the security and area overhead discussed in Section 5 also exists in this PUF based EDN methodology too.

Note that any attempt by the attacker to modify the voltage levels of the wires located inside the EDN will affect and change the result of the PUF due to its high sensitivity. This intrinsic tamper resistance of the PUF circuit acts as assurance against fault attacks targeting the EDN.

7 Conclusion

In this paper, we explored the integration of PUFs into the building blocks of finite state machines to provide security. In particular, we addressed the security of state-transitions (next-state logic) against fault-injection attacks, the integrity of secret information, and finally fault-resilience in error detection networks. We proposed PUF-based architectures for the security of these modules in a control unit, and showed that the probability of error detection is high. More importantly, the solution we propose provides security on the physical level as well as the logical level. Even if the adversary can find the appropriate fault to inject, there will still be a good chance of being detected by the change in the PUF behavior. The designs we propose in this paper are described from a higher level and therefore are far from being final solutions ready for implementation. Rather, these solutions are mainly intended to open a new door for research in the area of unclonable protection of FSMs. Such mechanisms can provide strong error detection with a relatively low hardware overhead. Our work here is a first step in this direction.

References

1. Agmon, S.: The relaxation method for linear inequalities. Canadian J. of Mathematics, 382–392 (1964)
2. Anderson, R., Kuhn, M.: Tamper resistance: a cautionary note. In: WOEC 1996: Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce, Berkeley, CA, USA, p. 1. USENIX Association (1996)
3. Anderson, R.J., Kuhn, M.G.: Low cost attacks on tamper resistant devices. In: Proceedings of the 5th International Workshop on Security Protocols, London, UK, pp. 125–136. Springer, London (1998)

4. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerer's apprentice guide to fault attacks. Proceedings of the IEEE 94, 370–382 (2006)
5. Berg, M.: Fault tolerant design techniques for asynchronous single event upsets within synchronous finite state machine architectures. In: 7th International Military and Aerospace Programmable Logic Devices (MAPLD) Conference. NASA (September 2004)
6. Bertoni, G., Breveglieri, L., Koren, I., Maistri, P., Piuri, V.: Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. IEEE Transactions on Computers 52(4), 492–505 (2003)
7. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
8. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
9. Cunningham, P., Anderson, R., Mullins, R., Taylor, G., Moore, S.: Improving Smart Card Security Using Self-Timed Circuits. In: Proceedings of the 8th international Symposium on Asynchronus Circuits and Systems, ASYNC, p. 211. IEEE Computer Society, Washington (2002)
10. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Delay-based Circuit Authentication and Applications. In: Proceedings of the 2003 ACM Symposium on Applied Computing, pp. 294–301 (2003)
11. Gaubatz, G., Sunar, B., Karpovsky, M.G.: Non-linear residue codes for robust public-key arithmetic. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 173–184. Springer, Heidelberg (2006)
12. Gaubatz, G., Sunar, B.: Robust finite field arithmetic for fault-tolerant public-key cryptography. In: Breveglieri, L., Koren, I. (eds.) 2nd Workshop on Fault Diagnosis and Tolerance in Cryptography - FDTC 2005 (September 2005)
13. Gaubatz, G., Sunar, B., Savas, E.: Sequential Circuit Design for Embedded Cryptographic Applications Resilient to Adversarial Faults. IEEE Transactions on Computers 57(1), 126–138 (2008)
14. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: FPGA intrinsic pUFs and their use for IP protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
15. Hammouri, G., Ozturk, E., Sunar, B.: A Tamper-Proof, Lightweight and Secure Authentication Scheme (under review)
16. Hammouri, G., Sunar, B.: PUF-HB: A Tamper-Resilient HB Based Authentication Protocol. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 346–365. Springer, Heidelberg (2008)
17. Joye, M.: Highly Regular Right-to-Left Algorithms for Scalar Multiplication. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, p. 135. Springer, Heidelberg (2007)
18. Joye, M., Yen, S.M.: The Montgomery Powering Ladder. In: Cryptographic Hardware and Embedded Systems-Ches 2002: 4th International Workshop, Redwood Shores, CA, USA: Revised Papers, August 13-15 (2002)
19. Karpovsky, M., Kulikowski, K.J., Taubin, A.: Differential fault analysis attack resistant architectures for the advanced encryption standard. In: Proc. World Computing Congress (2004)

20. Karpovsky, M., Kulikowski, K.J., Taubin, A.: Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard. In: DSN 2004: Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN 2004), Washington, DC, USA, p. 93. IEEE Computer Society Press, Los Alamitos (2004)
21. Karpovsky, M., Taubin, A.: A new class of nonlinear systematic error detecting codes. IEEE Trans. Info. Theory 50(8), 1818–1820 (2004)
22. Karri, R., Wu, K., Mishra, P., Kim, Y.: Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers. IEEE Transactions on computer-aided design of integrated circuits and systems 21(12), 1509–1517 (2002)
23. Kocar, O.: Estimation of keys stored in cmos cryptographic device after baking by using the charge shift. Cryptology ePrint Archive, Report 2007/134 (2007), `http://eprint.iacr.org/`
24. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Advances in Cryptology-Crypto 1999: 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999 Proceedings (1999)
25. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
26. Krasniewski, A.: Concurrent error detection in sequential circuits implemented using fpgas with embedded memory blocks. In: Proceedings of the 10th IEEE International On-Line Testing Symposium (IOLTS 2004) (2004)
27. Kulikowski, K.J., Karpovsky, M., Taubin, A.: Robust codes for fault attack resistant cryptographic hardware. In: Workshop on Fault Diagnosis and Tolerance in Cryptography 2005 (FTDC 2005) (2005)
28. Kulikowski, K.J., Karpovsky, M., Taubin, A.: Fault attack resistant cryptographic hardware with uniform error detection. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 185–195. Springer, Heidelberg (2006)
29. Lee, J.W., Daihyun, L., Gassend, B., Suhamd, G.E., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication applications. In: Symposium of VLSI Circuits, pp. 176–179 (2004)
30. Lim, D., Lee, J.W., Gassend, B., Edward Suh, G., van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. IEEE Trans. VLSI Syst. 13(10), 1200–1205 (2005)
31. Naccache, D.: Finding faults. IEEE Security and Privacy 3(5), 61–65 (2005)
32. Ozturk, E., Hammouri, G., Sunar, B.: Towards robust low cost authentication for pervasive devices. In: PERCOM 2008: Proceedings of the Sixth IEEE International Conference on Pervasive Computing and Communications (2008)
33. Posch, R.: Protecting Devices by Active Coating. Journal of Universal Computer Science 4(7), 652–668 (1998)
34. Ravikanth, P.S.: Physical One-Way Functions. PhD thesis, Massachusetts Institute Of Technology (2001)
35. Roos, C., Terlaky, T., Vial, J.-P.: Interior Point Methods for Linear Optimization, 2nd edn. Springer, Heidelberg (2005)
36. Schmidt, J.M., Hutter, M.: Optical and em fault-attacks on crt-based rsa: Concrete results. In: Austrochip 2007: Proceedings of the 15th Austrian Workshop on Microelectronics (2007)
37. Skoric, B., Maubach, S., Kevenaar, T., Tuyls, P.: Information-theoretic Analysis of Coating PUFs. Cryptology ePrint Archive, Report 2006/101 (2006)

38. Skorobogatov, S.P., Anderson, R.J.: Optical Fault Induction Attacks. In: Cryptographic Hardware and Embedded Systems-Ches 2002: 4th International Workshop, Redwood Shores, CA, USA, Revised Papers, August 13-15 (2002)
39. Sokolov, D., Murphy, J., Bystrov, A.V., Yakovlev, A.: Design and Analysis of Dual-Rail Circuits for Security Applications. IEEE Transactions on Computers 54(4), 449–460 (2005)
40. Tuyls, P., Schrijen, G.-J., Škorić, B., van Geloven, J., Verhaegh, N., Wolters, R.: Read-proof hardware from protective coatings. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 369–383. Springer, Heidelberg (2006)
41. Tuyls, P., Skoric, B.: Secret Key Generation from Classical Physics: Physical Uncloneable Functions. In: Mukherjee, S., Aarts, E., Roovers, R., Widdershoven, F., Ouwerkerk, M. (eds.) AmIware: Hardware Technology Drivers of Ambient Intelligence. Philips Research Book Series, vol. 5. Springer, Heidelberg (2006)
42. Waddle, J., Wagner, D.: Fault Attacks on Dual-Rail Encoded Systems. In: Proceedings of the 21st Annual Computer Security Applications Conference, pp. 483–494. ACSAC. IEEE Computer Society, Washington (2005), http://dx.doi.org/10.1109/CSAC.2005.25

Partition *vs.* Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices[⋆]

François-Xavier Standaert[1,⋆⋆], Benedikt Gierlichs[2], and Ingrid Verbauwhede[2]

[1] UCL Crypto Group, Université catholique de Louvain, B-1348 Louvain-la-Neuve
[2] K.U. Leuven, ESAT/SCD-COSIC and IBBT
fstandae@uclouvain.be,{bgierlic,iverbauw}@esat.kuleuven.be

Abstract. Given a cryptographic device leaking side-channel information, different distinguishers can be considered to turn this information into a successful key recovery. Such proposals include *e.g.* Kocher's original DPA, correlation and template attacks. A natural question is therefore to determine the most efficient approach. In the last years, various experiments have confirmed the effectiveness of side-channel attacks. Unfortunately, these attacks were generally conducted against different devices and using different distinguishers. Additionally, the public literature contains more proofs of concept (*e.g.* single experiments exhibiting a key recovery) than sound statistical evaluations using unified criteria. As a consequence, this paper proposes a fair experimental comparison of different statistical tests for side-channel attacks. This analysis allows us to revisit a number of known intuitions and to put forward new ones. It also provides a methodological contribution to the analysis of physically observable cryptography. Additionally, we suggest an informal classification of side-channel distinguishers that underlines the similarities between different attacks. We finally describe a new (but highly inspired from previous ones) statistical test to exploit side-channel leakages.

1 Introduction

Showing the effectiveness of a side-channel attack usually starts with a proof of concept. An adversary selects a leaking device of his choice and exploits the available physical information with a distinguisher. Recovering a cryptographic key (*e.g.* from a block cipher) is then used to argue that the attack works. But as for any experimental observation, a proof of concept has to be followed by a sound

[⋆] Work supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State, by FWO projects G.0475.05 and G.0300.07, by the European Commission under grant agreement 216646 ECRYPT NoE phase II, and by K.U. Leuven-BOF.

[⋆⋆] Associate researcher of the Belgian Fund for Scientific Research (F.R.S.-FNRS).

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 253–267, 2009.

statistical analysis. For example, one can compute the number of queries to a target cryptographic device required to recover a key with high confidence. Even better, one can compute the success rate or guessing entropy of a side-channel adversary in function of this number of queries. Various experimental and theoretical works describing different types of side-channel attacks can be found in the open literature. However, they are generally conducted independently and therefore are not straightforward to compare. Hence, we believe that a unifying analysis of actual distinguishers is important to enhance our understanding.

The contribution of this paper is threefold. First, we propose an informal classification of side-channel distinguishers into two categories, namely partition and comparison distinguishers. Second, we describe an alternative statistical test for partitioning attacks based on the sample variance. Most importantly and following the framework in [18], we provide a fair empirical comparison of statistical tests for univariate side-channel distinguishers against two unprotected software implementations of the AES Rijndael [5]. It includes Kocher's original Differential Power Analysis (DPA) [10], Pearson's correlation coefficient [2] and template attacks [3] as well as the recently proposed Mutual Information Analysis (MIA) [6]. Our results demonstrate the wide variety of flexibility *vs.* efficiency tradeoffs that can be obtained from different distinguishers and the effectiveness of template attacks when exploiting good leakage models. Additionally, they illustrate that claims on the efficiency of a given attack highly depend on an adversarial or implementation context. These results suggest that any new proposal of side-channel attack should come with a similar evaluation in order to show how these new proposals behave compared to former attacks. Note that we do not claim the novelty of our conclusions. As a matter of fact, several works already discussed similar comparison goals (see *e.g.* [4,11]). However, we believe that the approach, metrics and number of experiments proposed in this paper allow improving the evaluation of side-channel attacks and pinpointing their limitations.

The rest of the paper is structured as follows. Sections 2 and 3 describe our target implementations and the side-channel adversaries that will be used in our comparisons. Section 4 proposes an informal classification for side-channel distinguishers and details the different statistical tests that we will consider in our comparisons. It additionally describes a variance-based statistical test for side-channel attacks. Section 5 defines our evaluation metrics. Section 6 discusses the limitations and features of our classification and methodology. The description of our experiments and results are in Section 7 and conclusions are in Section 8.

2 Target Implementations

We target two implementations of the AES-128 in two 8-bit RISC-based microcontrollers. In the first setup, we used a PIC 16F877 running at a frequency around 4 MHz. In the second setup, we used an Atmel ATmega163 in a smart card body, clocked at 3.57 MHz. In both cases, our attacks aim to recover the first 8 bits of the block cipher master key k. We denote this part of the key as a key class s. The physical leakages were acquired with a digital oscillo-

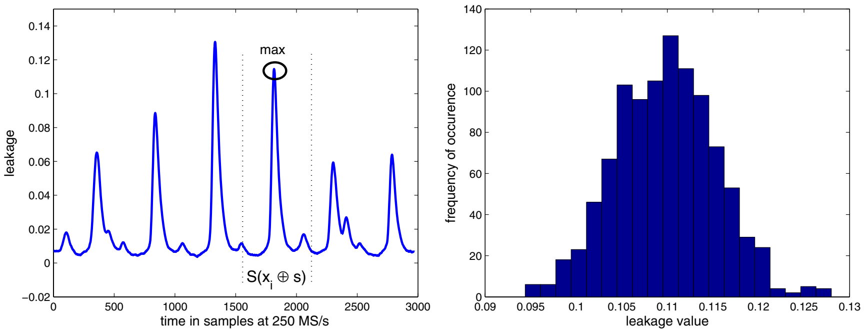


Fig. 1. Left: exemplary PIC power trace and selection of the meaningful samples. Right: Histogram for the statistical distribution of the electrical noise in the PIC leakages.

scope, respectively a Tektronix 7140 with a 1 GHz bandwidth running at a 250 MS/s sampling rate for the PIC and an Agilent Infinium 54832D with a 1GHz bandwidth running at a 200 MS/s sampling rate for the Atmel. We note that although the PIC and Atmel devices seem to be very similar, their leakages are substantially different, as will be confirmed in the following sections.

3 Side-Channel Adversary

The present analysis aims to compare different statistical tests for side-channel attacks. But statistical tests are only a part of a side-channel adversary. A fair comparison between such tests consequently requires that the other parts of the adversary are identical. Following the descriptions and definitions that are introduced in [18], it means that all our attacks *against a given target device* exploit the same input generation algorithm, the same leakage function and the same leakage reduction mapping. In addition, in order to illustrate the wide variety of tradeoffs that can be considered for such adversaries, we used slightly different settings for our two devices (*i.e.* PIC and Atmel). Specifically:

- We fed both devices with uniformly distributed random (known) plaintexts.
- We provided the statistical tests with the same sets of leakages, monitored with two similar measurement setups (*i.e.* one setup by target device).
- Only univariate side-channel adversaries were considered in our comparison.
  - For the PIC device, we used a reduction mapping R that extracts the leakage samples corresponding to the computation of $\mathsf{S}(x_i \oplus s)$ in the traces (this clock cycle is illustrated in the left part of Figure 1), where S is the 8-bit substitution box of the AES and x_i the first 8 bits of the plaintext. Then, only the maximum value of this clock cycle was selected. Hence, to each input plaintext vector $\mathbf{x}_q = [x_1, x_2, \ldots, x_q]$ corresponds a q-sample leakage vector $\mathsf{R}(\mathbf{l}_q) = [\mathsf{R}(l_1), \mathsf{R}(l_2), \ldots, \mathsf{R}(l_q)]$.
  - For the Atmel implementation, all the samples corresponding to the computation of the first AES round were first tested independently in

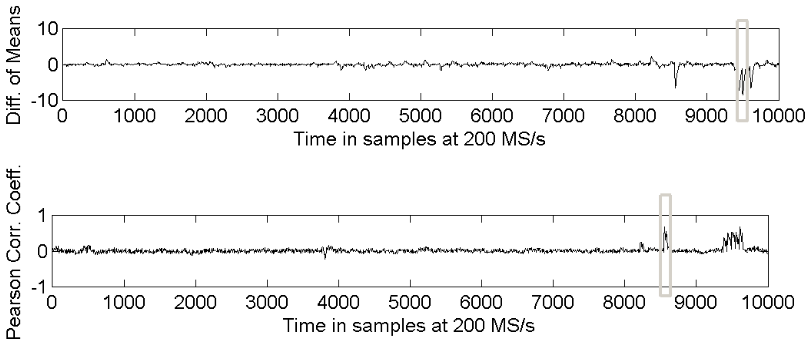


Fig. 2. Selection of the meaningful time samples for the Atmel

order to determine the sample giving rise to the best results, for each statistical test. Then, the actual analysis was only applied to this sample. Figure 2 illustrates this selection of time samples for two statistical tests to be defined in the next section, namely the Difference of Means (DoM) and Pearson's correlation coefficient. In other words, we used a different reduction mapping for each of the statistical tests in this case.

We mention that these choices are arbitrary: the only goal is to provide each statistical test with comparable inputs, for each target device. They also correspond to different (more or less realistic) attack scenarios. For the PIC device, we assume that one knows which sample to target: it considerably reduces the attack's time and memory complexities. But the selected time sample may not be optimal for a given attack. For the Atmel, no such assumption is made.

4 Classification of Distinguishers

In this section, we propose to informally classify the possible side-channel distinguishers as partition-based or comparison-based. More specifically:

- In a partition-based attack and for each key class candidate s^*, the adversary defines a partition of the leakages according to a function of the input plaintexts and key candidates. We denote such partitions as: $\mathsf{P}(s^*, \mathbf{v}_{s^*}^q)$, where $\mathbf{v}_{s^*}^q = \mathsf{V}(s^*, \mathbf{x}_q)$ are some (key-dependent) values in the implementation that are targeted by the adversary. For example, the S-box output $\mathsf{S}(x_i \oplus s^*)$ is a usual target. Then, a statistical test is used to check which partition is the most meaningful with respect to the real physical leakages. We denote this test as $\mathsf{T}(\mathsf{P}(s^*, \mathbf{v}_{s^*}^q), \mathsf{R}(\mathbf{l}_q))$. For example, Kocher's original DPA [10] partitions the leakages according to one bit in the implementation.
- In comparison-based attacks, the adversary models a part/function of the actual leakage emitted by the target device, for each key class candidate s^*. Depending on the attacks, the model can be the approximated probability density function of a reduced set of leakage samples denoted: $\mathsf{M}(s^*, \mathsf{R}(\mathbf{l}_q)) = \hat{\Pr}[s^* | \mathsf{R}(\mathbf{l}_q)]$, as when using templates [3]. Or the model is a deterministic

function (*e.g.* the Hamming weight) of some values in the implementation: $\mathsf{M}(s^*, \mathbf{v}_{s^*}^q)$, as in correlation attacks [2]. Then, a statistical test is used to compare each model $\mathsf{M}(s^*, .)$ with the actual leakages. Similarly to partitioning attacks, we denote this test as $\mathsf{T}(\mathsf{M}(s^*, .), \mathsf{R}(\mathbf{l}_q))$.

We note that the previous classification is purely informal in the sense that it does not relate to the capabilities of an adversary but to similarities between the way the different attacks are performed in practice. As the next sections will underline, it is only purposed to clarify the description of different statistical tests. As a matter of fact, one can partition according to (or model the leakage of) both single bits and multiple bits in an implementation. In both partition and comparison attacks, the expectation is that only the correct key class candidate will lead to a meaningful partition or good prediction of the actual leakages. Hence, for the two types of attacks, the knowledge of reasonable assumptions on the leakages generally improves the efficiency of the resulting key recovery. With this respect, the choice of the internal value used to build the partitions or models highly matters too. For example, one could use the AES S-box inputs $x_i \oplus s^*$ or outputs $\mathsf{S}(x_i \oplus s^*)$ for this purpose. But using the outputs generally gives rise to better attack results because the of the S-box non-linearity [14].

4.1 Statistical Tests for Partition Distinguishers

In a partition attack, for each key class candidate s^* the adversary essentially divides the leakages in several sets and stores them in the vectors $\mathbf{p}_{s^*}^1$, $\mathbf{p}_{s^*}^2$, $\ldots$, $\mathbf{p}_{s^*}^n$. These sets are built according to a *hypothetical* function of the internal values targeted by the adversary that we denote as H. It directly yields a variable $\mathbf{h}_{s^*}^q = \mathsf{H}(\mathbf{v}_{s^*}^q)$. In general, H can be any surjective function from the target values space $\mathcal{V}$ to a hypothetical leakage space $\mathcal{H}$. Examples of hypothetical leakages that can be used to partition a 16-element leakage vector $\mathsf{R}(\mathbf{l}_{16})$ include:

- a single bit of the target values (*i.e.* $n = 2$),
- two bits of the target values (*i.e.* $n = 4$),
- the Hamming weight of 4 bits of the target values (*i.e.* $n = 5$).

Such partitions are illustrated in Table 1 in which the indices of the $\mathsf{R}(l_i)$ values correspond to the input plaintexts $[x_1, \ldots, x_{16}]$. In a 1-bit partition, the 16 leakage values are stored in the vector $\mathbf{p}_{s^*}^1$ if the corresponding hypothetical leakage (*e.g.* one bit of $\mathsf{S}(x_i \oplus s^*)$) equals 0 and stored in $\mathbf{p}_{s^*}^2$ otherwise. As a result, we have one partition per key class candidate s^* and n vectors $\mathbf{p}_{s^*}^i$ per partition.

Kocher's DoM test. The first proposal for checking the relevance of a leakage partition is the difference of means test that was initially introduced in [10] and more carefully detailed in [12]. In this proposal and for each key class candidate s^*, the adversary only considers two vectors from each partition, respectively

Table 1. Examples of 1-bit, 2-bit and Hamming weight partitions

$\mathbf{p}^1_{s*}$	$\mathbf{p}^2_{s*}$
$\mathsf{R}(l_1)$	$\mathsf{R}(l_2)$
$\mathsf{R}(l_3)$	$\mathsf{R}(l_5)$
$\mathsf{R}(l_4)$	$\mathsf{R}(l_7)$
$\mathsf{R}(l_6)$	$\mathsf{R}(l_8)$
$\mathsf{R}(l_{10})$	$\mathsf{R}(l_9)$
$\mathsf{R}(l_{12})$	$\mathsf{R}(l_{11})$
$\mathsf{R}(l_{14})$	$\mathsf{R}(l_{13})$
$\mathsf{R}(l_{15})$	$\mathsf{R}(l_{16})$

$\mathbf{p}^1_{s*}$	$\mathbf{p}^2_{s*}$	$\mathbf{p}^3_{s*}$	$\mathbf{p}^4_{s*}$
$\mathsf{R}(l_3)$	$\mathsf{R}(l_1)$	$\mathsf{R}(l_5)$	$\mathsf{R}(l_6)$
$\mathsf{R}(l_4)$	$\mathsf{R}(l_2)$	$\mathsf{R}(l_9)$	$\mathsf{R}(l_7)$
$\mathsf{R}(l_{11})$	$\mathsf{R}(l_{10})$	$\mathsf{R}(l_8)$	$\mathsf{R}(l_{12})$
$\mathsf{R}(l_{15})$	$\mathsf{R}(l_{14})$	$\mathsf{R}(l_{16})$	$\mathsf{R}(l_{13})$

$\mathbf{p}^1_{s*}$	$\mathbf{p}^2_{s*}$	$\mathbf{p}^3_{s*}$	$\mathbf{p}^4_{s*}$	$\mathbf{p}^5_{s*}$
$\mathsf{R}(l_5)$	$\mathsf{R}(l_2)$	$\mathsf{R}(l_1)$	$\mathsf{R}(l_3)$	$\mathsf{R}(l_{14})$
	$\mathsf{R}(l_7)$	$\mathsf{R}(l_4)$	$\mathsf{R}(l_6)$	
	$\mathsf{R}(l_9)$	$\mathsf{R}(l_8)$	$\mathsf{R}(l_{12})$	
	$\mathsf{R}(l_{16})$	$\mathsf{R}(l_{10})$	$\mathsf{R}(l_{13})$	
		$\mathsf{R}(l_{11})$		
		$\mathsf{R}(l_{15})$		

denoted as $\mathbf{p}^A_{s^*}$ and $\mathbf{p}^B_{s^*}$. Applying such a difference of means test simply means that the adversary computes the difference between the sample means[1]:

$$\Delta_{s^*} = \hat{\mathbf{E}}(\mathbf{p}^A_{s^*}) - \hat{\mathbf{E}}(\mathbf{p}^B_{s^*}) \tag{1}$$

In single-bit attacks as when using the 1-bit partition in Table 1, the vectors $\mathbf{p}^A_{s^*}$ and $\mathbf{p}^B_{s^*}$ correspond to the two only columns of the partition. In multiple-bit attacks as when using the 2-bit partition in Table 1, the vectors $\mathbf{p}^A_{s^*}$ and $\mathbf{p}^B_{s^*}$ either correspond to two columns (out of several ones) in the partition, as in "all-or-nothing" multiple-bit attacks or they correspond to two combinations of columns in the partition, as in "generalized" multiple-bit attacks (see [12]). Note that "all-or-nothing" attacks have the drawback that several leakage samples are not exploited (*i.e.* corresponding to the unexploited columns of the partition). Note also that the best selection of the (combination of) columns requires to make assumptions about the leakages. For example, "all-or-nothing" attacks implicitly assume that the behavior of several bits is the same so that "all-or-nothing" partitions yield the largest Δ_s. As a result of the attack, the adversary obtains a vector $\mathbf{g}_q$ with the key candidates rated according to the test result, the most likely key corresponding to the highest absolute value for Δ_{s^*}.

MIA. Another proposal for exploiting a leakage partition in a more generic way than using a difference of means test has been described in [6]. It notably aims to exploit all the samples in a multiple-bit partition, without making any assumption on the leakage model. For this purpose, the adversary attempts to approximate the mutual information between the hypothetical leakages $\mathbf{h}^q_{s^*}$ and the actual leakages $\mathsf{R}(\mathbf{l}_q)$. For each vector $\mathbf{p}^i_{s^*}$, he first builds histograms in order to evaluate the joint distribution $\hat{\Pr}[\mathsf{R}(\mathbf{L}_q), \mathbf{H}^q_{s^*}]$ and the marginal distributions $\hat{\Pr}[\mathsf{R}(\mathbf{L}_q)]$ and $\hat{\Pr}[\mathbf{H}^q_{s^*}]$, for each key class candidate. Then, he estimates:

$$\hat{\mathrm{I}}(\mathsf{R}(\mathbf{L}_q); \mathbf{H}^q_{s^*}) = \hat{\mathrm{H}}[\mathsf{R}(\mathbf{L}_q)] + \hat{\mathrm{H}}(\mathbf{H}^q_{s^*}) - \hat{\mathrm{H}}[\mathsf{R}(\mathbf{L}_q), \mathbf{H}^q_{s^*}]$$

[1] In statistical textbooks, Difference of Means tests usually refer to more complex hypothesis tests. We use this simple version for illustration because it was extensively used in the cryptographic hardware community.

As in a difference of means test, the adversary obtains a vector $\mathbf{g}_q$ containing the key candidates rated according to the test result, the most likely key corresponding to the largest value for the mutual information.

4.2 Statistical Tests for Comparison Distinguishers

Pearson's correlation coefficient. In a correlation attack, the adversary essentially predicts a part/function of the leakage in the target device, for each key class candidate s^*. As a result, he obtains q-element vectors $\mathbf{m}^q_{s^*} = \mathsf{M}(s^*, \mathbf{v}^q_{s^*})$. For example, if a device is known to follow the Hamming weight leakage model, the vector typically contains the Hamming weights of the values $\mathsf{S}(x_i \oplus s^*)$. Since the reduced leakage vector $\mathsf{R}(\mathbf{l}_q)$ also contains q elements, the test can estimate the correlation between these two vectors, *e.g.* using Pearson's coefficient:

$$\rho_{s^*} = \frac{\sum_{i=1}^{q}(\mathsf{R}(l_i) - \hat{\mathbf{E}}(\mathsf{R}(\mathbf{l}_q))) \cdot (m^i_{s^*} - \hat{\mathbf{E}}(\mathbf{m}^q_{s^*}))}{\sqrt{\sum_{i=1}^{q}(\mathsf{R}(l_i) - \hat{\mathbf{E}}(\mathsf{R}(\mathbf{l}_q)))^2 \cdot \sum_{i=1}^{q}(m^i_{s^*} - \hat{\mathbf{E}}(\mathbf{m}^q_{s^*}))^2}} \tag{2}$$

Again, the adversary obtains a vector $\mathbf{g}_q$ with the key candidates rated according to the test result, the most likely key corresponding to the highest correlation.

Bayesian analysis. In template attacks, the adversary takes advantage of a probabilistic model for the leakages. He exploits an estimation of the conditional probabilities $\Pr[\mathsf{R}(\mathbf{l}_q)|s]$. From such an estimation, a straightforward strategy is to apply Bayes theorem and to select the keys according to their likelihood:

$$\lambda_{s^*} = \hat{\Pr}[s^*|\mathsf{R}(\mathbf{l}_q)] \tag{3}$$

It yields the same key candidate vector $\mathbf{g}_q$ as in the previous examples. Note that these attacks correspond to a stronger adversarial context than the other statistical tests in this section and require an estimation of the leakage probability distribution (*i.e.* to build templates). They should therefore be seen as a limit of what a side-channel adversary can achieve. We also note that in our simple context, the construction of templates was assumed to be unbounded[2]. But in more challenging scenarios, *i.e.* if the construction of templates is bounded, the use of stochastic models can be necessary for this purpose [15].

[2] Following [3], we assumed the leakages to be drawn from a normal distribution:

$$\mathcal{N}(\mathsf{R}(l_i)|\mu^i_s, \sigma^i_s) = \frac{1}{\sigma^i_s\sqrt{2\pi}} \exp\frac{-(\mathsf{R}(l_i) - \mu^i_s)^2}{2{\sigma^i_s}^2}, \tag{4}$$

in which the means μ^i_s and standard deviations σ^i_s specify completely the noise associated to each key class s. In practice, these means and standard deviations were estimated during a preliminary profiling step in which the adversary characterizes the target device (we constructed one template for each value of $\mathsf{S}(s \oplus x_i)$). That is, the probabilities $\Pr[s^*|\mathsf{R}(l_i)]$ are approximated in our attacks using Bayes theorem and the estimated Gaussian distribution $\hat{\Pr}[\mathsf{R}(l_i)|s^*] = \mathcal{N}(\mathsf{R}(l_i)|\hat{\mu}^i_{s^*}, \hat{\sigma}^i_{s^*})$, where $\hat{\mu}^i_{s^*}$ and $\hat{\sigma}^i_{s^*}$ respectively denote the sample mean and variance for a given leakage sample.

4.3 An Alternative Partition Distinguisher Using a Variance Test

The previous section described a number of statistical tests to evaluate the quality of a leakage model or partition. Of course, this list is not exhaustive: various other approaches have been and could be proposed. In this section, we suggest that under the common hypothesis of Gaussian noise in the physical leakages (confirmed in Figure 1 for the PIC), one can propose an alternative to the mutual information distinguisher [6]. Indeed, since in this context, the entropy of a good partition only depends on its variance, one can save the construction of histograms. Such a variance test can be described as follows. Let us denote the sample variance of the leakage and partition vectors as $\hat{\sigma}^2(\mathsf{R}(\mathbf{l}_q))$ and $\hat{\sigma}^2(\mathbf{p}^i_{s^*})$. From those variances, we compute the following statistic for each partition:

$$\sigma^2_{s^*} = \frac{\hat{\sigma}^2(\mathsf{R}(\mathbf{l}_q))}{\sum_{i=1}^{n} \frac{\#(\mathbf{p}^i_{s^*})}{q} \cdot \hat{\sigma}^2(\mathbf{p}^i_{s^*})} \tag{5}$$

where $\#(\mathbf{p}^i_{s^*})$ denotes the number of elements in a vector $\mathbf{p}^i_{s^*}$ of the partition. The most likely key is the one that gives rise to the highest variance ratio. Note that variance tests have been used in the context of timing attacks (*e.g.* in [9]). However, we could not find a reference using a similar test in the context of power analysis attacks. Any suggestion is welcome.

We finally mention that partition-based attacks generally require the partitions corresponding to different key candidates to be made of meaningful vectors $\mathbf{p}^i_{s^*}$. For example, an attack against the 8 key-bits corresponding to the first S-box of the AES using an 8-bit partition will give rise to vectors $\mathbf{p}^i_{s^*}$ containing only the leakages corresponding to one input x_i. Therefore, these partitions will not allow discriminating the key candidates. In other words, partition attacks cannot use bijective hypothetical leakage functions [6].

5 Evaluation Metrics

We propose to quantify the effectiveness of our distinguishers with two security metrics, namely the success rates of order o and guessing entropy. Let $\mathbf{g}_q$ be the vector containing the key candidates sorted according to the test result after a side-channel attack has been performed: $\mathbf{g}_q := [g_1, g_2, \ldots, g_{|\mathcal{S}|}]$. A success rate of order 1 (*resp.* 2, ...) relates to the probability that the correct key class is sorted first (*resp.* among the two first ones, ...) by the adversary. More formally, we define the success function of order o against a key class s as: $\mathsf{S}^o_s(\mathbf{g}_q)$=1 if $s \in [g_1, \ldots, g_o]$, else $\mathsf{S}^o_s(\mathbf{g}_q)$=0. It leads to the o^{th}-order success rate:

$$\mathbf{Succ}^o_{\mathcal{S}} = \underset{s}{\mathbf{E}} \; \underset{\mathbf{l}_q}{\mathbf{E}} \; \mathsf{S}^o_s(\mathbf{g}_q) \tag{6}$$

Similarly, the guessing entropy measures the average number of key candidates to test after a side-channel attack has been performed. Using the same notations as for the success rate, we define the index of a key class s in a side-channel

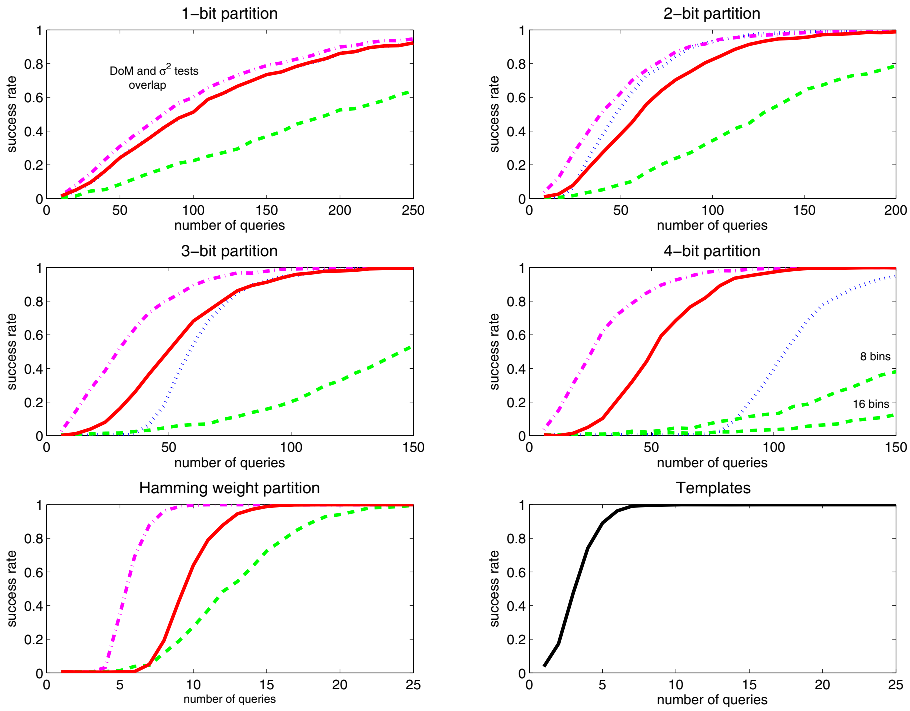


Fig. 3. PIC: 1^{st}-order SR for different statistical tests, partitions, models (dotted: DoM test, dash-dotted: correlation test, dashed: MIA, solid: variance test)

attack as: $\mathsf{l}_s(\mathbf{g}_q) = i$ such that $g_i = s$. It corresponds to the position of the correct key class s in the candidates vector $\mathbf{g}_q$. The guessing entropy is simply the average position of s in this vector:

$$\mathbf{GE}_S = \underset{s}{\mathbf{E}} \; \underset{\mathbf{l}_q}{\mathbf{E}} \; \mathsf{l}_s(\mathbf{g}_q) \tag{7}$$

Intuitively, a success rate measures an adversarial strategy with fixed computational cost after the physical leakages have been exploited. The guessing entropy measures the average computational cost after this exploitation. For a theoretical discussion of these metrics, we refer to [18].

6 Limitations of Our Classification and Methodology

Before moving to the description of our experimental results, let us emphasize again that the previous classification of attacks is informal. It is convenient to consider partition-based attacks since they all exploit a division of the leakages such as in Table 1. But as far as the adversarial capabilities are concerned, the most important classification relates to the need (or lack thereof) of a leakage model. For example, template attacks require the strongest assumptions, *i.e.* a

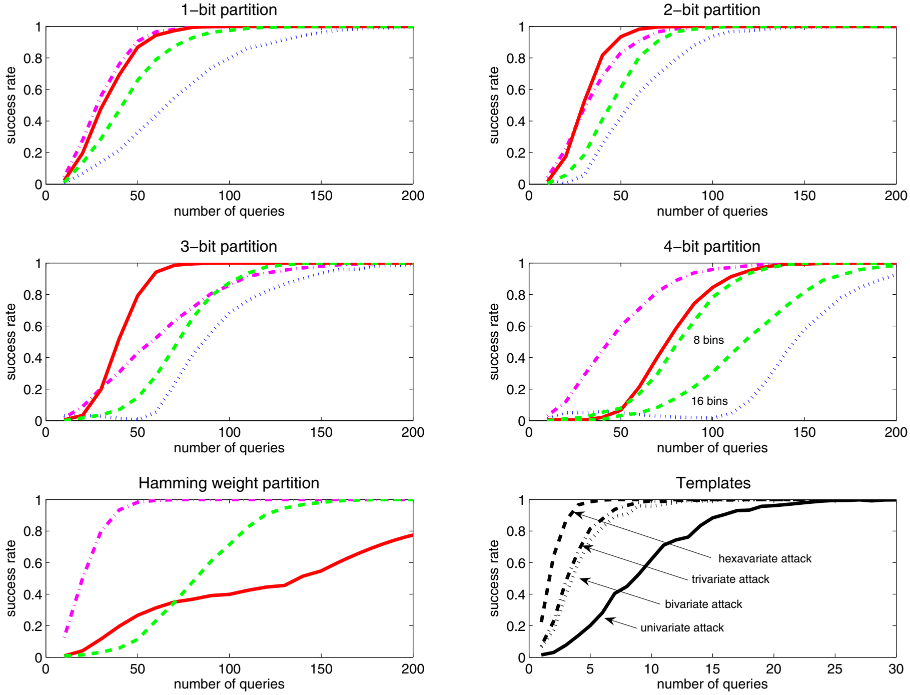


Fig. 4. Atmel: 1^{st}-order SR for different statistical tests, partitions, models (dotted: DoM test, dash-dotted: correlation test, dashed: MIA, solid: variance test)

precise characterization of the target device. Other attacks do not need but can be improved by such a characterization (*e.g.* correlation in [17]), with more or less resistance to a lack of knowledge on the target device. With this respect, the mutual information analysis is the most generic statistical test in the sense that it does not require any assumption on the leakage model.

Also, all our evaluations depend on the target implementations and attack scenarios, and hence are only valid within these fixed contexts. As will be shown in the next section, even two implementations of the same algorithm on similar platforms may yield contrasted results. Similarly, changing any part of the adversary in Section 3 (*e.g.* considering adaptively selected input plaintexts, another reduction mapping, . . .) or modifying the measurement setups could affect our conclusions. Importantly, these facts should not be seen as theoretical limitations of the proposed framework but as practical limitations in its application, related to the complex device-dependent mechanisms in side-channel attacks. Hence, it motivates the repetition of similar experiments in various other contexts.

7 Experimental Results

In this section, we present the different experiments that we carried out against our two target devices. We investigated partition distinguishers with DoM tests,

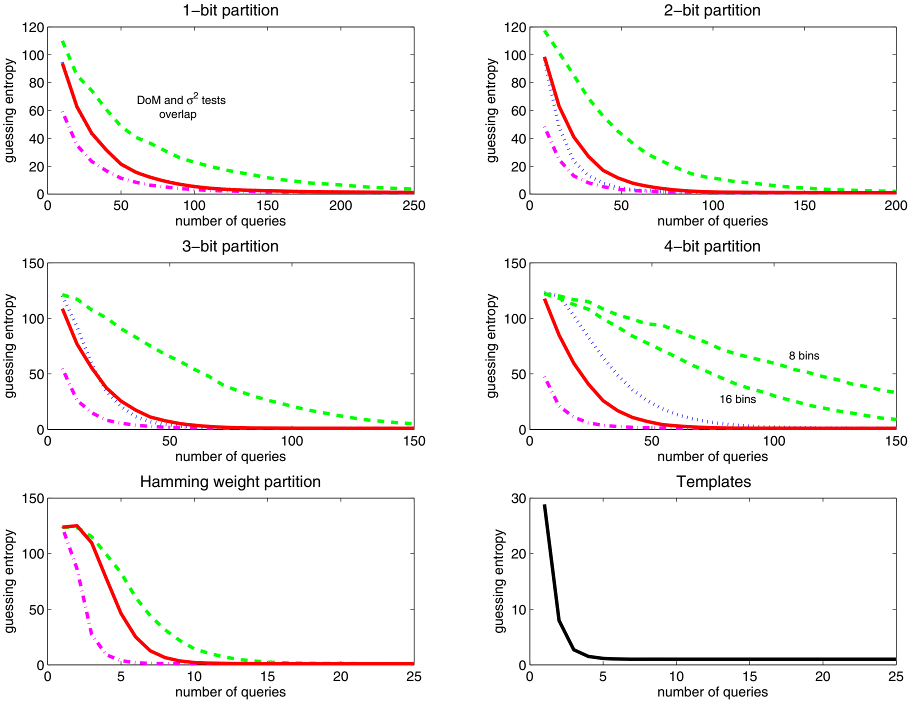


Fig. 5. PIC: Guessing entropy for different statistical tests, partitions, models (dotted: DoM test, dash-dotted: correlation test, dashed: MIA, solid: variance test)

MIA and variance tests. We also evaluated correlation attacks using Pearson's coefficient and template attacks. For this purpose and for each device, we generated 1000 leakage vectors, each of them corresponding to q=250 random input plaintexts. Then, for each of the previously described statistical tests, we computed different success rates and the guessing entropy for:

- various number of queries ($1 \leq q \leq 250$),
- various partitions and models (1-bit, 2-bit, . . . , Hamming weight).

For each value of q, our metrics were consequently evaluated from 1000 samples. The results are represented in Figures 3, 4, 5, 6, 7, (the latter ones in Appendix) and lead to a number of observations that we now detail.

1. The two devices have significantly different leakage behaviors. While the PIC leakages closely follow Hamming weight predictions (*e.g.* Figure 3, correlation test, lower left part), the Atmel leakages give rise to less efficient attacks in this context (*e.g.* Figure 4, correlation test, lower left part).
2. By contrast 1-bit and 2-bit partitions give rise to more efficient attacks against the Atmel device than against the PIC (*e.g.* Figures 3 and 4 again).

The assumed reason for these observations is that different bits in the Atmel implementation contribute differently to the overall leakage. In particular, we

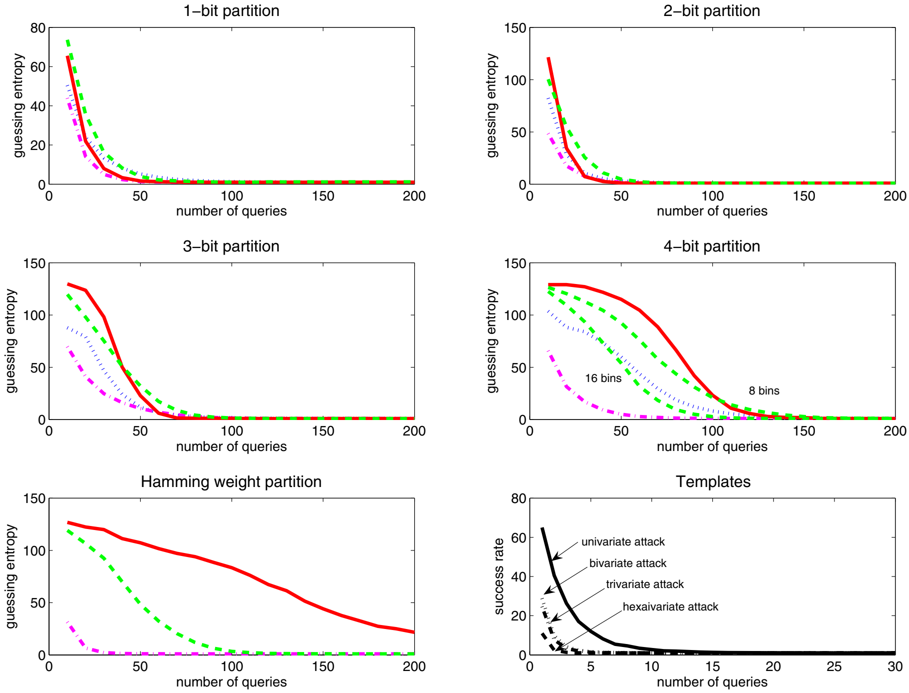


Fig. 6. Atmel: Guessing entropy for different statistical tests, partitions, models (dotted: DoM test, dash-dotted: correlation test, dashed: MIA, solid: variance test)

observed experimentally that 1-bit attacks were the most efficient when targeting the S-box output LSB against the Atmel (the success rate was significantly lower with other bits). This assumption also explains why multiple-bit attacks lead to relatively small improvement of the attacks, compared to the PIC.

3. As far as the comparison of distinguishers is concerned, the main observation is that template attacks are the most efficient ones against both devices, confirming the expectations of [3,18]. However, it is worth noting that while univariate templates directly lead to very powerful attacks against the PIC implementation, the exploitation of multiple samples significantly improves the success rate in the Atmel context (*e.g.* Figure 4, lower right part). Note again that the unbounded construction of our templates has a significant impact on this observation. The effect of a bounded construction of templates to the final attack effectiveness has been studied in [7].
4. By contrast, no general conclusions can be drawn for the non-profiled distinguishers (DoM and variance tests, correlation attack, MIA). This confirms that different adversarial contexts (*e.g.* types of leakages, distributions of the noise, selections of the meaningful samples, ...) can lead to very different results for these attacks. A distinguisher can also be fast to reach low success rates but slow to reach high success rates. Or different distinguishers could be more or less immune to noise addition or other countermeasures against

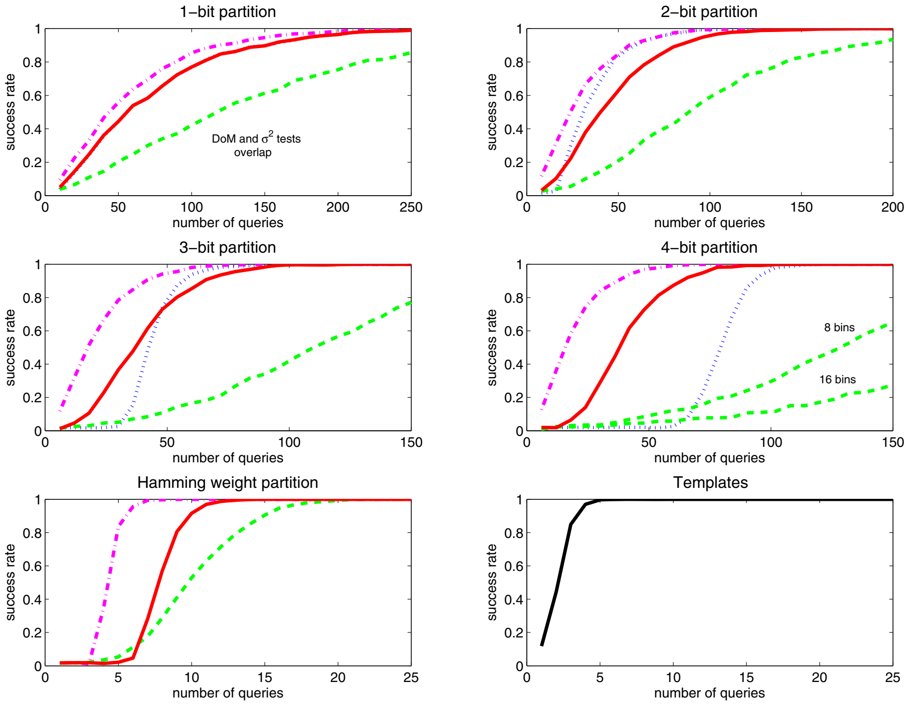


Fig. 7. PIC: 4^{th}-order SR for different statistical tests, partitions, models (dotted: DoM test, dash-dotted: correlation test, dashed: MIA, solid: variance test)

side-channel attacks. For example, in our experiments the DoM test shows an effectiveness similar the other distinguishers against the PIC with 1-bit partitions while it is the least efficient against the Atmel.

Next to these general observations, more specific comments can be made, *e.g.*:

- The results for the DoM test against the PIC (Figure 3) experimentally confirm the prediction of Messerges in [13]: in the context of "*all-or-nothing*" multiple-bit attacks using a DoM test, the best partition size is 3 bit out of 8 if the leakages have strong Hamming weight dependencies. It corresponds to the best tradeoff between the amplitude of the DoM peak (that increases with the size of the partition) and the number of traces that are not used by the test because not corresponding to an "*all zeroes/ones*" vector.
- The different metrics introduced, although correlated, bring different insights on the attacks efficiencies: Figures 5, 6 illustrate the guessing entropies of different attacks for our two devices; Figure 7 contains the 4^{th}-order success rates for the PIC. Interestingly, the variance test using 4-bit partitions against the Atmel allows a better 1^{st}-order success rate than guessing entropy, compared to other distinguishers (*e.g.* Figures 4, 6: middle right parts).
- The number of bins used to build the histograms in the MIA has a significant impact on the resulting attack efficiency. More bins generally allow a

better estimation of the mutual information $\hat{\mathrm{I}}(\mathsf{R}(\mathbf{L}_q);\mathbf{H}^q_{s^*})$ but can lead to less discriminant attacks if the number of leakage samples is bounded. In general, we use as many bins as the number of vectors in our partitions. For the 4-bit partitions, we additionally considered 8-bins-based attacks to illustrate the impact of a change of this parameter. The optimal selection of these bins and their number is an interesting scope for further research.

This list of comments is of course not exhaustive and only points out exemplary facts that can be extracted from our experiments. We finally emphasize the importance of a sufficient statistical sampling in the approximation of the success rates or guessing entropy in order to provide meaningful conclusions. While an actual adversary only cares about recovering keys (*i.e.* one experiment may be enough for this purpose) the evaluation and understanding of side-channel attacks requires confidence in the analysis of different statistical tests. In practice, such evaluations are obviously limited by the amount of traces that one can acquire, store and process. With this respect, we computed our success rates and guessing entropies from sets of 1000 samples (*i.e.* 1000 leakage vectors of 250 encrypted plaintexts each). Both the smoothness of the curves in our figures and the confidence intervals that can be straightforwardly extracted for the success rates confirm that this sampling was enough to obtain sound observations.

8 Conclusions

This paper describes a fair empirical comparison of different side-channel distinguishers against two exemplary devices. Our results essentially highlight the implementation-dependent nature of such comparisons. It shows that any conclusion about the efficiency of a side-channel attack is only valid within a specific context. Therefore it emphasizes the importance of performing similar evaluations against various other implementations. In particular, countermeasures against side-channel attacks (*e.g.* masked [8] or dual-rail circuits [19]) are an interesting evaluation target. Other scopes for further research include the integration of more complex side-channel attacks in the comparisons, *e.g.* based on collisions [16] or the investigation of advanced statistical tools for key extraction, *e.g.* [1]. The methodology described in this work is expected to prevent wrong general claims on side-channel attacks and to allow a better understanding of both the target devices and the attacks used to exploit physical leakages.

References

1. Batina, L., Gierlichs, B., Lemke-Rust, K.: Comparative Evaluation of Rank Correlation based DPA on an AES Prototype Chip. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 341–354. Springer, Heidelberg (2008)
2. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)

3. Chari, S., Rao, J., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
4. Coron, J.S., Naccache, D., Kocher, P.: Statistics and Secret Leakage. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 157–173. Springer, Heidelberg (2001)
5. FIPS 197, Advanced Encryption Standard, Federal Information Processing Standard, NIST, U.S. Dept. of Commerce, November 26 (2001)
6. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis - A Generic Side-Channel Distinguisher. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
7. Gierlichs, B., Lemke, K., Paar, C.: Templates vs. Stochastic Methods. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 15–29. Springer, Heidelberg (2006)
8. Goubin, L., Patarin, J.: DES and Differential Power Analysis. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
9. Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
10. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
11. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks. Springer, Heidelberg (2007)
12. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Examining Smart-Card Security under the Threat of Power Analysis Attacks. IEEE Transactions on Computers 51(5), 541–552 (2002)
13. Messerges, T.S.: Power Analysis Attacks and Countermeasures for Cryptographic Algorithms, PhD Thesis, University of Illinois at Urbana Champaign (2000)
14. Prouff, E.: DPA Attacks and S-Boxes. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 424–441. Springer, Heidelberg (2005)
15. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side-Channel Cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
16. Schramm, K., Leander, G., Felke, P., Paar, C.: A Collision-Attack on AES: Combining Side Channel and Differential Attack. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 163–175. Springer, Heidelberg (2004)
17. Standaert, F.-X., Peeters, E., Macé, F., Quisquater, J.-J.: Updates on the Security of FPGAs Against Power Analysis Attacks. In: Bertels, K., Cardoso, J.M.P., Vassiliadis, S. (eds.) ARC 2006. LNCS, vol. 3985, pp. 335–346. Springer, Heidelberg (2006)
18. Standaert, F.-X., Malkin, T.G., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks, Cryptology ePrint Archive, Report 2006/139
19. Tiri, K., Akmal, M., Verbauwhede, I.: A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand DPA on Smart Cards. In: The proceedings of ESSCIRC 2003, Estoril, Portugal (September 2003)

A Single-Key Domain Extender for Privacy-Preserving MACs and PRFs

Kan Yasuda

NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midoricho Musashino-shi, Tokyo 180-8585 Japan
yasuda.kan@lab.ntt.co.jp

Abstract. We present a CBC (cipher block chaining)-like mode of operation for MACs (message authentication codes) using a hash function. The new construction iCBC (imbalanced CBC) does not follow the Merkle-Damgård design but rather iterates the underlying compression function directly in a CBC-like manner. Many of the prior MAC constructions, including HMAC, assume PRF (pseudo-random function) properties of the underlying primitive. In contrast, our iCBC-MAC makes only a PP-MAC (privacy-preserving MAC) assumption about the compression function. Despite the fact that PP-MAC is a strictly weaker requirement than PRF, iCBC-MAC works with a single key like HMAC and runs as efficiently as HMAC. Moreover, iCBC-MAC becomes even faster than HMAC, depending on the choice of security parameters. Additionally, iCBC-MAC is multi-property-preserving in the sense that it operates as a domain extender for both PP-MACs and PRFs.

Keywords: imbalanced cipher block chaining, iCBC, message authentication code, MAC, privacy-preserving, domain extension.

1 Introduction

HMAC [1] is a popular, widely-used mode of operation for message authentication codes (MACs). It is based on the Merkle-Damgård structure which iterates a hash compression function. HMAC is single-keyed and provably secure [2] in the sense that it is a pseudo-random function (PRF) on the assumption that the underlying compression function is a PRF.

Being a PRF is a strictly stronger condition than being a secure MAC. For the purpose of message authentication, PRF properties are not necessarily required. Moreover, the strong PRF assumptions might not hold true for some of the actual compression functions in use.

In practice, several dedicated compression functions, including md4, md5 and sha-0, turn out to be *not* PRFs. These facts are demonstrated by the successive attacks [3,4,5,6] on the instantiations of NMAC/HMAC with those "weak" compression functions. The attacks do not violate the security proof of HMAC but merely show that HMAC is indeed vulnerable when the underlying assumption becomes void.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 268–285, 2009.

The case of HMAC reminds us that it is desirable to base the security of constructions on weaker assumptions about the building block. In particular, PRF assumptions about the primitive seem to be inessential to constructing secure MACs, especially as there already exist a few secure constructions of MACs that require only a weaker-than-PRF property of the underlying compression function.

Weakening PRF assumption. The NI construction [7] provides a secure MAC on the sole assumption that the underlying compression function is a secure MAC. NI requires two independent keys, and this problem is resolved by the newer CS construction [8].

The major drawback of the NI and CS constructions is inefficiency due to their dependence on the *keyed* Merkle-Damgård structure. Consider for example md5 : $\{0,1\}^{128+512} \rightarrow \{0,1\}^{128}$. The usual, unkeyed Merkle-Damgård iteration can process 512 bits of a message per invocation to the md5 compression function. In contrast, the keyed Merkle-Damgård iteration consumes (say) 128 bits for a key at each invocation and hence processes only 384 bits of a message per block. Thus, the NI and CS constructions perform at the rate of about $384/512 = 75\%$ to HMAC in this particular case. We are interested in resolving this dilemma between weakening the assumption and achieving efficiency:

> ***Q. Can we construct a single-key mode of operation satisfying the following requirements?***
>
> - ***The mode provides a secure MAC based on a weaker-than-PRF assumption about the underlying compression function, and***
> - ***The mode runs as fast as or faster than HMAC.***

Our Results. We aim at the privacy-preserving MAC (PP-MAC) property, a notion that lies between that of a PRF and that of a secure MAC.[1] By doing so, we successfully weaken the PRF assumption about the compression function without losing overall performance. The key idea is that owing to the privacy-preserving property we become able to employ a new iterative structure that is more efficient than the keyed Merkle-Damgård construction.

We call the new configuration "imbalanced" cipher block chaining (iCBC). It is similar to the usual CBC that iterates a block cipher. The obvious difference from the normal CBC lies in the fact that the hash size n is always smaller than the block size b, as we are dealing with a *compression* function. We can easily resolve the mismatch by padding each hash value with 0's to b bits before xor-ing it into the next message block.

Adopting the iCBC method rather than the Merkle-Damgård iteration, our mode of operation, iCBC-MAC, accomplishes the following features:

[1] It turns out that the "non-PRF" compression functions md4, md5 and sha-0 are not even PP-MAC, but we shall show that in theory there is a significant difference between the two notions of PRF and PP-MAC. See Sect. 4.

1. **Extending PP-MAC Property.** We show that iCBC-MAC is a PP-MAC based on the sole assumption that the underlying compression function is a PP-MAC. This means that for the purpose of message authentication, iCBC-MAC does not require its compression function to be a PRF.
2. **Extending PRF Property.** We also show that iCBC-MAC is a PRF under the condition that the compression function is a PRF. Hence, one can replace HMAC with iCBC-MAC without losing the functionality as a PRF.
3. **Keeping High Performance.** The iCBC configuration is as efficient as the *unkeyed* Merkle-Damgård construction. Depending on the key size κ and the hash size n, iCBC becomes even faster than the unkeyed Merkle-Damgård iteration (*i.e.*, when $\kappa < n$).
4. **Being "Truly" Single-Keyed.** The iCBC-MAC scheme provides a single-key solution. Every invocation to the compression function takes the same secret key.[2]

The disadvantage of iCBC-MAC may be the fact that it requires direct access to the compression function. It means that iCBC-MAC cannot make good use of off-the-shelf hash functions that are already implemented in the Merkle-Damgård style. However, several studies [9,10,11,12] reveal various structural defects in the Merkle-Damgård construction as a mode of operation for hash functions, which leads us to question whether the Merkle-Damgård construction will maintain its dominant position.

Organization. In Sect. 2 we review related work in the field. Section 3 provides notation and terminology used in the paper. Section 4 defines security notions, including PRFs and PP-MACs. Section 5 is devoted to proving the main lemma concerning the security of iCBC configuration. In Sect. 6 we define the iCBC-MAC construction and prove its security, using the result of the main lemma. In Sect. 7 we mention techniques for further optimization of the iCBC-MAC construction. We discuss the efficiency of iCBC-MAC in Sect. 8, making a performance comparison with HMAC.

2 Prior Work

In this section we go over previous modes of operation for MACs and PRFs. Some of the prominent ones are listed in Table 1 with their security results.

There are a few domain extensions of MACs, such as NI, CS, ESh [13], MDP [16] and enciphered CBC [17]. All of them are subject to performance loss as compared to PRF-based constructions. NI, CS, ESh and MDP (in its MAC mode) iterate a compression function in the keyed Merkle-Damgård style. The enciphered CBC iterates a length-preserving primitive (*e.g.*, a block cipher), and it is twice as slow as an ordinary, PRF-based CBC-MAC such as OMAC [18].

[2] In the single-key scenario a folklore technique is to stretch a key to its double length using a PRF (as in the derivation of HMAC from NMAC [2]). This technique is inapplicable to our case of PP-MAC compression functions.

Table 1. Comparison among HMAC, iCBC-MAC and other MACs. PRF(two) stands for PRF against two queries. Note that HMAC and BNMAC are in the keyless setting, while CS and iCBC-MAC are in the dedicated-key setting [13].

	Performance	Goal	Assumption	Ref.
CS	<HMAC	MAC	MAC	[8]
HMAC	=HMAC	MAC	MAC+CR+PRF(two)	[1]
		MAC	NM+PRF(two)	[14]
		PP-MAC	PP-MAC+PRF(two)	[2]
		PRF	PRF	[2]
iCBC-MAC	**≥HMAC**	**PP-MAC**	**PP-MAC**	—
		PRF	**PRF**	—
BNMAC	>HMAC	PRF	PRF(related key)	[15]

There are several proofs known for the security of HMAC. All of them more or less rely on a PRF property of the compression function. The original proof [1] demands that the compression function be MAC-secure and collision-resistant (CR), along with a PRF assumption for key derivation. The recent work [14] provides a security proof of NMAC based on non-malleability (NM), which is a weaker condition than PRF. Still, the proof requires a PRF assumption for key derivation in the case of HMAC. The refined proof [2] by one of the designers shows that NMAC is a PP-MAC under the condition that the compression function is a PP-MAC and its Merkle-Damgård iteration is computationally almost universal (cAU). The cAU property is derived from a PRF property of the compression function. To prove that HMAC is a PP-MAC, the proof requires again a PRF assumption for key derivation. The work [2] also proves that HMAC is a PRF based on PRF assumptions about the compression function.

ENMAC [19], MDP and BNMAC [15] are modes of operation with improved performance over HMAC. ENMAC essentially inherits security results of HMAC. MDP (in its PRF mode) and BNMAC require even stronger requirements of PRF under related-key attacks.

3 Preliminaries

General Notation. Given two strings $x, y \in \{0,1\}^*$, we write $x \| y$ for the concatenation of x and y. We write $|x| = n$ when $x \in \{0,1\}^n$. The xor $x \oplus y$ is defined for two strings x and y of equal length, *i.e.*, $|x| = |y|$. Given a string $x \in \{0,1\}^*$ with $|x| \geq n$, we define $[x]^n$ as the string consisting of the leftmost n bits of x, so that $[x]^n \in \{0,1\}^n$. We write 0^n for the string $00 \cdots 0 \in \{0,1\}^n$ and likewise 1^m. The symbol $\|$ is often omitted, *e.g.*, $0^n 1^m = 0^n \| 1^m$.

The notation $x \leftarrow y$ means the operation of assigning the value y to variable x. Given a set X, we write $x \xleftarrow{\$} X$ for selecting an element from X uniformly at random and assigning its value to variable x. We write $x, y \xleftarrow{\$} X$ to mean $x \xleftarrow{\$} X$, $y \xleftarrow{\$} X$. Given two positive integers α and β with $\alpha \leq \beta$, $[\alpha, \beta]$ denotes the set of integers i such that $\alpha \leq i \leq \beta$. The notation $\langle i \rangle_n$ represents some canonical n-bit encoding of a positive integer i.

Compression Functions and iCBC. We pick a compression function $g_k : \{0,1\}^b \to \{0,1\}^n$ with keys $k \in \{0,1\}^\kappa$. For function g_k to be *compressing*, we require $b > n$. For the moment we assume no other relations between b, n, and κ. Throughout the paper we fix g and hence b, n and κ.

Given a message $M \in \{0,1\}^*$, we first need to pad M so that its length becomes a multiple of b bits. Put $\lambda = \lceil (|M| + 1)/b \rceil$. λ is the length of M *in blocks*. Divide M into b-bit blocks as $M = m_1 \| m_2 \| \cdots \| m_{\lambda-1} \| \tilde{m}_\lambda$, so that $|m_1| = |m_2| = \cdots = |m_{\lambda-1}| = b$ and $0 \le |\tilde{m}_\lambda| \le b - 1$. Then the message M is padded as $M \leftarrow M \| 10^{b-|\tilde{m}_\lambda|-1}$. We write $M \leftarrow M \| 10^*$ as shorthand for this padding process. Given a string M whose length is equal to $b\lambda$ bits, the notation $m_1 \| m_2 \| \cdots \| m_\lambda \leftarrow M$ means dividing the string M into b-bit blocks and assigning the value of each block to variables $m_1, m_2, \ldots, m_\lambda$, so that $m_1 \| m_2 \| \cdots \| m_\lambda = M$.

Given g_k we construct its iCBC iteration $H_k : \{0,1\}^* \to \{0,1\}^n$ as follows:[3]

```
Algorithm $H_k(M)$
$M \leftarrow M \| 10^*$; $m_1 \| m_2 \| \cdots \| m_\lambda \leftarrow M$; $y_1 \leftarrow 0^n$
For $i = 1, 2, \ldots, \lambda$ do $\bar{y}_i \leftarrow y_i \| 0^{b-n}$, $x_i \leftarrow \bar{y}_i \oplus m_i$, $y_{i+1} \leftarrow g_k(x_i)$ endFor
Output $y_{\lambda+1}$.
```

Adversaries, Games and Resources. An adversary A is a probabilistic algorithm with access to zero or more oracles. We write $A^{\mathcal{O}}$ to indicate the fact that adversary A interacts with oracle $\mathcal{O}$. We let $A^{\mathcal{O}}$ also denote the value that adversary A outputs after its interaction with oracle $\mathcal{O}$. The notation $x \leftarrow A^{\mathcal{O}}$ means assigning variable x the value output by A. We write $A^{\mathcal{O}} = \sigma$ to indicate the event that the value output by A happens to be equal to σ.

We write $\mathcal{G}(A)$ for running adversary A as described in game $\mathcal{G}$. We let $\mathcal{G}(A)$ also denote the value returned by game $\mathcal{G}$. We write $x \leftarrow \mathcal{G}(A)$ and $\mathcal{G}(A) = \sigma$ likewise. Games often involve flags. All flags are assumed to be initially set to 0 prior to execution of the games. For example, for a flag **Flag** we omit the initialization step **Flag** $\leftarrow 0$ in the description of the game. **Flag** gets set with the statement **Flag** $\leftarrow 1$. With abuse of notation we let **Flag** represent also the event "$\mathcal{G}(A)$ sets **Flag**" when game $\mathcal{G}$ and adversary A are clear from the context.

A security property defines its associated advantage function $\mathbf{Adv}_f^{***}(A)$ (see Sect. 4 for specific definitions), measuring the advantage of adversary A attacking the $***$ property of scheme f. We define

$$\mathbf{Adv}_f^{***}(t, q, \ell) \overset{\text{def}}{=} \max_A \mathbf{Adv}_f^{***}(A),$$

where max runs over all adversaries A having running time at most t, making at most q queries to its oracles, each query being at most ℓ blocks, outputting a set of values, each value being at most ℓ blocks. To measure time complexity we

[3] Note that this definition of H gives just a "raw" iCBC. In particular, H does not provide a secure MAC. In order to make it secure, we shall "envelope" H with a specialized invocation to g for our iCBC-MAC in Sect. 6.

fix a model of computation. The running time of adversary A includes its code size. The symbol $T_f(\ell)$ represents the time complexity needed to perform one computation of f on an ℓ-block input. One or more of the resource parameters t, q, ℓ may be irrelevant in some cases, in which the unnecessary parameters are omitted from the notation.

4 Security Definitions: PRF vs. PP-MAC

In this section we give definitions of PRF and PP-MAC notions. Intuitively, PRF requires that the output distribution should "look" uniformly random regardless of inputs, whereas PP-MAC requires that, in addition to being secure as a MAC, the output distribution should look the same (but not necessarily random) for all inputs. The practical compression functions md4, md5 and sha-0, which are not PRFs, also fail to be PP-MAC (The attacks [3,4,5,6] give this fact). Still, we believe that the gap between the two notions is of theoretical significance as explained below.

PRF. A prf-adversary A attacking a family of functions $f_k : X \to Y$ tries to distinguish between the Real oracle and the Random oracle. The Real oracle picks a key $k \overset{\$}{\leftarrow} K$ at the beginning of each experiment and returns the value $f_k(x)$ to A upon query $x \in X$. The Random oracle picks a function $\varphi : X \to Y$ uniformly at random (from the space of all functions $X \to Y$) at the beginning of each experiment and returns the value $\varphi(x)$ to A upon query $x \in X$. Define $\mathbf{Adv}_f^{\mathrm{prf}}(A) \overset{\mathrm{def}}{=} \Pr\big[A^{\mathrm{Real}} = 1\big] - \Pr\big[A^{\mathrm{Random}} = 1\big]$, where the probabilities are over all coins of the oracle and the adversary (This principle applies to all probabilities below).

PP-MAC. The notion of PP-MAC [2] is a combination of unforgeability and left-or-right indistinguishability. We start with defining the latter notion of privacy preservation. Let $f_k : X \to Y$ be a function family. The Left oracle picks a key $k \overset{\$}{\leftarrow} K$ at the beginning of each experiment and replies $y \leftarrow f_k(x)$ upon query (x, x'), whereas the Right oracle replies $f_k(x')$ upon query (x, x'). Define $\mathbf{Adv}_f^{\mathrm{ind}}(A) \overset{\mathrm{def}}{=} \Pr\big[A^{\mathrm{Left}} = 1\big] - \Pr\big[A^{\mathrm{Right}} = 1\big]$, where we demand that ind-adversary A's queries be legitimate. Let $(x_1, x'_1), (x_2, x'_2), \ldots$ represent the queries made by adversary A. We say that the queries made by A are *legitimate* if "$x_i = x_j$ implies $x'_i = x'_j$" and "$x'_i = x'_j$ implies $x_i = x_j$." This means that the values $x_1, x_2, \ldots$ are all distinct, and so are $x'_1, x'_2, \ldots$, except when A repeats its queries.

The notion of privacy preservation itself is not a demanding property, as already pointed out by [2]. For example, a function that always outputs $f_k(x) = 00 \cdots 0$ is privacy-preserving.

For the notion of unforgeability, we adopt the standard one-time-verification model.[4] A mac-adversary A is given access to the oracle $f_k(\cdot)$ and outputs a pair of values (x^*, y^*). The adversary A wins if $f_k(x^*) = y^*$ and x^* is new. We say

[4] It suffices to consider only the one-time verification model in the current work, because we are dealing with deterministic MACs [20].

that $x^* \in X$ is *new* if it has not been queried by A to its oracle $f_k(\cdot)$. Succinctly, define $\mathbf{Adv}_f^{\mathrm{mac}}(A) \overset{\mathrm{def}}{=} \Pr\bigl[x^* \text{ is new and } f_k(x^*) = y^* \bigm| (x^*, y^*) \leftarrow A^{f_k(\cdot)}\bigr]$.

PRF $\Rightarrow$ PP-MAC but PRF $\not\Leftarrow$ PP-MAC. Let $f_k : X \to \{0,1\}^n$ be a function family with keys $k \in K$. If f is a PRF, then it is a PP-MAC. Indeed, we have $\mathbf{Adv}_f^{\mathrm{ind}}(t, q, \ell) \le 2 \cdot \mathbf{Adv}_f^{\mathrm{prf}}(t, q, \ell)$ [2] and

$$\mathbf{Adv}_f^{\mathrm{mac}}(t, q, \ell) \le \mathbf{Adv}_f^{\mathrm{prf}}(t', q+1, \ell) + \tfrac{1}{2^n}, \tag{1}$$

where $t' = t + T_f(\ell)$ [20].

On the other hand, being a PP-MAC does not necessarily imply being a PRF. For example, define $f'_k : X \to \{0,1\}^{n+1}$ as $f'_k(x) \overset{\mathrm{def}}{=} f_k(x)\|0$ for $k \in K$ and $x \in X$. If f is a PP-MAC, then so is f'. However, f' is clearly not a PRF.

5 Main iCBC Lemma

In this section we analyze the security of the “raw” iCBC construction H and obtain a main lemma that is used to prove security results for iCBC-MAC.

Higher-Order AU. We introduce the notion of higher-order almost universal (HOAU) functions. A hoau-adversary A, given access to its oracle $g_k(\cdot)$, outputs a pair of messages $M, M' \in \{0,1\}^*$. A’s goal is to come up with such a pair as $H_k(M) = H_k(M')$, where H is the iCBC that is constructed of g. In order for A to succeed, we demand (i) $M \neq M'$, and (ii) M and M' be fresh. We say that a message M is *fresh* if in its computation of $H_k(M)$ all the input values to g are new, *i.e.*, none of the input values to g has been queried by A to its oracle $g_k(\cdot)$ (The input values correspond to the values $x_1, x_2, \ldots, x_\lambda$ in the definition of the iCBC H). Succinctly, define

$$\mathbf{Adv}_g^{\mathrm{hoau}}(A) \overset{\mathrm{def}}{=} \Pr\bigl[M, M' \text{ are fresh, } M \neq M' \text{ and } H_k(M) = H_k(M') \\ \bigm| (M, M') \leftarrow A^{g_k(\cdot)}\bigr].$$

Note that when the maximum number q of queries to the oracle is equal to 0, the notion of HOAU corresponds to that of cAU [2] for H.

Lemma 1 (Main iCBC Lemma). *Let $g_k : \{0,1\}^b \to \{0,1\}^n$ be a compression function with keys $k \in \{0,1\}^\kappa$. If g is a secure MAC, then g is HOAU. Specifically, we have*

$$\mathbf{Adv}_g^{\mathrm{hoau}}(t, q, \ell) \le \tfrac{7\ell^2}{2} \cdot \mathbf{Adv}_g^{\mathrm{mac}}(t', q + 2\ell - 1),$$

where $t' = t + (2\ell - 1) \cdot T_g$.[5]

Proof. Let A be an adversary, attacking the HOAU property of g, having running time at most t, making at most q queries to the $g_k(\cdot)$ oracle, outputting a pair of messages, each being at most ℓ blocks. Let M, M' denote the two messages

```
Adversary $S$
Sub $Q$; Choose $\alpha \xleftarrow{\$} [1, \lambda']$
Sub $R'(\alpha)$
Output $(x'_\alpha, 0^n)$ as a forgery
```

```
Adversary $O_1$
Sub $Q$; Choose $\alpha \xleftarrow{\$} [1, \lambda]$ and $\beta \xleftarrow{\$} [1, \lambda']$
Sub $R(\alpha + 1)$; Sub $R'(\beta)$
Output $(x'_\beta, y_{\alpha+1})$ as a forgery
```

```
Adversary $O_2$
Sub $Q$
Choose $\alpha^*, \beta^* \xleftarrow{\$} [1, \lambda]$ with $\alpha^* < \beta^*$
Sub $R(\beta^*)$
Output $(x_{\beta^*}, y_{\alpha^*+1})$ as a forgery
```

```
Adversary $I_1$
Sub $Q$
Choose $\alpha \xleftarrow{\$} [1, \lambda]$ and $\beta \xleftarrow{\$} [1, \lambda']$
Sub $R(\alpha)$; Sub $R'(\beta)$; $\tau \leftarrow [x_\alpha \oplus m'_{\alpha+\lambda'-\lambda}]^n$
Output $(x'_\beta, \tau)$ as a forgery
```

```
Adversary $I_2$
Sub $Q$
Choose $\alpha, \beta \xleftarrow{\$} [1, \lambda]$ with $\beta < \alpha - 1$
Sub $R(\alpha - 1)$
$\tau \leftarrow [\bar{y}_{\beta+1} \oplus m'_{\alpha+\lambda'-\lambda} \oplus m_\alpha]^n$
Output $(x_{\alpha-1}, \tau)$ as a forgery
```

```
Adversary $I_3$
Sub $Q$
Choose $\alpha^*, \beta^* \xleftarrow{\$} [1, \lambda]$ with $\alpha^* < \beta^*$
Sub $R(\beta^* - 1)$
$\tau \leftarrow [x_{\alpha^*} \oplus m_{\beta^*}]^n$
Output $(x_{\beta^*-1}, \tau)$ as a forgery
```

```
Subroutine $Q$
$(M, M') \leftarrow A^{g_k(\cdot)}$; $M \leftarrow M \| 10^*$; $M' \leftarrow M' \| 10^*$
$m_1 \| m_2 \| \cdots \| m_\lambda \leftarrow M$; $m'_1 \| m'_2 \| \cdots \| m'_{\lambda'} \leftarrow M'$
```

```
Subroutine $R(\alpha)$
$y_1 \leftarrow 0^n$; $\bar{y}_1 \leftarrow 0^b$; $x_1 \leftarrow m_1$
For $i = 2, 3, \ldots, \alpha$ do
   Make a query $y_i \leftarrow g_k(x_{i-1})$
   $\bar{y}_i \leftarrow y_i \| 0^{b-n}$; $x_i \leftarrow \bar{y}_i \oplus m_i$ endFor
```

```
Subroutine $R'(\alpha)$
$y'_1 \leftarrow 0^n$; $\bar{y}'_1 \leftarrow 0^b$; $x'_1 \leftarrow m'_1$
For $i = 2, 3, \ldots, \alpha$ do
   Make a query $y'_i \leftarrow g_k(x'_{i-1})$
   $\bar{y}'_i \leftarrow y'_i \| 0^{b-n}$; $x'_i \leftarrow \bar{y}'_i \oplus m'_i$ endFor
```

Fig. 1. Six adversaries used in main iCBC lemma

that A outputs. Without loss of generality we assume that $|M| \leq |M'|$, so that we have $\lambda \leq \lambda'$ in the definition of subroutine Q in Fig. 1.

Using the adversary A, we construct adversaries that break the MAC security of g. Specifically, we come up with the six adversaries defined in Fig. 1. These adversaries simply simulate the $g_k(\cdot)$ oracle for A using their $g_k(\cdot)$ oracle as in the definition of subroutine Q. Note that these queries do not affect the forgery probabilities of the six adversaries, owing to the freshness requirement of HOAU. The six adversaries correspond to certain specific events that occur when adversary A succeeds in outputting a fresh, colliding pair (M, M'). In the following we describe these events and relate them to the six adversaries.

Let variables m_i, m'_i, x_i, x'_i, y_i, y'_i be as in the definition of subroutines Q, R and R'. We observe that whenever adversary A succeeds, at least one of the following events occurs:

Clear-Cut Suffix: This is the case when $m_i = m'_{i+\lambda'-\lambda}$ for all $i \in [1, \lambda]$ (*i.e.*, M is a suffix of M') and $y'_{1+\lambda'-\lambda} = 0^n$. It also implies that $x_i = x'_{i+\lambda'-\lambda}$ and $y_i = y'_{i+\lambda'-\lambda}$ for all $i \in [1, \lambda]$.

[5] This lemma only claims that g is HOAU and particularly its iCBC H cAU. Note that H is not a secure MAC, as already illustrated by [7] for CBC-MACs.

Output Collision: In this case there exists an index $\alpha \in [1, \lambda]$ such that $x_\alpha \neq x'_{\alpha+\lambda'-\lambda}$ and $y_{\alpha+1} = y'_{\alpha+1+\lambda'-\lambda}$.

Input Collision: This last case is when there exists an index $\alpha \in [1, \lambda]$ such that $x_\alpha = x'_{\alpha+\lambda'-\lambda}$ and $y_\alpha \neq y'_{\alpha+\lambda'-\lambda}$. Note that $y_1 = y'_1 = 0^n$ by definition. In this case we also have $m_\alpha \neq m'_{\alpha+\lambda'-\lambda}$.

If M is not a clear-cut suffix of M', then it means that either (i) there exists an index $j \in [1, \lambda]$ such that $m_j \neq m'_{j+\lambda'-\lambda}$, or (ii) $y'_{1+\lambda'-\lambda} \neq 0^n$. Together with the fact that $y_{\lambda+1} = y'_{\lambda'+1}$, (i) or (ii) guarantees that either an output collision or an input collision occurs.

We start with the case **Clear-Cut Suffix**. We let **Suff** denote the event that this case occurs, over random choice of keys $k \xleftarrow{\$} \{0,1\}^\kappa$ and internal coins of A. The event **Suff** implies $y'_{1+\lambda'-\lambda} = 0^n$ and hence $g_k(x'_{\lambda'-\lambda}) = 0^n$. So let $\alpha \in [1, \lambda' - \lambda]$ be the smallest index such that $x'_\alpha = x'_{\lambda'-\lambda}$. Then we see that adversary S succeeds in producing a forgery at least when S guesses the correct value of α. The choice of value α in the description of adversary S is independent of event **Suff**, so we obtain $\mathbf{Adv}_g^{\mathrm{mac}}(S) \geq \frac{1}{\ell} \cdot \Pr\big[\mathbf{Suff}\big]$.

We next treat the case **Output Collision**, in which we have $x_\alpha \neq x'_{\alpha+\lambda'-\lambda}$ and $y_{\alpha+1} = y'_{\alpha+1+\lambda'-\lambda}$ for some $\alpha \in [1, \lambda]$. We divide the case into two smaller cases:

Case $\boldsymbol{x_i \neq x'_{\alpha+\lambda'-\lambda}}$ for all $\boldsymbol{i \in [1, \alpha]}$. We set **OutColl1** to be the event that this case occurs. Let $\beta \in [1, \alpha + \lambda' - \lambda]$ be the smallest index such that $x'_\beta = x'_{\alpha+\lambda'-\lambda}$. Then observe that upon event **OutColl1** adversary O_1 succeeds in producing a forgery at least when value (α, β) is guessed correctly. We therefore obtain $\mathbf{Adv}_g^{\mathrm{mac}}(O_1) \geq \frac{1}{\ell^2} \cdot \Pr\big[\mathbf{OutColl1}\big]$.

Case $\boldsymbol{x_\beta = x'_{\alpha+\lambda'-\lambda}}$ for some $\boldsymbol{\beta \in [1, \alpha - 1]}$. We write **OutColl2** for the event that this case occurs. Let $i \in [1, \alpha]$ be the smallest index such that $x_i = x_\alpha$ and $j \in [1, \beta]$ the smallest index such that $x_j = x_\beta$. The condition $x_\alpha \neq x_\beta$ assures that $i \neq j$. Put $\alpha^* \leftarrow \min\{i, j\}$ and $\beta^* \leftarrow \max\{i, j\}$. The adversary O_2 succeeds in breaking the MAC security of g at least when adversary O_2 guesses values (α^*, β^*) correctly upon event **OutColl2**. Thus we get $\mathbf{Adv}_g^{\mathrm{mac}}(O_2) \geq \frac{1}{\binom{\ell}{2}} \cdot \Pr\big[\mathbf{OutColl2}\big]$.

We finally proceed to the case **Input Collision**. Recall that in this case we have $x_\alpha = x'_{\alpha+\lambda'-\lambda}$ and $y_\alpha \neq y'_{\alpha+\lambda'-\lambda}$ for some $\alpha \in [1, \lambda]$. Note that this case also implies that $m_\alpha \neq m'_{\alpha+\lambda'-\lambda}$. We divide this case into the following two cases:

Case $\boldsymbol{x_i \neq x'_{\alpha-1+\lambda'-\lambda}}$ for all $\boldsymbol{i \in [1, \alpha - 1]}$. Let **InColl1** represent the event that this case occurs. We set $\beta \in [1, \alpha-1+\lambda'-\lambda]$ to be the smallest index such that $x'_\beta = x'_{\alpha-1+\lambda'-\lambda}$. We then see that adversary I_1 succeeds in producing a forgery at least when event **InColl1** occurs and I_1 guesses values (α, β) correctly. So we obtain $\mathbf{Adv}_g^{\mathrm{mac}}(I_1) \geq \frac{1}{\ell^2} \cdot \Pr\big[\mathbf{InColl1}\big]$.

Case $\boldsymbol{x_\beta = x'_{\alpha-1+\lambda'-\lambda}}$ for some $\boldsymbol{\beta \in [1, \alpha - 2]}$. We further divide this case into the following two situations:

Case $\boldsymbol{x_j \neq x_{\alpha-1}}$ for all $\boldsymbol{j \in [1, \alpha-2]}$. Write **InColl2** for the event of this case. This case guarantees that adversary I_2 never makes the query $x_{\alpha-1}$ to its oracle. Thus, adversary I_2 successfully forges upon event **InColl2**, when values (α, β) are guessed correctly. Therefore we get $\mathbf{Adv}_g^{\mathrm{mac}}(I_2) \geq \frac{1}{\binom{\ell}{2}} \cdot \Pr\big[\mathbf{InColl2}\big]$.

Case $\boldsymbol{x_\gamma = x_{\alpha-1}}$ for some $\boldsymbol{\gamma \in [1, \alpha-2]}$. We let **InColl3** denote the event that this case occurs. Define a set $U \overset{\text{def}}{=} \big\{(i,j) \bigm| 1 \leq i < j \leq \lambda,\ x_i = x_j\big\}$. The fact that we are in case **OutColl3** guarantees that U is non-empty, because $(\gamma, \alpha-1)$ is indeed in the set U. We introduce a linear order $\preceq$ to the set U: Given two elements $(i,j), (i',j') \in U$, define $(i,j) \preceq (i',j') \Leftrightarrow$ "$j < j'$" or "$j = j'$ and $i \leq i'$." Let (α^*, β^*) represent the minimum element of U. We have $x_{\alpha^*} = x_{\beta^*}$, and the value x_{β^*-1} never appears in $\{x_1, x_2, \ldots, x_{\beta^*-2}\}$, due to the fact that (α^*, β^*) is the minimum. Thus we see that adversary I_3 always succeeds upon event **InColl3**, provided that the values (α^*, β^*) are guessed correctly. Hence we have $\mathbf{Adv}_g^{\mathrm{mac}}(I_3) \geq \frac{1}{\binom{\ell}{2}} \cdot \Pr\big[\mathbf{InColl3}\big]$.

Collecting the six cases considered above, we get

$$\begin{aligned}
\mathbf{Adv}_g^{\mathrm{hoau}}(A) &\leq \Pr\big[\mathbf{Suff} \vee \mathbf{OutColl1} \vee \mathbf{OutColl2} \vee \mathbf{InColl1} \vee \mathbf{InColl2} \vee \mathbf{InColl3}\big] \\
&\leq \Pr\big[\mathbf{Suff}\big] + \Pr\big[\mathbf{OutColl1}\big] + \Pr\big[\mathbf{OutColl2}\big] \\
&\quad + \Pr\big[\mathbf{InColl1}\big] + \Pr\big[\mathbf{InColl2}\big] + \Pr\big[\mathbf{InColl3}\big] \\
&\leq \ell \cdot \mathbf{Adv}_g^{\mathrm{mac}}(S) + \ell^2 \cdot \mathbf{Adv}_g^{\mathrm{mac}}(O_1) + \tbinom{\ell}{2} \cdot \mathbf{Adv}_g^{\mathrm{mac}}(O_2) \\
&\quad + \ell^2 \cdot \mathbf{Adv}_g^{\mathrm{mac}}(I_1) + \tbinom{\ell}{2} \cdot \mathbf{Adv}_g^{\mathrm{mac}}(I_2) + \tbinom{\ell}{2} \cdot \mathbf{Adv}_g^{\mathrm{mac}}(I_3) \\
&\leq \tfrac{7\ell^2}{2} \cdot \mathbf{Adv}_g^{\mathrm{mac}}(t', q + 2\ell - 1),
\end{aligned}$$

where $t' = t + (2\ell - 1) \cdot T_g$. □

6 Description and Security of iCBC-MAC

In this section we define our iCBC-MAC scheme and provide its security results. Define the iCBC-MAC mode of operation $\boldsymbol{G}_k : \{0,1\}^* \to \{0,1\}^n$ as:

```
Algorithm $\boldsymbol{G}_k(M)$
$y \leftarrow H_k(1^b \| M)$; $\xi \leftarrow 0^n \| y \| 0^{b-2n}$; $\tau \leftarrow g_k(\xi)$; Output $\tau$.
```

Note that we add a special finalization step in the iteration. This step is crucial to the security of the scheme. Also, this (and only this) invocation is where privacy preservation is required. We remark that this construction requires $b \geq 2n$ (We shall discuss this constraint in Sect. 7). See Fig. 2 for an illustration of $\boldsymbol{G}$ (We have $\lambda = \lceil(|M|+1)/b\rceil$ in Fig. 2).

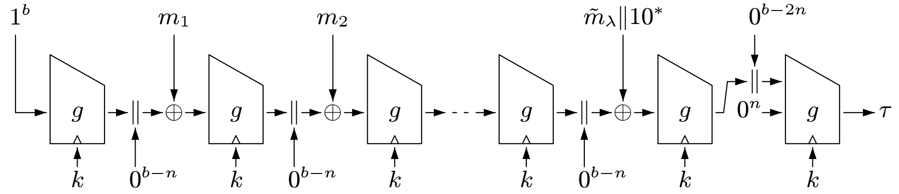


Fig. 2. Description of iCBC-MAC

```
$\boxed{\textbf{Game } \mathcal{G}_1(A)}$      Game $\mathcal{G}_2(A)$
$k \xleftarrow{\$} \{0,1\}^\kappa$; $y_1 \leftarrow g_k(1^b)$; $Y \leftarrow \varnothing$
$(M^*, \tau^*) \leftarrow A^{(\cdot)}$ where upon $A$'s $i$-th query $M$ do
  $M \leftarrow M\|10^*$; $m_1\|m_2\|\cdots\|m_\lambda \leftarrow M$
  For $j = 1, 2, \ldots, \lambda$ do $\bar{y}_j \leftarrow y_j\|0^{b-n}$; $x_j \leftarrow \bar{y}_j \oplus m_j$
    If $[x_j]^n = 0^n$ then Zero $\leftarrow 1$; Return 1 endIf
    $y_{j+1} \leftarrow g_k(x_j)$ endFor
  $y^{(i)} \leftarrow y_{\lambda+1}$; If $y^{(i)} \in Y$ then Coll $\leftarrow 1$; Return 1 endIf
  $Y \leftarrow Y \cup \{y^{(i)}\}$; $\xi \leftarrow 0^n\| \langle i \rangle_n \|0^{b-2n}$; $\boxed{\xi \leftarrow 0^n\|y^{(i)}\|0^{b-2n}}$
  $\tau \leftarrow g_k(\xi)$; Reply $\tau$ to $A$
$M^* \leftarrow M^*\|10^*$; $m_1^*\|m_2^*\|\cdots\|m_{\lambda^*}^* \leftarrow M^*$; $y_1^* \leftarrow y_1$
For $i = 1, 2, \ldots, \lambda^*$ do $\bar{y}_i^* \leftarrow y_i^*\|0^{b-n}$; $x_i^* \leftarrow \bar{y}_i^* \oplus m_i^*$
  If $[x_i^*]^n = 0^n$ then Zero $\leftarrow 1$; Return 1 endIf
  $y_{i+1}^* \leftarrow g_k(x_i^*)$ endFor
$y^{(q+1)} \leftarrow y_{\lambda^*+1}^*$; If $y^{(q+1)} \in Y$ then Coll $\leftarrow 1$; Return 1 endIf
$\xi^* \leftarrow 0^n\|y^{(q+1)}\|0^{b-2n}$; $\boxed{\text{If } g_k(\xi^*) = \tau^* \text{ then } \textbf{Forge} \leftarrow 1 \text{ endIf}}$; Return 0
```

Fig. 3. Game $\mathcal{G}_1$ with boxed statements and game $\mathcal{G}_2$ without the boxed statements

Theorem 1 (Security of iCBC-MAC). *Let $\boldsymbol{G}_k : \{0,1\}^* \to \{0,1\}^n$ be the iCBC-MAC mode constructed of a compression function $g_k : \{0,1\}^b \to \{0,1\}^n$ with $b \geq 2n$. If g is a PP-MAC, then so is $\boldsymbol{G}$.*[6] *Specifically, we have*

$$\mathbf{Adv}_{\boldsymbol{G}}^{\mathrm{mac}}(t, q, \ell) \leq \mathbf{Adv}_{g}^{\mathrm{mac}}(t', 4\ell q) + \mathbf{Adv}_{g}^{\mathrm{ind}}(t', 4\ell q) + 4\ell^2 q^2 \cdot \mathbf{Adv}_{g}^{\mathrm{mac}}(t'', 2\ell + q), \tag{2}$$

$$\mathbf{Adv}_{\boldsymbol{G}}^{\mathrm{ind}}(t, q, \ell) \leq 3 \cdot \mathbf{Adv}_{g}^{\mathrm{ind}}(t', 4\ell q) + 9\ell^2 q^2 \cdot \mathbf{Adv}_{g}^{\mathrm{mac}}(t'', 2\ell + q), \tag{3}$$

where $t' = t + 4\ell q \cdot T_g$ and $t'' = t + (2\ell + q) \cdot T_g$. Also, if g is a PRF, then so is $\boldsymbol{G}$. Specifically, we have

$$\mathbf{Adv}_{\boldsymbol{G}}^{\mathrm{prf}}(t, q, \ell) \leq \mathbf{Adv}_{g}^{\mathrm{prf}}(t', 4\ell q) + \tfrac{9\ell^2 q^2}{4} \cdot \tfrac{1}{2^n}. \tag{4}$$

[6] Our construction requires that the underlying compression function be PP-MAC (rather than just a secure MAC). The construction would not be secure on the sole assumption that the compression function is a secure MAC—The counterexample of attacks on CBC-MACs by [7] can be extended to iCBC-MAC.

Proof. As a typical case we prove (2). Proofs for (3) and (4) are sketched in App. A and B. So let A be an adversary, attacking $\boldsymbol{G}$ in the MAC sense, having running time at most t, making exactly (rather than at most) q queries to its oracle, each query being at most ℓ blocks, outputting a forgery attempt (M^*, τ^*), M^* being at most ℓ blocks. Without loss of generality we assume that A never repeats a query.

Consider the two games defined in Fig. 3. We see that if adversary A succeeds in forgery, then at least one of the events **Forge**, **Zero** or **Coll** occurs in game $\mathcal{G}_1$. Note that these events are so defined as they are disjoint (As soon as one of the events occurs the game terminates), and therefore

$$\begin{aligned}\mathbf{Adv}_{\boldsymbol{G}}^{\mathrm{mac}}(A) &\le \Pr\bigl[\mathcal{G}_1(A) \text{ sets } \mathbf{Forge}, \mathbf{Zero} \text{ or } \mathbf{Coll}\bigr] \\ &= \Pr\bigl[\mathcal{G}_1(A) \text{ sets } \mathbf{Forge}\bigr] + \Pr\bigl[\mathcal{G}_1(A) \text{ sets } \mathbf{Zero} \text{ or } \mathbf{Coll}\bigr] \\ &\qquad - \Pr\bigl[\mathcal{G}_2(A) \text{ sets } \mathbf{Zero} \text{ or } \mathbf{Coll}\bigr] + \Pr\bigl[\mathcal{G}_2(A) \text{ sets } \mathbf{Zero} \text{ or } \mathbf{Coll}\bigr] \\ &= \Pr\bigl[\mathbf{Forge}\bigr] + \Pr\bigl[\mathcal{G}_1(A) = 1\bigr] - \Pr\bigl[\mathcal{G}_2(A) = 1\bigr] \\ &\qquad + \Pr\bigl[\mathcal{G}_2(A) \text{ sets } \mathbf{Zero}\bigr] + \Pr\bigl[\mathcal{G}_2(A) \text{ sets } \mathbf{Coll}\bigr].\end{aligned}$$

In the following we bound each of these quantities as

$$\begin{aligned}\Pr[\mathbf{Forge}] &\le \mathbf{Adv}_g^{\mathrm{mac}}(t', \ell q + \ell + q + 1) \le \mathbf{Adv}_g^{\mathrm{mac}}(t', 4\ell q), &(5)\\ \Pr\bigl[\mathcal{G}_1(A) = 1\bigr] - \Pr\bigl[\mathcal{G}_2(A) = 1\bigr] &\le \mathbf{Adv}_g^{\mathrm{ind}}(t', \ell q + \ell + q + 1) \\ &\le \mathbf{Adv}_g^{\mathrm{ind}}(t', 4\ell q), &(6)\\ \Pr\bigl[\mathcal{G}_2(A) \text{ sets } \mathbf{Zero}\bigr] &\le (q+1) \cdot \tbinom{\ell+1}{2} \cdot \mathbf{Adv}_g^{\mathrm{mac}}(t'', \ell + q) \\ &\le \tfrac{\ell^2 q^2}{2} \cdot \mathbf{Adv}_g^{\mathrm{mac}}(t'', 2\ell + q), &(7)\\ \Pr\bigl[\mathcal{G}_2(A) \text{ sets } \mathbf{Coll}\bigr] &\le \tbinom{q+1}{2} \cdot \mathbf{Adv}_g^{\mathrm{hoau}}(t''', q, \ell) \\ &\le \tfrac{7\ell^2 q^2}{2} \cdot \mathbf{Adv}_g^{\mathrm{mac}}(t'', 2\ell + q), &(8)\end{aligned}$$

where $t' = t + (\ell q + \ell + q + 1) \cdot T_g$, $t'' = t + (\ell + q) \cdot T_g$ and $t''' = t + q \cdot T_g$.

We begin with proving (5). Consider the adversary F defined in Fig. 4. Observe that the event **Forge**, which is disjoint with **Zero** and with **Coll** in game $\mathcal{G}_1$ by definition, assures that adversary F succeeds in producing a forgery. Thus we get $\Pr\bigl[\mathbf{Forge}\bigr] \le \mathbf{Adv}_g^{\mathrm{mac}}(F)$. The running time of F is at most $t + (\ell q + \ell + q + 1) \cdot T_g$, and F makes at most $\ell q + \ell + q + 1$ queries. This proves (5).

We next prove (6). Consider the adversary D defined in Fig. 4. If the oracle $\mathcal{O}$ with which adversary D interacts is the Left g_k oracle, then running $D^{\mathcal{O}}$ coincides with $\mathcal{G}_1(A)$. Similarly, if oracle $\mathcal{O}$ is the Right g_k oracle, then $D^{\mathcal{O}}$ coincides with $\mathcal{G}_2(A)$. Note that queries made by D are legitimate, because D terminates as soon as **Zero** or **Coll** occurs. Thus we get $\Pr\bigl[\mathcal{G}_1(A) = 1\bigr] - \Pr\bigl[\mathcal{G}_2(A) = 1\bigr] = \Pr\bigl[D^{\mathrm{Left}} = 1\bigr] - \Pr\bigl[D^{\mathrm{Right}} = 1\bigr] = \mathbf{Adv}_g^{\mathrm{ind}}(D) \le \mathbf{Adv}_g^{\mathrm{ind}}(t', \ell q + \ell + q + 1)$. This proves (6).

We proceed to proving (7). We use the adversary Z defined in Fig. 5. Consider the event **Zero** in game $\mathcal{G}_2$. This occurs at A's α-th query for some $\alpha \in [1, q+1]$.

```
Adversary $F$    $\boxed{\textbf{Adversary } D}$
$Y \leftarrow \varnothing$; Make a query $\boldsymbol{y_1 \leftarrow g_k(1^b)}$ $\boxed{y_1 \leftarrow \mathcal{O}(1^b, 1^b)}$
$(M^*, \tau^*) \leftarrow A^{(\cdot)}$ where upon $A$'s $i$-th query $M$ do
   $M \leftarrow M\|10^*$; $m_1\|m_2\|\cdots\|m_\lambda \leftarrow M$
   For $j = 1, 2, \ldots, \lambda$ do
      $\bar{y}_j \leftarrow y_j\|0^{b-n}$; $x_j \leftarrow \bar{y}_j \oplus m_j$; $\boxed{\text{If } [x_j]^n = 0^n \text{ then output 1 endIf}}$
      Make a query $\boldsymbol{y_{j+1} \leftarrow g_k(x_j)}$ $\boxed{y_{j+1} \leftarrow \mathcal{O}(x_j, x_j)}$ endFor
   $\boxed{\text{If } y_{\lambda+1} \in Y \text{ then output 1 else } Y \leftarrow Y \cup \{y_{\lambda+1}\} \text{ endIf}}$
   $\xi \leftarrow 0^n\|y_{\lambda+1}\|0^{b-2n}$; $\xi' \leftarrow 0^n\| \langle i \rangle_n \|0^{b-2n}$
   Make a query $\boldsymbol{\tau \leftarrow g_k(\xi)}$ $\boxed{\tau \leftarrow \mathcal{O}(\xi, \xi')}$; Reply $\tau$ to $A$
$M^* \leftarrow M^*\|10^*$; $m_1^*\|m_2^*\|\cdots\|m_{\lambda^*}^* \leftarrow M^*$; $y_1^* \leftarrow y_1$
For $i = 1, 2, \ldots, \lambda^*$ do
   $\bar{y}_i^* \leftarrow y_i^*\|0^{b-n}$; $x_i^* \leftarrow \bar{y}_i^* \oplus m_i^*$; $\boxed{\text{If } [x_i]^n = 0^n \text{ then output 1 endIf}}$
   Make a query $\boldsymbol{y_{i+1}^* \leftarrow g_k(x_i^*)}$ $\boxed{y_{i+1}^* \leftarrow \mathcal{O}(x_i^*, x_i^*)}$ endFor
$\boxed{\text{If } y_{\lambda^*+1}^* \in Y \text{ then output 1 else output 0 endIf}}$
$\xi^* \leftarrow 0^n\|y_{\lambda^*+1}^*\|0^{b-2n}$; Output $(\xi^*, \tau^*)$ as a forgery
```

Fig. 4. Bold statements apply only to the definition of forger F while boxed statements only to distinguisher D

```
Adversary $Z$
Choose $\alpha \xleftarrow{\$} [1, q+1]$
Run $A^{(\cdot)}$ and upon its $i$-th query $(i < \alpha)$ do
   $\xi \leftarrow 0^n\| \langle i \rangle_n \|0^{b-2n}$; Make a query $\tau \leftarrow g_k(\xi)$; Reply $\tau$ to $A$
   until $A$ makes its $\alpha$-th query $M$        // This is $(M, \tau)$ when $\alpha = q+1$
$M \leftarrow M\|10^*$; $m_1\|m_2\|\cdots\|m_\lambda \leftarrow M$; Choose $\gamma, \beta \xleftarrow{\$} [0, \lambda]$ so that $\gamma < \beta$; $x_0 \leftarrow 1^b$
For $j = 1, 2, \ldots, \gamma$ do
   Make a query $y_j \leftarrow g_k(x_{j-1})$; $\bar{y}_j \leftarrow y_j\|0^{b-n}$; $x_j \leftarrow \bar{y}_j \oplus m_j$ endFor
Output $(x_\gamma, [m_\beta]^n)$ as a forgery
```

```
Adversary $C$
Choose $\alpha, \beta \xleftarrow{\$} [1, q+1]$ so that $\alpha < \beta$
Run $A^{(\cdot)}$ and upon its $i$-th query $M_i$ $(i < \beta)$ do
   If $i = \alpha$ then $M \leftarrow M_i$ endIf
   $\xi \leftarrow 0^n\| \langle i \rangle_n \|0^{b-2n}$; Make a query $\tau \leftarrow g_k(\xi)$; Reply $\tau$ to $A$
   until $A$ makes its $\beta$-th query $M_\beta$        // This is $(M_\beta, \tau)$ when $\beta = q+1$
$M' \leftarrow M_\beta$; Output $(M, M')$
```

Fig. 5. Definition of adversaries Z and C

Let M be the α-th query, and put $m_1\|m_2\|\cdots\|m_\lambda \leftarrow M$. Let $x_1, x_2, \ldots, x_\lambda$ be the variables as usual. Then the event **Zero** occurs at x_β for some $\beta \in [1, \lambda]$. In such a case we have $g_k(x_{\beta-1}) = [m_\beta]^n$. Now let $\gamma \in [0, \beta-1]$ be the smallest index such that $x_\gamma = x_{\beta-1}$. Then we observe that adversary Z wins if it correctly guesses the values α, β, γ upon event **Zero** in game $\mathcal{G}_2$. Therefore, we get $\mathbf{Adv}_g^{\mathrm{mac}}(Z) \geq \frac{1}{q+1} \cdot \frac{1}{\binom{\ell+1}{2}} \cdot \Pr\bigl[\mathcal{G}_2(A) \text{ sets } \textbf{Zero}\bigr]$. Z makes at most $q + \ell$ queries and runs in time at most $t + (\ell + q) \cdot T_g$. This proves (7).

Lastly we prove (8). Consider the adversary C defined in Fig. 5. Given two integers i, j such that $1 \leq i < j \leq q+1$, let $\mathbf{Coll}_{i,j}$ denote the event that the flag **Coll** in game $\mathcal{G}_2$ gets set at A's j-th query in such a way as $y^{(i)} = y^{(j)}$. Note that the events $\mathbf{Coll}_{i,j}$ are disjoint. Now observe that adversary C wins in the HOAU sense if the guessed values α, β coincide with i, j, because the message pair (M, M') is guaranteed to be fresh owing to the fact that **Coll** and **Zero** are disjoint in game $\mathcal{G}_2$ by definition. Therefore, $\mathbf{Adv}_g^{\mathrm{hoau}}(C) \geq \Pr\big[\bigvee_{i,j}\big(\mathbf{Coll}_{i,j} \wedge (i,j) = (\alpha,\beta)\big)\big] = \sum_{i,j} \Pr\big[\mathbf{Coll}_{i,j} \wedge (i,j) = (\alpha,\beta)\big] = \sum_{i,j} \Pr\big[\mathbf{Coll}_{i,j}\big] \cdot \Pr\big[(i,j) = (\alpha,\beta)\big] = \frac{1}{\binom{q+1}{2}} \cdot \Pr\big[\mathcal{G}_2(A) \text{ sets } \mathbf{Coll}\big]$. C makes at most q queries and runs in time at most $t + q \cdot T_g$. This proves (8). □

7 Further Optimization of iCBC-MAC Construction

In this section we mention three techniques to alter the iCBC-MAC construction. The first two are for better efficiency, while the third one is for more general applicability.

Reducing the IV Size. In our iCBC-MAC scheme, the first block is computed as $y_1 \leftarrow g_k(1^b)$, and it takes no bits of a message. We can improve performance by replacing $y \leftarrow H_k(1^b \| M)$ with $y \leftarrow H_k(1^r \| M)$ in the definition of $\boldsymbol{G}$. Here, any positive integer r such that $1 \leq r \leq b$ works. A preferred choice might be $r = 1$, 8, 32, *etc.*.

Improving the Padding 1*0. When the original size of a message happens to be exactly equal to a multiple of b bits, the padding $M \leftarrow M \| 10^*$ produces an extra block consisting of 10^{b-1}, which might be undesirable. We can get rid of this extra block by "tweaking" the last invocation to g. Recall that in iCBC-MAC the last invocation to g is done as $\xi \leftarrow 0^n \| y_{\lambda+1} \| 0^{b-2n}$, $\tau \leftarrow g_k(\xi)$. The following is an example of such tweaking: (i) If $|M|$ is not a multiple of b, then compute τ as in the original $\boldsymbol{G}$, and (ii) if $|M|$ happens to be equal to a multiple of b, then compute $\xi \leftarrow 0^{n-1}1 \| y_\lambda \| 0^{b-2n}$, $\tau \leftarrow g_k(\xi)$ instead.

Getting Rid of the Constraint $\boldsymbol{b \geq 2n}$. The iCBC-MAC construction requires $b \geq 2n$. This is not a serious limitation in practice, but one might want to eliminate this constraint. This can be resolved at the expense of performance: Given a message $M \in \{0,1\}^*$, pad $M \leftarrow M \| 10^*$ in such a way as the length of the padded message becomes a multiple of $b - 1$ bits. Divide the message as $m_1 \| m_2 \| \cdots \| m_\lambda \leftarrow M$ with the block length of $b - 1$ bits, which yields $|m_1| = |m_2| = \cdots = |m_\lambda| = b - 1$. Put $M' \leftarrow m_1 \| 1 \| m_2 \| 1 \| \cdots \| m_{\lambda-1} \| 1 \| m_\lambda$ so that $|M'| = b\lambda - 1$. Compute $y \leftarrow H_k(M')$, $\xi \leftarrow y \| 0^{b-n}$ and $\tau \leftarrow g_k(\xi)$.

8 Performance Issues

Both κ and n correspond to security parameters. When $\kappa = n$, the iCBC iteration should be as efficient as the unkeyed Merkle-Damgård construction if we neglect the cost of an xor operation per block. It makes sense to set $\kappa < n$, especially $2\kappa = n$, due to the birthday attacks [21] on MACs and wide-pipe designs [12] for hash functions (The final tag size may be truncated in these cases). For example, consider the case of sha-256 : $\{0,1\}^{256+512} \to \{0,1\}^{256}$ with a 128-bit key. HMAC processes 512 bits per block in such a case, but iCBC-MAC can process $256 + 512 - 128 = 640$ bits per invocation, yielding a 25% increase in performance. Additionally, note that the single-key structure of the iCBC construction accommodates primitives that might be equipped with a heavy key-schedule algorithm like block ciphers.

Acknowledgments

The author would like to thank ICISC 2008 anonymous referees for their valuable comments. The author is grateful especially to one of the referees for conducting a thorough review of the manuscript and pointing out some typos in the proofs. The author also wishes to express his appreciation to Liting Zhang at the conference for pointing out a couple of typos in the security definition.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
2. Bellare, M.: New proofs for NMAC and HMAC: Security without collision resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
3. Kim, J., Biryukov, A., Preneel, B., Hong, S.: On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1 (Extended abstract). In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
4. Contini, S., Yin, Y.L.: Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
5. Fouque, P.A., Leurent, G., Nguyen, P.Q.: Full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
6. Wang, L., Ohta, K., Kunihiro, N.: New key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 237–253. Springer, Heidelberg (2008)
7. An, J.H., Bellare, M.: Constructing VIL-MACs from FIL-MACs: Message authentication under weakened assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 252–269. Springer, Heidelberg (1999)
8. Maurer, U.M., Sjödin, J.: Single-key AIL-MACs from any FIL-MAC. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 472–484. Springer, Heidelberg (2005)
9. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)

10. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
11. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
12. Lucks, S.: A failure-friendly design principle for hash functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
13. Bellare, M., Ristenpart, T.: Hash functions in the dedicated-key setting: Design choices and MPP transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)
14. Fischlin, M.: Security of NMAC and HMAC based on non-malleability. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 138–154. Springer, Heidelberg (2008)
15. Yasuda, K.: Boosting Merkle-Damgård hashing for message authentication. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 216–231. Springer, Heidelberg (2007)
16. Hirose, S., Park, J.H., Yun, A.: A simple variant of the Merkle-Damgård scheme with a permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg (2007)
17. Dodis, Y., Pietrzak, K., Puniya, P.: A new mode of operation for block ciphers and length-preserving MACs. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 198–219. Springer, Heidelberg (2008)
18. Iwata, T., Kurosawa, K.: OMAC: One-key CBC MAC. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (2003)
19. Patel, S.: An efficient MAC for short messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 353–368. Springer, Heidelberg (2003)
20. Bellare, M., Goldreich, O., Mityagin, A.: The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive: Report 2004/304 (2004)
21. Preneel, B., van Oorschot, P.C.: MDx-MAC and building fast MACs from hash functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)

A Proof Sketch of (3)

Let A be an ind-adversary attacking $\boldsymbol{G}$, having running time at most t, making at most q queries, each query being at most ℓ blocks. Without loss of generality we assume that A never repeats a query. Consider the five games defined in Fig. 6. Of the five games defined, game $\mathcal{G}_{\mathrm{L}}$ coincides with the Left game for A, while game $\mathcal{G}_{\mathrm{R}}$ does with the Right game. We let **Zero** and **Coll** be the events as usual. We have

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ind}}_{\boldsymbol{G}}(A) &= \Pr\bigl[A^{\mathrm{Left}} = 1\bigr] - \Pr\bigl[A^{\mathrm{Right}} = 1\bigr]\\ &= \Pr\bigl[A^{\mathrm{Left}} = 1\bigr] - \Pr\bigl[\mathcal{G}^{\times}_{\mathrm{L}}(A) = 1\bigr] \qquad (9)\\ &\quad + \Pr\bigl[\mathcal{G}^{\times}_{\mathrm{L}}(A) = 1\bigr] - \Pr\bigl[\mathcal{G}^{\times}_{\mathrm{R}}(A) = 1\bigr] \qquad (10)\\ &\quad + \Pr\bigl[\mathcal{G}^{\times}_{\mathrm{R}}(A) = 1\bigr] - \Pr\bigl[A^{\mathrm{Right}} = 1\bigr]. \qquad (11)\end{aligned}$$

```
$\textbf{Game } \mathcal{G}_{\mathrm{L}}(A)$ and $\boxed{\textbf{Game } \mathcal{G}_{\mathrm{L}}^{\times}(A)}$
$k \xleftarrow{\$} \{0,1\}^{\kappa}$; $Y_{\mathrm{L}} \leftarrow \varnothing$; $Y_{\mathrm{R}} \leftarrow \varnothing$
$\sigma \leftarrow A^{(\cdot,\cdot)}$ where on query $(M_{\mathrm{L}}, M_{\mathrm{R}})$ do
  $y_{\mathrm{L}} \leftarrow H_k(M_{\mathrm{L}})$; $y_{\mathrm{R}} \leftarrow H_k(M_{\mathrm{R}})$
  If $y_{\mathrm{L}} \in Y_{\mathrm{L}}$ or $y_{\mathrm{R}} \in Y_{\mathrm{R}}$
    then $\mathbf{Coll} \leftarrow 1$; $\boxed{\text{Return } 1}$ endIf
  If $y_{\mathrm{L}} = 0^n$ or $y_{\mathrm{R}} = 0^n$
    then $\mathbf{Zero} \leftarrow 1$; $\boxed{\text{Return } 1}$ endIf
  $Y_{\mathrm{L}} \leftarrow Y_{\mathrm{L}} \cup \{y_{\mathrm{L}}\}$; $Y_{\mathrm{R}} \leftarrow Y_{\mathrm{R}} \cup \{y_{\mathrm{R}}\}$
  $\xi \leftarrow 0^n \| y_{\mathrm{L}} \| 0^{b-2n}$; $\tau \leftarrow g_k(\xi)$
  Reply $\tau$ to $A$
Return $\sigma$
```

```
$\textbf{Game } \mathcal{G}_{\mathrm{R}}(A)$ and $\boxed{\textbf{Game } \mathcal{G}_{\mathrm{R}}^{\times}(A)}$
$k \xleftarrow{\$} \{0,1\}^{\kappa}$; $Y_{\mathrm{L}} \leftarrow \varnothing$; $Y_{\mathrm{R}} \leftarrow \varnothing$
$\sigma \leftarrow A^{(\cdot,\cdot)}$ where on query $(M_{\mathrm{L}}, M_{\mathrm{R}})$ do
  $y_{\mathrm{L}} \leftarrow H_k(M_{\mathrm{L}})$; $y_{\mathrm{R}} \leftarrow H_k(M_{\mathrm{R}})$
  If $y_{\mathrm{L}} \in Y_{\mathrm{L}}$ or $y_{\mathrm{R}} \in Y_{\mathrm{R}}$
    then $\mathbf{Coll} \leftarrow 1$; $\boxed{\text{Return } 1}$ endIf
  If $y_{\mathrm{L}} = 0^n$ or $y_{\mathrm{R}} = 0^n$
    then $\mathbf{Zero} \leftarrow 1$; $\boxed{\text{Return } 1}$ endIf
  $Y_{\mathrm{L}} \leftarrow Y_{\mathrm{L}} \cup \{y_{\mathrm{L}}\}$; $Y_{\mathrm{R}} \leftarrow Y_{\mathrm{R}} \cup \{y_{\mathrm{R}}\}$
  $\xi \leftarrow 0^n \| y_{\mathrm{R}} \| 0^{b-2n}$; $\tau \leftarrow g_k(\xi)$
  Reply $\tau$ to $A$
Return $\sigma$
```

```
$\textbf{Game } \mathcal{G}_3(A)$
$k \xleftarrow{\$} \{0,1\}^{\kappa}$; $Y_{\mathrm{L}} \leftarrow \varnothing$; $Y_{\mathrm{R}} \leftarrow \varnothing$; $i \leftarrow 0$
$A^{(\cdot,\cdot)}$ where on query $(M_{\mathrm{L}}, M_{\mathrm{R}})$ do
  $y_{\mathrm{L}} \leftarrow H_k(M_{\mathrm{L}})$; $y_{\mathrm{R}} \leftarrow H_k(M_{\mathrm{R}})$
  If $y_{\mathrm{L}} \in Y_{\mathrm{L}}$ or $y_{\mathrm{R}} \in Y_{\mathrm{R}}$ then $\mathbf{Coll} \leftarrow 1$; Return 1 endIf
  If $y_{\mathrm{L}} = 0^n$ or $y_{\mathrm{R}} = 0^n$ then $\mathbf{Zero} \leftarrow 1$; Return 1 endIf
  $Y_{\mathrm{L}} \leftarrow Y_{\mathrm{L}} \cup \{y_{\mathrm{L}}\}$; $Y_{\mathrm{R}} \leftarrow Y_{\mathrm{R}} \cup \{y_{\mathrm{R}}\}$; $i \leftarrow i+1$; $\xi \leftarrow 0^n \| \langle i \rangle_n \| 0^{b-2n}$; $\tau \leftarrow g_k(\xi)$
  Reply $\tau$ to $A$
Return 0
```

Fig. 6. Definitions of five games. Note that games $\mathcal{G}_{\mathrm{L}}^{\times}$, $\mathcal{G}_{\mathrm{R}}^{\times}$ are with boxed statements and games $\mathcal{G}_{\mathrm{L}}$, $\mathcal{G}_{\mathrm{R}}$ are without them.

We bound (9) as

$$\begin{aligned}
\Pr\big[A^{\mathrm{Left}} = 1\big] - \Pr\big[\mathcal{G}_{\mathrm{L}}^{\times}(A) = 1\big] &\le \Pr\big[A^{\mathrm{Left}} \text{ sets } \mathbf{Zero} \text{ or } \mathbf{Coll}\big] \\
&= \Pr\big[A^{\mathrm{Left}} \text{ sets } \mathbf{Zero} \text{ or } \mathbf{Coll}\big] \\
&\qquad - \Pr\big[\mathcal{G}_3(A) \text{ sets } \mathbf{Zero} \text{ or } \mathbf{Coll}\big] \\
&\qquad\quad + \Pr\big[\mathcal{G}_3(A) \text{ sets } \mathbf{Zero} \text{ or } \mathbf{Coll}\big] \\
&\le \mathbf{Adv}_g^{\mathrm{ind}}(t', 2\ell q + q + 1) \\
&\qquad + 2q \cdot \tbinom{\ell+1}{2} \cdot \mathbf{Adv}_g^{\mathrm{mac}}(t'', \ell + q) \\
&\qquad\quad + 2 \cdot \tbinom{q}{2} \cdot \mathbf{Adv}_g^{\mathrm{hoau}}(t''', q, \ell) \\
&\le \mathbf{Adv}_g^{\mathrm{ind}}(t', 4\ell q) + \tfrac{9\ell^2 q^2}{2} \cdot \mathbf{Adv}_g^{\mathrm{mac}}(t'', 2\ell + q),
\end{aligned}$$

where $t' = t + (2\ell q + q + 1) \cdot T_g$, $t'' = t + (\ell + q) \cdot T_g$ and $t''' = t + q \cdot T_g$. We do a similar analysis for (11).

To bound (10), we see that

$$\Pr\big[\mathcal{G}_{\mathrm{L}}^{\times}(A) = 1\big] - \Pr\big[\mathcal{G}_{\mathrm{R}}^{\times}(A) = 1\big] \le \mathbf{Adv}_g^{\mathrm{ind}}(t', 2\ell q + q + 1) \le \mathbf{Adv}_g^{\mathrm{ind}}(t', 4\ell q).$$

Thus we have shown that $\boldsymbol{G}$ is privacy-preserving on the assumption that g is a PP-MAC. □

B Proof Sketch of (4)

Let A be a prf-adversary attacking $\boldsymbol{G}$, having running time at most t, making at most q queries, each query being at most ℓ blocks. Let $\gamma : \{0,1\}^b \to \{0,1\}^n$ be a random function, and construct $\boldsymbol{\Gamma} : \{0,1\}^* \to \{0,1\}^n$ as:

```
**Algorithm** $\boldsymbol{\Gamma}(M)$
$y_1 \leftarrow \gamma(1^b)$
$M \leftarrow M \| 10^*$; $m_1 \| m_2 \| \cdots \| m_\lambda \leftarrow M$
For $i = 1, 2, \ldots, \lambda$ do $\bar{y}_i \leftarrow y_i \| 0^{b-n}$; $x_i \leftarrow \bar{y}_i \oplus m_i$; $y_{i+1} \leftarrow \gamma(x_i)$ endFor
$\xi \leftarrow 0^n \| y_{\lambda+1} \| 0^{b-2n}$; $\tau \leftarrow \gamma(\xi)$; Output $\tau$.
```

Now we see that

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\boldsymbol{G}}(A) &= \Pr\big[A^{\mathrm{Real}} = 1\big] - \Pr\big[A^{\mathrm{Random}} = 1\big] \\ &= \Pr\big[A^{\boldsymbol{G}} = 1\big] - \Pr\big[A^{\boldsymbol{\Gamma}} = 1\big] \\ &\qquad + \Pr\big[A^{\boldsymbol{\Gamma}} = 1\big] - \Pr\big[A^{\mathrm{Random}} = 1\big].\end{aligned}$$

Then we get

$$\Pr\big[A^{\boldsymbol{G}} = 1\big] - \Pr\big[A^{\boldsymbol{\Gamma}} = 1\big] \le \mathbf{Adv}^{\mathrm{prf}}_{g}(t', \ell q + q),$$

where $t' = t + (\ell q + q) \cdot T_g$.

Let **Zero** and **Coll** be defined similarly as in game $\mathcal{G}_1$. Observe that $\boldsymbol{\Gamma}$ behaves just like a truly random function unless **Zero** or **Coll** occurs. Hence

$$\begin{aligned}\Pr\big[A^{\boldsymbol{\Gamma}} = 1\big] - \Pr\big[A^{\mathrm{Random}} = 1\big] &\le \Pr\big[A^{\boldsymbol{\Gamma}} \text{ sets } \mathbf{Zero} \text{ or } \mathbf{Coll}\big] \\ &\le q \cdot \tbinom{\ell+1}{2} \cdot \tfrac{1}{2^n} + \tfrac{7\ell^2}{2} \cdot \tbinom{q}{2} \cdot \tfrac{1}{2^n} \\ &\le \tfrac{9\ell^2 q^2}{4} \cdot \tfrac{1}{2^n}.\end{aligned}$$

Thus we have proven that $\boldsymbol{G}$ is a PRF under the condition that g is a PRF. □

Extended Models for Message Authentication

Liting Zhang[1,2], Wenling Wu[1], and Peng Wang[2]

[1] State Key Laboratory of Information Security
Institute of Software, Chinese Academy of Sciences, Beijing 100190, P.R. China
{zhangliting,wwl}@is.iscas.ac.cn
[2] State Key Laboratory of Information Security
Graduate University of Chinese Academy of Sciences, Beijing 100049, P.R. China
wp@is.ac.cn

Abstract. In recent years, several side channel attacks have been given to some provably secure Message Authentication (MA) schemes. These side channel attacks help adversaries to get some information about secret values (such like internal states) in MA-schemes, which is beyond the original models consider about, so the provable security completely lose. To fix this problem, we extend the original models for message authentication, taking the information about secret values in MA-schemes into account. The extended models can not only provide a framework under which one can discuss security of MA-schemes facing side channel attacks, but also give us an insight view of MA-schemes. As an example, we consider the security of $f9$ (a MA-scheme in 3GPP) and its variants in an extended model. The result helps us to know $f9$ better, e.g. how to use it safely and what measures need to be taken in case of potential attacks.

Keywords: Security Model, Message Authentication, Side Channel Attack, Provable Security.

1 Introduction

1.1 Message Authentication

Message authentication is to provide data integrity and data origin authentication, and it is widely used in practice, e.g. the Internet security protocols like SSL, SSH and IPSEC. Normally, a Message Authentication (MA) scheme is a symmetric primitive, which implies that the users should agree on a secret key K in advance. When the sender wants to send a message M, he/she runs the tag generation algorithm TG in a MA-scheme with K and M as input, and gets the output as Tag. Then he/she sends M and Tag to the receiver. On receipt of them, the receiver runs the verification algorithm VF in the same MA-scheme, verifying whether Tag is corresponding to M.

1.2 The Security of MA-schemes

Security Models. A MA-scheme is said to be secure, if it is unforgeable. Roughly speaking, adversaries have access to the MA-scheme, and they can adaptively

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 286–301, 2009.

input messages they choose into its TG algorithm, getting the corresponding tags. In the end, they are asked to output a pair of legal (M, T), where M's tag equals to T, and M has never been input to TG or T has never been output by VF in the MA-scheme. If all the adversaries can do this within only negligible probability, the MA-scheme is deemed to be secure, or unforgeable.

Sometimes, adversaries are allowed to challenge more than once. That is, they can try a series of challenging (M^i, T^i) and are deemed to be successful if at least one of them is legal. Thus, according to whether M is new or T is new in the forgery and whether the adversaries are allowed to challenge more than once or not, there are four specific security models. The relationships among them have been analyzed in [1], but we point out that all these four security models treat the MA-schemes as a whole. That is, the adversaries are not allowed to get information about the secret values (such like internal states, etc.) in the MA-schemes.

In the four security models mentioned above, many MA-schemes have been proved to be secure, usually bounding their security to the total length of queries, such like [2].

Attacks on MA-schemes. On the other hand, many attacks have been given to those provably secure MA-schemes, and almost all these attacks take the following strategy: (1) try to find internal collisions by birthday paradox, and then (2) transform these internal collisions into forgeries, such as done in [3,4].

However, there is another kind of attack emerging in the past few years. As far as our present knowledge, Okeya and Iwata are the first to give side channel attacks to MA-schemes [5]. Similar to the attacks mentioned above, their attacks try to get some information (partial or complete values) about the secret values (including internal states and secret masks) in MA-schemes first, and then use them to make forgeries successfully. However, the way they get information about the secret values is not by birthday paradox, but by some kinds of side channel attacks, i.e. Simple Power Analysis (SPA) and Differential Power Analysis (DPA). Thanks to these two special methods, their attacks are much more efficient than the others. Since then, several side channel attacks have been given to other MA-schemes [6,7], and what is more, all of them are generic, because they take no advantage of the underlying primitives but assume them to be secure against side channel attacks.

A common character of all the attacks mentioned above is that they try to get some information (collisions, partial or complete values, and so on) about the secret values (including internal states, etc.) in MA-schemes first, and then make forgeries by the information they get.

Thus, it seems that the information about secret values in MA-schemes have some influence on the security of MA-schemes. However, it is natural to ask, how important are they? What would happen to MA-schemes if any one of them leaks out? Are there some MA-schemes whose security keeps unchanged even if some of them were obtained by adversaries?

Work to be done. Now we have got an overview of the issues in message authentication: on one hand, the current security models all make the MA-scheme a block box to adversaries, not allowing adversaries to get information about the secret values (including internal states, etc.) in MA-schemes; on the other hand, many attacks to MA-schemes have been given based on some information (partial or complete values, and so on) about the secret values in the MA-schemes, usually by side channel attacks. So, the current models are not sufficient to discuss the security of MA-schemes now, and it is necessary to introduce new security models for MA-schemes, taking the information about the secret values in MA-schemes into account.

1.3 Our Contributions

In the remaining of this paper, we extend the basic security models to some new ones, consider the influence of information about the secret values, and study how important they are on the security of MA-schemes. In doing this, we first give some general models to illustrate the idea, and then give a specific one, studying the security of $f9$ in 3GPP as an example.

The general models not only answer the question that why those MA-schemes provably secure in the original models still have been attacked, but also provide a framework of security models, under which one can study the security of their MA-schemes in different aspects freely and naturally, and in the end get a better understanding of their MA-schemes.

The discussions of $f9$-like MA-schemes in the specific model help us to know $f9$ better. In a short, we get the knowledge that how to use $f9$ safely and what measures should be taken on it in case of potential attacks.

2 Preliminaries

2.1 Notations and Definitions

Suppose A is a set, then $\#A$ denotes the number of different elements in set A, and $x \overset{\$}{\leftarrow} A$ denotes that x is chosen from set A uniformly at random. In the rest of this paper, every time we say "random", we mean "uniformly random". When A is an algorithm, $x \overset{\$}{\leftarrow} A$ stands for A outputs x by taking some random inputs. If $a, b \in \{0,1\}^*$ are strings of equal length then $a \oplus b$ is their bitwise XOR. If $a, b \in \{0,1\}^*$ are strings, then $a||b$ denotes their concatenation. However, we sometimes write ab for $a||b$ if there is no confusion.

If $M \in \{0,1\}^*$ is a string then $|M|$ stands for its length in bits, while $||M||_n = \mathbf{max}\{1, \lceil |M|/n \rceil\}$. For any bit string $M \in \{0,1\}^*$, we let $\mathbf{pad}(M) = M10^{n-1-|M| \bmod n}$; for any bit string $M \in \{0,1\}^{ln}$, we let $\mathbf{partition}(M) = M_1M_2 \cdots M_l$ such that $M_1 \cdots M_l = M$, $|M_i| = n$ for $1 \leq i \leq l$.

Definition 1. *A message authentication scheme* MA = (KG, TG, VF) *consists of three algorithms, as follows:*

1. *The key generation algorithm* KG *is a randomized algorithm that returns a key K; we write $K \overset{\$}{\leftarrow} \mathrm{KG}$.*

2. *The tag generation algorithm* TG *is a (possibly randomized or stateful) algorithm that takes the key K and a message M to return a tag T; we write $T \xleftarrow{\$} \mathrm{TG}_K(M)$.*
3. *The verification algorithm* VF *is a deterministic algorithm that takes the key K, a message M, and a candidate tag T for M to return a bit; we write $d \leftarrow \mathrm{VF}_K(M,T)$.*

*Associated to the scheme is a message space **Plaintexts** from which M is allowed to be drawn. We require that $\mathrm{VF}_K(M, \mathrm{TG}_K(M)) = 1$ for all $M \in$ **Plaintexts**.*

2.2 $f9$ and Its Generalized Version $f9'$

Within the security architecture of 3GPP system, there is a standardized MA-scheme called $f9$ as specified in the left side of Fig. 1. $f9$ takes the block cipher KASUMI [8] as its underlying primitive, and a 128-bit constant KM= $0x$AA$\cdots$AA as key modifier. The tag generation algorithm TG in $f9$ takes a 128-bit key K, a 32-bit counter COUNT, a 32-bit random number FRESH, a 1-bit direction identifier DIRECTION, and a message $M \in \{0,1\}^*$ as inputs, returning a 32-bit tag T. Moreover, the verification algorithm VF works by running the tag generation algorithm TG again. For more details, see [9].

```
Algorithm $f9_K(M)$
    $M \leftarrow$COUNT||FRESH||$M$||DIRECTION
    $M \leftarrow$pad$(M)$
    $M_1 M_2 \cdots M_l \leftarrow$partition$(M)$
    $Y_0 \leftarrow 0^n$, $C_0 \leftarrow 0^n$
    for $i \leftarrow 1$ to $l$ do
        $X_i \leftarrow M_i \oplus Y_{i-1}$
        $Y_i \leftarrow$KASUMI$_K(X_i)$
        $C_i \leftarrow C_{i-1} \oplus Y_i$
    end for
    Tag$\leftarrow$KASUMI$_{K \oplus \mathrm{KM}}(C_l)$
    return the leftmost 32 bits of Tag
```

```
Algorithm $f9'_K(M)$

    $M \leftarrow$pad$(M)$
    $M_1 M_2 \cdots M_l \leftarrow$partition$(M)$
    $Y_0 \leftarrow 0^n$, $C_0 \leftarrow 0^n$
    for $i \leftarrow 1$ to $l$ do
        $X_i \leftarrow M_i \oplus Y_{i-1}$
        $Y_i \leftarrow E_K(X_i)$
        $C_i \leftarrow C_{i-1} \oplus Y_i$
    end for
    Tag$\leftarrow E_{K \oplus \triangle}(C_l)$
    return the leftmost $t$ bits of Tag
```

Fig. 1. Specification of $f9$ and $f9'$, where $\triangle$ is a non-zero k-bit key modifier. For any bit string $M \in \{0,1\}^*$, **pad**$(M) = M10^{n-1-|M| \bmod n}$; for any bit string $M \in \{0,1\}^{ln}$, **partition**$(M) = M_1 M_2 \cdots M_l$ such that $|M_i| = n$ for $1 \le i \le l$.

Based on the structure of $f9$, researchers give a generalized version of it, $f9'$ [2]. As specified in the right side of Fig. 1, $f9'$ removes COUNT, FRESH, DIRECTION in $f9$, making it a deterministic one.

2.3 Basic Security Models for MA-schemes

The security of MA-schemes, in the basic security models, is evaluated by how unforgeable the MA-schemes are. To be concrete, let MA=(KG,TG,VF) be a

```
Experiment $\mathbf{Exp}^{uf-1}_{\mathrm{MA},\mathcal{A}}$
$K \xleftarrow{\$} \mathrm{KG}$;
while $\mathcal{A}$ makes a query $M$ to $\mathrm{TG}_K(\cdot)$, do
        $\mathrm{Tag} \xleftarrow{\$} \mathrm{TG}_K(M)$; return Tag to $\mathcal{A}$;
if $\mathcal{A}$ makes a query $(M, T)$ to $\mathrm{VF}_K(\cdot,\cdot)$
    s.t. $\mathrm{VF}_K(M, T)$ returns 1 and
        $M$ was never queried to $\mathrm{TG}_K(\cdot)$;
then return 1 else return 0.
```

```
Experiment $\mathbf{Exp}^{suf-1}_{\mathrm{MA},\mathcal{A}}$
$K \xleftarrow{\$} \mathrm{KG}$;
while $\mathcal{A}$ makes a query $M$ to $\mathrm{TG}_K(\cdot)$, do
        $\mathrm{Tag} \xleftarrow{\$} \mathrm{TG}_K(M)$; return Tag to $\mathcal{A}$;
if $\mathcal{A}$ makes a query $(M, T)$ to $\mathrm{VF}_K(\cdot,\cdot)$
    s.t. $\mathrm{VF}_K(M, T)$ returns 1 and
        $T$ was never returned by $\mathrm{VF}_K(\cdot,\cdot)$
        in response to query $M$;
then return 1 else return 0.
```

Fig. 2. Basic experiments defining security of MA-schemes, here the adversaries are allowed to challenge only once

MA-scheme and *type* $\in \{uf-1, suf-1\}$ and let $\mathcal{A}$ be an adversary that has access to two oracles $\mathrm{TG}_K(\cdot)$ and $\mathrm{VF}_K(\cdot,\cdot)$; then, consider the experiments in Fig. 2:

Let $\mathrm{Adv}^{type}_{\mathrm{MA},\mathcal{A}}$ be the probability that experiment $\mathbf{Exp}^{type}_{\mathrm{MA},\mathcal{A}}$ returns 1, then for any t, q, μ let

$$\mathrm{Adv}^{type}_{\mathrm{MA}}(t, q, \mu) = \max_{\mathcal{A}}\{\mathrm{Adv}^{type}_{\mathrm{MA},\mathcal{A}}\}$$

where the maximum is over all $\mathcal{A}$ running in time t, making at most q oracle queries, and such that the sum of the lengths of all oracle queries plus the length of the message M in the output forgery is at most μ bits.

We say that MA=(KG,TG,VF) is *type*-secure if the function
$\mathrm{Adv}^{type}_{\mathrm{MA}}(t, q, \mu)$ is negligible for any polynomial time adversary $\mathcal{A}$.

When the adversaries are allowed to challenge more than once, similar experiments and security definitions can be given from above, or refer to [1]. Furthermore, in [1] Bellare, etc. have analyzed the relationships among the four basic models.

However, all these four basic models treat the MA-schemes as a whole, not allowing the adversaries to obtain some information about the secret values in the MA-scheme. Now, considering the various attacks have been presented here and there [5,6,7], it seems that the four basic models are not sufficient, since the adversaries would not like to obey the rules in these models. So, why not extend them to some new ones, give the adversaries more power, and reconsider the security of MA-schemes from a new point of view?

3 Extended Security Models for MA-schemes

In this section, we extend the basic security models for MA-schemes, taking some information about the secret values in MA-schemes into account, and result in some stronger models. In these new models, adversaries not only have access to the tag generation algorithms and verification algorithms in MA-schemes, but also can obtain some information (partial or complete values, and so on) about the secret values (including internal states, etc.) in MA-schemes. At last, they are still asked to make a legal forgery, and MA-schemes are deemed to be secure if all adversaries can do this within only negligible probability.

We first introduce the general models; however, since they are not specific enough to solve problems, we then give a specific one.

3.1 The General Models

To be concrete, let MA=(KG,TG,VF) be a MA-scheme and $type^+ \in \{uf^+ - 1, suf^+ - 1\}$. Let $\mathcal{A}$ be an adversary that has access to three oracles $\mathrm{TG}_K(\cdot)$, $\mathrm{VF}_K(\cdot,\cdot)$ and an extra information oracle $O_K^+(\cdot)$, where $O_K^+(\cdot)$ takes the same input as $\mathrm{TG}_K(\cdot)$, and outputs some information about the secret values during the operations of $\mathrm{TG}_K(\cdot)$. Furthermore, $\mathcal{A}$ is not allowed to query $O_K^+(\cdot)$ independently of $\mathrm{TG}_K(\cdot)$, because in the general models $O_K^+(\cdot)$ is to simulate the leakage of secret values when $\mathrm{TG}_K(\cdot)$ is running. Then, consider the experiments in Fig. 3:

```
Experiment $\mathbf{Exp}_{\mathrm{MA},\mathcal{A}}^{uf^+-1}$
$K \xleftarrow{\$} \mathrm{KG}$;
while $\mathcal{A}$ makes a query $M$ to $\mathrm{TG}_K(\cdot)$, do
        $\mathrm{Tag} \xleftarrow{\$} \mathrm{TG}_K(M)$; return Tag to $\mathcal{A}$;
        $s \xleftarrow{\$} O_K^+(M)$; return $s$ to $\mathcal{A}$;
if $\mathcal{A}$ makes a query $(M,T)$ to $\mathrm{VF}_K(\cdot,\cdot)$
    s.t. $\mathrm{VF}_K(M,T)$ returns 1 and
        $M$ was never queried to $\mathrm{TG}_K(\cdot)$;
then return 1 else return 0.
```

```
Experiment $\mathbf{Exp}_{\mathrm{MA},\mathcal{A}}^{suf^+-1}$
$K \xleftarrow{\$} \mathrm{KG}$;
while $\mathcal{A}$ makes a query $M$ to $\mathrm{TG}_K(\cdot)$, do
        $\mathrm{Tag} \xleftarrow{\$} \mathrm{TG}_K(M)$; return Tag to $\mathcal{A}$;
        $s \xleftarrow{\$} O_K^+(M)$; return $s$ to $\mathcal{A}$;
if $\mathcal{A}$ makes a query $(M,T)$ to $\mathrm{VF}_K(\cdot,\cdot)$
    s.t. $\mathrm{VF}_K(M,T)$ returns 1 and
        $T$ was never returned by $\mathrm{VF}_K(\cdot,\cdot)$
        in response to query $M$;
then return 1 else return 0.
```

Fig. 3. Extended experiments defining security of MA-schemes, here the adversaries are allowed to challenge only once

Let $\mathrm{Adv}_{\mathrm{MA},\mathcal{A}}^{type^+}$ be the probability that experiment $\mathbf{Exp}_{\mathrm{MA},\mathcal{A}}^{type^+}$ returns 1, then for any t, q, μ let

$$\mathrm{Adv}_{\mathrm{MA}}^{type^+}(t,q,\mu) = \max_{\mathcal{A}}\{\mathrm{Adv}_{\mathrm{MA},\mathcal{A}}^{type^+}\}$$

where the maximum is over all $\mathcal{A}$ running in time t, making at most q oracle queries, and such that the sum of the lengths of all oracle queries plus the length of the message M in the output forgery is at most μ bits.

We say that MA=(KG,TG,VF) is $type^+$-secure if the function $\mathrm{Adv}_{\mathrm{MA}}^{type^+}(t,q,\mu)$ is negligible for any polynomial time adversary $\mathcal{A}$.

When the adversaries are allowed to challenge more than once, similar experiments and security definitions can be given from above.

The general models add an extra information oracle $O_K^+(\cdot)$ to the four basic ones, implying adversaries have more power now. For discussing security of specific MA-schemes, one has to make $O_K^+(\cdot)$ a specific one. We note that if $O_K^+(\cdot)$ is not properly restricted, maybe there is no MA-scheme secure in that model.

However, this is not the point. The purpose of the general models is to provide a framework of security models, under which one can study how properly can he/she do to restrict the extra information oracle $O_K^+(\cdot)$, and then consider the security of their MA-scheme in such a model, and finally get a better understanding of the security about their MA-scheme. That is, by making different restrictions on $O_K^+(\cdot)$, one knows under what conditions their MA-scheme is secure and under what conditions it is not, thus he/she knows better about their MA-scheme.

Moreover, review the attacks have been presented, such like [5,6,7], the general models can explain why these attacks are able to win. To do this, just specify the extra information oracle $O_K^+(\cdot)$ as those attacks do, then consider the security of MA-schemes in these models. Obviously, the attacked MA-schemes are no longer secure in these models.

Nevertheless, we stress that insecurity of a MA-scheme in some specific models does not necessarily imply insecurity of the MA-scheme in practice, since the cost to obtain the extra information may be expensive. That is, it is possibly too hard to realize $O_K^+(\cdot)$ of the model in practice.

Now, we give an example to show how to use the general models.

3.2 A Specific Model

Follow the framework in the general models, we now specify the extra information oracle $O_K^+(\cdot)$ in the left side of Fig. 3, resulting in a specific one that is suitable to solve some problems.

$O_K^+(\cdot)$ is defined as follows: every time after adversary $\mathcal{A}$ queries M, we let $O_K^+(\cdot)$ output one of the internal states at some particular position when the tag generation algorithm $\mathrm{TG}_K(\cdot)$ is processing M.

From an engineering point of view, since SPA and DPA have already been presented in [5,6,7], it is possible and practical for $O_K^+(\cdot)$ to output some internal states in this specific model. Furthermore, notice that the practical SPA and DPA have to repeat the same operations lots of times to obtain some secret values, thus letting $O_K^+(\cdot)$ output only one internal state each time is reasonable and enough.

From a theoretical point of view, it seems that such a model is too strong that no MA-scheme can survive, since the internal states have great influence over the security of MA-schemes [3,4,5,6,7]. However, as we will prove later, different internal states play different roles in the security of MA-schemes, and if the "particular position" is properly restricted, a nonce-based $f9$ is secure in such a model.

Nevertheless, the security results of nonce-based $f9$ in our specific model can tell us more than that. Since the nonce-based $f9$ is insecure in our specific model if $O_K^+(\cdot)$ is not restricted to the particular position, we know special measures should be taken to protect the vulnerable positions. Thus, we give some advice on the implementation of $f9$ in the real world.

The detailed discussions of $f9$-like MA-schemes in our specific model are given in the next section.

4 The Security of $f9$-like in the Specific Model

We now consider the security of $f9$-like MA-schemes in the specific model. In doing this, we will consider every position in the operations of these MA-schemes, and if it is insecure, we give an attack; otherwise, we give a security proof.

4.1 Negative Results

Let $f9^{'}_{P_1,P_2}$ be an algorithm that replaces the underlying block ciphers E_K and $E_{K\oplus\triangle}$ with P_1 and P_2, where P_1 and P_2 are two independently random permutations over $\{0,1\}^n$.

Theorem 1. *If the extra information oracle $O_K^+(\cdot)$ outputs only one of the internal states X_i for $i = 2, 3, \cdots, l$ or Y_i for $i = 1, 2, \cdots, l$ or C_i for $i = 1, 2, \cdots, l$ after each query ($l = ||M||_n$), then $f9^{'}_{P_1,P_2}$ is not secure in the specific model.*

Proof. We prove this theorem by giving some attacks on $f9^{'}_{P_1,P_2}$, where the adversary can make a forgery with probability 1.

If the adversary can obtain a Y_i after each query ($s = Y_i$ in $\mathbf{Exp}^{uf^+-1}_{\mathrm{MA},\mathcal{A}}$), then we launch a forgery attack on $f9^{'}_{P_1,P_2}$ as follows:

1. Adversary $\mathcal{A}$ selects a message $M \in \{0,1\}^*$ at random, and queries the oracle $\mathrm{TG}_K(\cdot)$ and the extra information oracle $O_K^+(\cdot)$. $\mathrm{TG}_K(\cdot)$ returns T to $\mathcal{A}$, and $O_K^+(\cdot)$ returns Y_i to $\mathcal{A}$.
2. $\mathcal{A}$ makes a forgery $(M^{'}, T)$, where $\mathbf{pad}(M)$=$M_1||M_2||\cdots||M_l$ and $\mathbf{pad}(M^{'})$= $M_1||\cdots||M_i||Y_i \oplus M_1||M_2||\cdots||M_i||Y_i \oplus M_1||M_2||\cdots||M_l$.

It is easy to check that the forgery $(M^{'}, T)$ is always legal, according the algorithm of $f9^{'}_{P_1,P_2}$; thus adversary $\mathcal{A}$ makes a forgery with probability 1.

If adversary $\mathcal{A}$ can obtain any X_i for $i = 2, 3, \cdots, l$, he/she can make a forgery similarly, since $X_i = M_i \oplus Y_{i-1}$.

Furthermore, notice that if adversary $\mathcal{A}$ can obtain C_i and C_{i+1} in two queries whose first r ($r \geq i$) blocks are the same, then $\mathcal{A}$ can obtain $Y_{i+1} = C_{i+1} \oplus C_i$, a similar attack can be launched. □

4.2 Positive Results

Now we prove that a nonce-based $f9$, N-$f9$ as defined in Fig. 4, is secure in the specific model, where nonce $N \in \{0,1\}^n$. For convenience, we do not consider the tag truncation here.

Let N-$f9_{P_1,P_2}$ be an algorithm that replaces the block ciphers E_K and $E_{K\oplus\triangle}$ with P_1 and P_2 in N-$f9$, where P_1 and P_2 are two independently random permutations over $\{0,1\}^n$. First we note that if adversary $\mathcal{A}$ can obtain one of the internal states X_i for $i = 2, 3, \cdots, l+1$ or Y_i for $i = 1, 2, \cdots, l+1$, similar attacks as given above also apply to N-$f9_{P_1,P_2}$, so N-$f9_{P_1,P_2}$ is insecure in such conditions. However, if only one of C_j ($j \geq 2$) is obtained by $\mathcal{A}$, then N-$f9_{P_1,P_2}$ is provably secure.

```
Algorithm $N$-$f9$ : $\mathrm{TG}_K(N, M)$
    $M \leftarrow$ pad$(N||M)$
    $M_1 M_2 \cdots M_{l+1} \leftarrow$ partition$(M)$
    $Y_0 \leftarrow 0^n$, $C_0 \leftarrow 0^n$
    for $i \leftarrow 1$ to $l+1$ do
        $X_i \leftarrow M_i \oplus Y_{i-1}$
        $Y_i \leftarrow E_K(X_i)$
        $C_i \leftarrow C_{i-1} \oplus Y_i$
    end for
    Tag$\leftarrow E_{K \oplus \triangle}(C_{l+1})$
    return Tag
```

```
Algorithm $N$-$f9$ : $\mathrm{VF}_K(N, M, T)$
    $M \leftarrow$ pad$(N||M)$
    $M_1 M_2 \cdots M_{l+1} \leftarrow$ partition$(M)$
    $Y_0 \leftarrow 0^n$, $C_0 \leftarrow 0^n$
    for $i \leftarrow 1$ to $l+1$ do
        $X_i \leftarrow M_i \oplus Y_{i-1}$
        $Y_i \leftarrow E_K(X_i)$
        $C_i \leftarrow C_{i-1} \oplus Y_i$
    end for
    Tag$\leftarrow E_{K \oplus \triangle}(C_{l+1})$
    if (Tag=T) return 1, else return 0
```

Fig. 4. The specification of N-$f9$

Theorem 2. *Suppose the extra information oracle $O_K^+(\cdot,\cdot)$ outputs only one of the internal states C_j for $j = 2, 3, \cdots, ||M||_n + 1$ after each query M ($s = C_j$ in $\mathbf{Exp}^{uf^+-1}_{\mathrm{MA},\mathcal{A}}$). Let $\mathcal{A}$ be a nonce-respecting ($\mathcal{A}$ never makes a query with a nonce appeared before.) adversary which asks at most q queries, having aggregate length of at most σ blocks. Then*

$$\mathrm{Adv}^{uf^+-1}_{N-f9_{P_1,P_2}} \le \frac{\sigma^2+2q^2+2\sigma q}{2^n}$$

To prove Theorem 2, let us review the definition of pseudorandom functions first; as we will show later, N-$f9$:$\mathrm{TG}_K(\cdot,\cdot)$ is in fact a nonce-based variable-length input pseudorandom function from $\{0,1\}^n \times \{0,1\}^*$ to $\{0,1\}^{2n}$ in the specific model. Furthermore, notice that it has been proved a pseudorandom function is a secure MA-scheme [10], thus N-$f9$ is a secure MA-scheme.

Let $R(\cdot,\cdot)$ be a random function on inputs (N, M) returns a random string of $2n$ bits, where $N \in \{0,1\}^n$ and $M \in \{0,1\}^*$. A nonce-respecting adversary $\mathcal{A}$ is asked to distinguish $F_K(\cdot,\cdot)$ (with a randomly chosen key K) from the random function $R(\cdot,\cdot)$. We define the advantage of $\mathcal{A}$ as

$$\begin{cases} \mathrm{Adv}^{viprf}_F(\mathcal{A}) \overset{\mathrm{def}}{=} |\Pr(K \overset{\$}{\leftarrow} \mathcal{K}_F : \mathcal{A}^{F_K(\cdot,\cdot)} = 1) - \Pr(\mathcal{A}^{R(\cdot,\cdot)} = 1)|, \\ \mathrm{Adv}^{viprf}_F(t, q, \sigma) \overset{\mathrm{def}}{=} \max_{\mathcal{A}}\{\mathrm{Adv}^{viprf}_F(\mathcal{A})\}. \end{cases}$$

where the maximum is over all adversaries who run in time at most t, and make at most q queries, having aggregate length of at most σ blocks ($\sigma = \sum_{i=1}^{q} ||M^i||_n$). We say that $F_K(\cdot,\cdot)$ is a viprf (Variable-length Input PseudoRandom Function) if $\mathrm{Adv}^{viprf}_F(t, q, \sigma)$ is negligible.

Without loss of generality, adversaries are assumed to never ask a query outside the domain of the oracle.

The pseudorandomness of N-$f9$ is proved by the counting method, which has been used for many times, e.g. [11,2]. To make the whole proof for N-$f9$ more clear and structural, we first give two lemmas.

Lemma 1. *Define set* $M \overset{\text{def}}{=} \{(M^1, M^2, \cdots, M^q) | M^i \in \{0,1\}^*\}$ *and set* $N \overset{\text{def}}{=} \{(N^1, N^2, \cdots, N^q) | N^i \in \{0,1\}^n,\ N^i \neq N^j, 1 \leq i < j \leq q\}$*, if there exists a set* $T = \{(T^1, T^2, \cdots, T^q) | T^i \in \{0,1\}^{2n}, 1 \leq i \leq q\}$ *and a function family* $F : \mathcal{K}_F \times \{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^{2n}$ *such that,*

1) $\#T \geq 2^{2qn}(1 - \epsilon_1)$*,*

2) $\forall (N^1, N^2, \cdots, N^q) \in N, \forall (M^1, M^2, \cdots, M^q) \in M, \forall (T^1, T^2, \cdots, T^q) \in T,$

$$\Pr(K \overset{\$}{\leftarrow} \mathcal{K}_F : T^i = F_K(N^i, M^i)) \geq \tfrac{1-\epsilon_2}{2^{2qn}}$$

then for any nonce-respecting adversary $\mathcal{A}$ *(computationally unbounded) asking* q *queries,*

$$|\Pr(K \overset{\$}{\leftarrow} \mathcal{K}_F : \mathcal{A}^{F_K(\cdot,\cdot)} = 1) - \Pr(\mathcal{A}^{R(\cdot,\cdot)} = 1)| \leq \epsilon_1 + \epsilon_2.$$

The idea to prove Lemma 1 is from [11,2], and the detailed proof is in the full version of this paper [12].

Lemma 2 is a mathematical property of N-$f9$.

Lemma 2. *Define sets* M, N *as in Lemma 1 and* $T \overset{\text{def}}{=} \{(C^i, S^i) | C^i \in \{0,1\}^n, C^i \neq C^j, S^i \in \{0,1\}^n, 1 \leq i < j \leq q\}$*. Let* $Rand(n,n)$ *be the set of all functions from* $\{0,1\}^n$ *to* $\{0,1\}^n$*, and* $g_1, g_2 \in Rand(n,n)$ *be the underlying functions in* N-$f9_{g_1,g_2}$*. Let* $\mathrm{TG}_{g_1,g_2}(\cdot,\cdot)$ *be the tag generation algorithm in* N-$f9_{g_1,g_2}$*. Then the number of functions* (g_1, g_2) *satisfying*
$\forall (N^1, N^2, \cdots, N^q) \in N, \forall (M^1, M^2, \cdots, M^q) \in M, \forall (T^1, T^2, \cdots, T^q) \in T,$

1) $S^i = \mathrm{TG}_{g_1,g_2}(N^i, M^i), 1 \leq i \leq q$*, and*

2) $C^i_{j_i}$ *is an internal state of* $\mathrm{TG}_{g_1,g_2}(N^i, M^i)$ *at position* j_i *,for* $1 \leq i \leq q$ *and* $2 \leq j_i \leq ||M^i||_n + 1$
is at least $N_{g_1,g_2} \geq (1 - \frac{\sigma^2+q^2+2\sigma q-\sigma-q}{2^{n+1}}) \times (2^n)^{2^{n+1}-2q}$*, where* $\sigma = \sum_{i=1}^{q} ||M^i||_n$*.*

For the proof of Lemma 2, refer to Appendix A.

Based on the two lemmas given above, let us prove Theorem 2.

Proof. According to Lemma 2, we get

$$\begin{aligned}
&\Pr(g_1, g_2 \overset{\$}{\leftarrow} \mathrm{Rand}(n,n) : \text{conditions 1 and 2 hold}) \\
&\geq \frac{(1 - \frac{\sigma^2+q^2+2\sigma q-\sigma-q}{2^{n+1}}) \times (2^n)^{2^{n+1}-2q}}{(2^n)^{2^n} \times (2^n)^{2^n}} \\
&= \frac{1 - \frac{\sigma^2+q^2+2\sigma q-\sigma-q}{2^{n+1}}}{2^{2qn}},
\end{aligned}$$

which indicates $\epsilon_2 = \frac{\sigma^2+q^2+2\sigma q-\sigma-q}{2^{n+1}}$ in Lemma 1.

Furthermore, notice that $\#T = 2^{2qn}(1 - \binom{q}{2} 2^{-n})$, which means $\epsilon_1 = \frac{q^2-q}{2^{n+1}}$. Applying Lemma 1, it follows that

$$|\Pr(g_1, g_2 \overset{\$}{\leftarrow} \mathrm{Rand}(n,n) : \mathcal{A}^{\mathrm{TG}_{g_1,g_2}(\cdot,\cdot)} = 1) - \Pr(\mathcal{A}^{R(\cdot,\cdot)} = 1)|$$

$$\begin{aligned}&\leq \epsilon_1 + \epsilon_2\\ &= \frac{\sigma^2 + 2q^2 + 2\sigma q - \sigma - 2q}{2^{n+1}}.\end{aligned} \tag{1}$$

Moreover, we can obtain

$$\begin{aligned}&|\Pr(\mathcal{A}^{\mathrm{TG}_{g_1,g_2}(\cdot,\cdot)} = 1) - \Pr(\mathcal{A}^{\mathrm{TG}_{P_1,P_2}(\cdot,\cdot)} = 1)|\\ \leq\ &|\Pr(g_1, g_2 \xleftarrow{\$} \mathrm{Rand}(n,n) : \mathcal{B}_{\mathcal{A}}{}^{(g_1(\cdot),g_1(\cdot))} = 1)\\ &- \Pr(P_1, P_2 \xleftarrow{\$} \mathrm{Perm}(n) : \mathcal{B}_{\mathcal{A}}{}^{(P_1(\cdot),P_2(\cdot))} = 1)| &(2)\\ \leq\ &\binom{\sigma+q}{2} 2^{-n} + \binom{q}{2} 2^{-n} &(3)\\ =\ &\frac{\sigma^2 + 2q^2 + 2\sigma q - \sigma - 2q}{2^{n+1}}, &(4)\end{aligned}$$

Inequality (2) holds because for every adversary $\mathcal{A}$ distinguishing $\mathrm{TG}_{g_1,g_2}(\cdot,\cdot)$ and $\mathrm{TG}_{P_1,P_2}(\cdot,\cdot)$, we can construct another adversary $\mathcal{B}_{\mathcal{A}}$ to distinguish (g_1, g_2) and (P_1, P_2), by running $\mathcal{A}$. According to the "PRP and PRF Switching Lemma" [10], inequality (3) holds, where the adversary queries $g_1(\cdot)$ (or $P_1(\cdot)$) $\sigma+q$ times, and queries $g_2(\cdot)$ (or $P_2(\cdot)$) q times.

Then, by inequalities (1) and (4), we get

$$\begin{aligned}&\mathrm{Adv}^{viprf}_{N-f9_{P_1,P_2}}\\ &= |\Pr(P_1, P_2 \xleftarrow{\$} \mathrm{Perm}(n) : \mathcal{A}^{\mathrm{TG}_{P_1,P_2}(\cdot,\cdot)} = 1) - \Pr(\mathcal{A}^{R(\cdot,\cdot)} = 1)|\\ &\leq \frac{\sigma^2 + 2q^2 + 2\sigma q - \sigma - 2q}{2^n}\end{aligned}$$

Now we have proved that TG_{P_1,P_2} is a viprf from $\{0,1\}^n \times \{0,1\}^*$ to $\{0,1\}^{2n}$. Obviously, if we abandon the random bits $C^i_{j_i}$, TG_{P_1,P_2} is a viprf from $\{0,1\}^n \times \{0,1\}^*$ to $\{0,1\}^n$.

Finally, we can conclude that N-$f9_{P_1,P_2}$ is a secure MA-scheme, by the fact that a pseudorandom function is a secure MA-scheme [10],

$$\mathrm{Adv}^{uf^+-1}_{N-f9_{P_1,P_2}} \leq \mathrm{Adv}^{viprf}_{N-f9_{P_1,P_2}} + \frac{1}{2^n} < \frac{\sigma^2 + 2q^2 + 2\sigma q}{2^n}.$$

□

Corollary 1. *Suppose the underlying block cipher E in N-$f9$ is a secure PRP-RKA (this implies that, for any key K, E_K and $E_{K\oplus\triangle}$ are indistinguishable from two independently random permutations P_1 and P_2), and suppose the extra information oracle $O^+_K(\cdot,\cdot)$ outputs only one of the internal states C_j for $j = 2, 3, \cdots, ||M||_n + 1$ after each query M. Let $\mathcal{A}$ be a nonce-respecting adversary which asks at most q queries, having aggregate length of at most σ blocks. Then*

$$\mathrm{Adv}^{uf^+-1}_{N-f9} \leq \mathrm{Adv}^{\mathrm{prp-rka}}_{E} + \tfrac{\sigma^2+2q^2+2\sigma q}{2^n}$$

The proof for Corollary 1 is easy, or refer to [2].

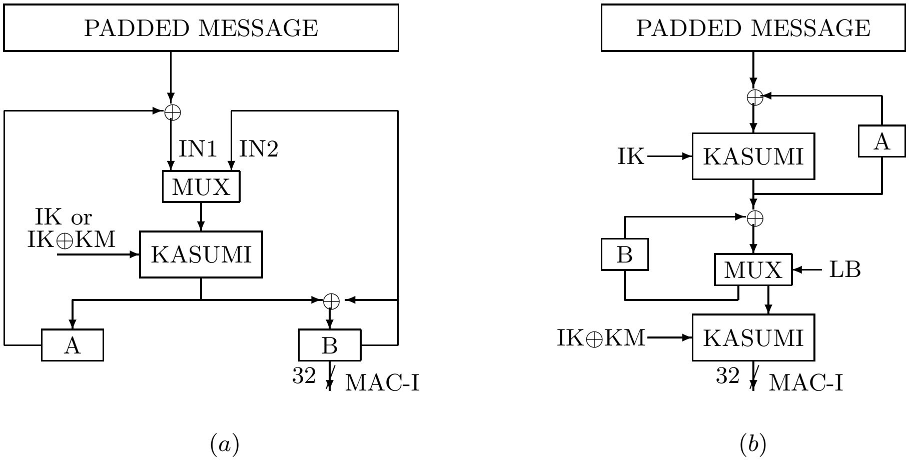


Fig. 5. Two implementations of $f9$ from [14,15], where "A" and "B" are two registers, "MUX" is a selector and "LB" denotes Last Block

4.3 Application of the Results in the Specific Model

Notice that $f9$ has already been used in practice, and the first block it processes is COUNT||FRESH, which can be seen as a nonce. Furthermore, its underlying block cipher, KASUMI, can be seen as a secure PRP-RKA (a PseudoRandom Permutation that is secure against a certain kind of Related-Key Attacks [2]) although it has been theoretically broken out [13], because adversaries are not allowed to take related-key attacks on KASUMI in practical environments of $f9$. Thus, N-$f9$ is indeed close to the real world. Nevertheless, we suggest KASUMI be replaced by more stronger block ciphers, if feasible.

In Fig. 5, we illustrate two implementations of $f9$ ((a) Section 3.3 in [14] and (b) Section 4.3 in [15]), which major on higher efficiency in practice. Then, the results in the specific model can fulfill their works, in the aspect of security:

1. From an engineering point of view, register A in Fig. 5 should been specially protected, in case that adversaries try to acquire some (partial or complete) secret values (X_i and Y_i) to launch an attack.
2. From an attacker's point of view, attacking register A to reveal some information about X_i and Y_i will give them great chances to make a successful attack; while attacking register B they have little probability to win (except that they can obtain some information about C_1).

It is interesting to consider the security of other MA-schemes in this specific model; however, as far as we have done, no other MA-scheme secure in this model has been found. We stress that this result does not imply the other MA-schemes are insecure in practice, since the practical environments may not allow adversaries to launch an attack in the specific model. Nevertheless, the result tells us that, from the point of view in the specific model, $f9$ is stronger than the others in practice.

5 Conclusions

To sum up, we extend the basic security models for message authentication in this paper, based on the fact that many attacks on MA-schemes have been presented by some ways that are outside of the basic models consider about, and finally get some extended ones suitable to fix the problem. The general models introduced here offer the users great freedom to consider the security of their MA-schemes in different aspects, thus help them to get a better understanding of their MA-schemes. While the security discussions of $f9$-like MA-schemes in the specific model give us more knowledge about how to use $f9$ safely.

Acknowledgments. The authors would like to thank the anonymous referees for their valuable comments. Furthermore, this work is supported by the National High-Tech Research and Development 863 Plan of China (No. 2007AA01Z470), the National Natural Science Foundation of China (No. 60873259 and No. 90604036), and the National Grand Fundamental Research 973 Program of China (No. 2004CB318004).

References

1. Bellare, M., Goldreich, O., Mityagin, A.: The Power of Verification Queries in Message Authentication and Authenticated Encryption. Cryptology ePrint Archive: Report 2004/309
2. Iwata, T., Kohno, T.: New Security Proofs for the 3GPP Confidentiality and Integrity Algorithms. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 427–445. Springer, Heidelberg (2004)
3. Preneel, B., van Oorschot, P.: On the Security of Iterated Message Authentication Codes. IEEE Transactions on Information Theory 45(1), 188–199 (1999)
4. Knudsen, L.R., Mitchell, C.J.: Analysis of 3GPP-MAC and Two-Key 3GPP-MAC. Discrete Applied Mathematics 128(1), 181–191 (2003)
5. Okeya, K., Iwata, T.: Side Channel Attacks on Message Authentication Codes. In: Molva, R., Tsudik, G., Westhoff, D. (eds.) ESAS 2005. LNCS, vol. 3813, pp. 205–217. Springer, Heidelberg (2005)
6. Okeya, K.: Side Channel Attacks against HMACs based on Block-Cipher based Hash Functions. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 432–443. Springer, Heidelberg (2006)
7. Gauravaram, P., Okeya, K.: An Update on the Side Channel Cryptanalysis of MACs based on Cryptographic Hash Functions. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 393–403. Springer, Heidelberg (2007)
8. ETSI TS 35.202 V7.0.0: Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification, `http://www.3gpp.org/ftp/Specs/html-info/35201.htm`
9. ETSI TS 35.201 V7.0.0: Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 1: $f8$ and $f9$ Specification, `http://www.3gpp.org/ftp/Specs/html-info/35201.htm`
10. Bellare, M., Kilian, J., Rogaway, P.: The Security of Cipher Block Chaining. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 341–358. Springer, Heidelberg (1994)

11. Patarin, J.: A Proof of Security in O (2ˆ n) for the Xor of Two Random Permutations. Cryptology ePrint Archive: Report 2008/010
12. Zhang, L., Wu, W., Wang, P.: Extended Models for Message Authentication (full version), available from the authors
13. Biham, E., Dunkelman, O., Keller, N.: A Related-Key Rectangle Attack on the Full KASUMI. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 443–461. Springer, Heidelberg (2005)
14. Kitsos, P., Sklavos, N., Koufopavlou, O.: UMTS Security: System Architecture and Hardware Implementation. Wireless Communications and Mobile Computing 7(4), 483–494 (2007)
15. Marinis, K., Moshopoulos, N.K., Karoubalis, F., Pekmestzi, K.Z.: On the Hardware Implementation of the 3GPP Confidentiality and Integrity Algorithms. In: Davida, G.I., Frankel, Y. (eds.) ISC 2001. LNCS, vol. 2200, pp. 248–265. Springer, Heidelberg (2001)

A Proof for Lemma 2

Proof. To prove Lemma 2, we give a method to find a class of functions (g_1, g_2) such that 1) $S^i = \mathrm{TG}_{g_1,g_2}(N^i, M^i), 1 \le i \le q$, and 2) $C^i_{j_i}$ is an internal state of $\mathrm{TG}_{g_1,g_2}(N^i, M^i)$ at position j_i for $1 \le i \le q$ and $2 \le j_i \le ||M^i||_n + 1$.

To do this, we start from two undefined functions (g_1, g_2), and run $\mathrm{TG}_{g_1,g_2}(\cdot,\cdot)$. By making the inputs to (g_1, g_2) pairwise distinct all the time, we can completely control the outputs of (g_1, g_2), which are used to control the next inputs to (g_1, g_2). Finally, we can establish a bridge between every (N^i, M^i) and $(C^i_{j_i}, S^i)$ for $i = 1, 2, \cdots, q$. Then, we count the number of functions (g_1, g_2) by the input-output limitations we have made.

Concretely, consider the program in Fig. 6.

At the beginning, we make functions g_1 and g_2 undefined. Domain_2 denotes the domain of g_2, so it is empty (line 00). Then, we select q nonces, which are going to be used in $\mathrm{TG}_{g_1,g_2}(\cdot,\cdot)$. Since all the nonces are the inputs to g_1, we let $\mathrm{Domain}_1 \leftarrow \{N^1, N^2, \cdots, N^q\}$ (line 00).

Then, for every (N^i, M^i) $(i = 1, 2, \cdots, q)$, we establish a bridge between (N^i, M^i) and $(C^i_{j_i}, S^i)$ by three steps. The program in Fig. 6 follows $\mathrm{TG}_{g_1,g_2}(\cdot,\cdot)$ naturally, and we only explain some key points.

Step 1. We deal with the first $j_i - 2$ blocks in $\overline{M^i}$. Since $g_1(X^i_j)$ for $j = 1, 2, \cdots, j_i - 2$ has not been defined, we can randomly select Y^i_j (lines 07 and 08) such that the next input to g_1, $X^i_{j+1} = Y^i_j \oplus \overline{M^i_{j+1}} \notin \mathrm{Domain}_1$. Notice that Y^i_j has $2^n - \#\mathrm{Domain}_1$ choices.

Step 2. To satisfy condition 2) $C^i_{j_i}$ is an internal state of $\mathrm{TG}_{g_1,g_2}(N^i, M^i)$ at position j_i for $1 \le i \le q$ and $2 \le j_i \le ||M^i||_n + 1$, we have to randomly select $Y^i_{j_i-1}$ (lines 12 and 14) satisfying not only the next input to g_1, $X^i_{j_i} = Y^i_{j_i-1} \oplus \overline{M^i_{j_i}} \notin \mathrm{Domain}_1$ but also the further input to g_1, $X^i_{j_i+1} = Y^i_{j_i-1} \oplus C^i_{j_i-2} \oplus C^i_{j_i} \oplus \overline{M^i_{j_i+1}} \notin \mathrm{Domain}_1$. Notice that $Y^i_{j_i-1}$ has $2^n - (\#\mathrm{Domain}_1 + \#\mathrm{Domain}_1 + 1)$ choices and $Y^i_{j_i}$ has only one choice (line 17).

```
00. $g_1 \leftarrow$undefined, $g_2 \leftarrow$undefined, Domain$_2 \leftarrow \phi$, Domain$_1 \leftarrow \{N^1, N^2, \cdots, N^q\}$.
01. for $i \leftarrow 1$ to $q$ do
02.     $\overline{M^i} \leftarrow$pad$(N^i || M^i)$
03.     $\overline{M_1^i M_2^i \cdots M_{l_i+1}^i} \leftarrow$partition$(\overline{M^i})$
04.     $Y_0^i \leftarrow 0^n$, $C_0^i \leftarrow 0^n$
05.     for $j \leftarrow 1$ to $j_i - 2$ do
06.         $X_j^i \leftarrow Y_{j-1}^i \oplus \overline{M_j^i}$, Domain$_1 \leftarrow \{X_j^i\} \cup$Domain$_1$
07.         $Y_j^i \xleftarrow{\$} \{0,1\}^n$
08.         while $(Y_j^i \oplus \overline{M_{j+1}^i} \in$Domain$_1)$ do $Y_j^i \xleftarrow{\$} \{0,1\}^n$ end while
09.         $C_j^i \leftarrow C_{j-1}^i \oplus Y_j^i$, $g_1(X_j^i) \leftarrow Y_j^i$
10.     end for
11.     $X_{j_i-1}^i \leftarrow Y_{j_i-2}^i \oplus \overline{M_{j_i-1}^i}$, Domain$_1 \leftarrow \{X_{j_i-1}^i\} \cup$Domain$_1$
12.     $Y_{j_i-1}^i \xleftarrow{\$} \{0,1\}^n$
13.     while $(Y_{j_i-1}^i \oplus \overline{M_{j_i}^i} \in$Domain$_1$ or $Y_{j_i-1}^i \oplus C_{j_i-2}^i \oplus C_{j_i}^i \oplus \overline{M_{j_i+1}^i} \in$Domain$_1)$
14.         do $Y_{j_i-1}^i \xleftarrow{\$} \{0,1\}^n$ end while
15.     $C_{j_i-1}^i \leftarrow C_{j_i-2}^i \oplus Y_{j_i-1}^i$, $g_1(X_{j_i-1}^i) \leftarrow Y_{j_i-1}^i$
16.     $X_{j_i}^i \leftarrow Y_{j_i-1}^i \oplus \overline{M_{j_i}^i}$, Domain$_1 \leftarrow \{X_{j_i}^i\} \cup$Domain$_1$
17.     $Y_{j_i}^i \leftarrow C_{j_i-1}^i \oplus C_{j_i}^i$, $g_1(X_{j_i}^i) \leftarrow Y_{j_i}^i$
18.     for $j \leftarrow j_i + 1$ to $l_i + 1$ do
19.         $X_j^i \leftarrow Y_{j-1}^i \oplus \overline{M_j^i}$, Domain$_1 \leftarrow \{X_j^i\} \cup$Domain$_1$
20.         $Y_j^i \xleftarrow{\$} \{0,1\}^n$
21.         while $(j < l_i + 1$ and $Y_j^i \oplus \overline{M_{j+1}^i} \in$Domain$_1)$ do $Y_j^i \xleftarrow{\$} \{0,1\}^n$ end while
22.         while $(j = l_i + 1$ and $Y_{l_i+1}^i \oplus C_{l_i}^i \in$Domain$_2)$
23.             do $Y_{l_i+1}^i \xleftarrow{\$} \{0,1\}^n$ end while
24.         $C_j^i \leftarrow C_{j-1}^i \oplus Y_j^i$, $g_1(X_j^i) \leftarrow Y_j^i$
25.     end for
26.     Domain$_2 \leftarrow \{C_{l_i+1}^i\} \cup$Domain$_2$, $g_2(C_{l_i+1}^i) \leftarrow S^i$
27. end for
```

Fig. 6. A program searching for (g_1, g_2) satisfying Lemma 2

Step 3. Then we deal with the last $l_i + 1 - j_i$ blocks in $\overline{M^i}$. First, we randomly select Y_j^i for $j = j_i + 1, j_i + 2, \cdots, l_i$ (lines 20 and 21) such that the next input to g_1, $X_{j+1}^i = Y_j^i \oplus \overline{M_{j+1}^i} \notin$Domain$_1$, and Y_j^i has $2^n - \#$Domain$_1$ choices. Then, we randomly select $Y_{l_i+1}^i$ (lines 20 and 23) such that the next input to g_2, $C_{l_i+1}^i = Y_{l_i+1}^i \oplus C_{l_i}^i \notin$Domain$_2$. Notice that $Y_{l_i+1}^i$ has $2^n - \#$Domain$_2 = 2^n - (i-1)$ choices and $C_{l_i+1}^i$ has only one choice (line 26).

Finally, we have established a bridge between every (N^i, M^i) and $(C_{j_i}^i, S^i)$ for $i = 1, 2, \cdots, q$, and left function g_1 $2^n - (\sigma + q)$ elements in its domain undefined and function g_2 $2^n - q$ elements in its domain undefined. Since these elements have nothing to do with the two requirements in Lemma 2, we randomly select a value from $\{0,1\}^n$ for each of them. In other words, they have 2^n choices each.

Now, we count the number of (g_1, g_2). Let $D_i = \#\mathrm{Domain}_1$ just before $\mathrm{TG}_{g_1,g_2}(N^i, M^i)$ is going to work. Then it is easy to check that $D_1 = q$, $D_2 = D_1 + l_1$, $D_3 = D_2 + l_2$, $\cdots$, $D_q = D_{q-1} + l_{q-1}$. Let $\sigma = \sum_{i=1}^{q} l_i$, so $D_q = q + \sigma - l_q$.

Let

$$\begin{aligned} N_{g_1}^i &= (2^n - D_i) \times (2^n - (D_i + 1)) \times \cdots \times (2^n - (D_i + j_i - 2 + 1)) \\ &\quad \times (2^n - (D_i + j_i - 2) - (D_i + j_i - 1)) \times 1 \times (2^n - (D_i + j_i)) \\ &\quad \times (2^n - (D_i + j_i + 1)) \times \cdots \times (2^n - (D_i + l_i - 1)) \times (2^n - (i - 1)), \end{aligned}$$

then the number of g_1 satisfying Lemma 2 is

$$\begin{aligned} N_{g_1} &= \prod_{i=1}^{q} N_{g_1}^i \times (2^n)^{2^n - (\sigma + q)} \\ &\geq ((2^n)^\sigma - (1 + 2 + \cdots + (q + \sigma - 1)) \times (2^n)^{\sigma - 1}) \times (2^n)^{2^n - (\sigma + q)} \\ &= (1 - \frac{\sigma^2 + q^2 + 2\sigma q - \sigma - q}{2^{n+1}}) \times (2^n)^{2^n - q}. \end{aligned}$$

On the other hand, $\overline{M^i}$ $(i = 1, 2, \cdots, q)$ adds one element to Domain_2 each, so the number of g_2 satisfying Lemma 2 is $N_{g_2} = 1^q \times (2^n)^{2^n - q}$.

Finally, the number of (g_1, g_2) satisfying Lemma 2 is

$$N_{g_1,g_2} = N_{g_1} \times N_{g_2} \geq (1 - \frac{\sigma^2 + q^2 + 2\sigma q - \sigma - q}{2^{n+1}}) \times (2^n)^{2^{n+1} - 2q}.$$

□

A Preimage Attack for 52-Step HAS-160

Yu Sasaki and Kazumaro Aoki

NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midoricho, Musashino-shi, Tokyo 180-8585 Japan
sasaki.yu@lab.ntt.co.jp

Abstract. In this paper, we propose preimage attacks on the hash function HAS-160, which is standardized in Korea. We propose two approaches to generate a preimage of step-reduced HAS-160 faster than a brute force attack, which costs 2^{160}. The first approach is a simple application of previously known techniques, which are so-called splice-and-cut ant partial-matching techniques. This approach generates a pseudo-preimage of 48 steps of HAS-160 with a complexity of 2^{128} and a preimage with a complexity of 2^{145}. In the second approach, we consider a variant of so-called local-collision technique. This approach generates a pseudo-preimage of 52 steps of HAS-160 with a complexity of 2^{144} and a preimage with a complexity of 2^{153}. To the best of our knowledge, this is the first paper that analyzes the preimage resistance of HAS-160.

Keywords: HAS-160, splice-and-cut, hash function, local collision, one-way, preimage.

1 Introduction

Hash functions are important cryptographic primitives that generates a relatively short bit-string from an arbitrary-length input message. They are used for various purposes in the modern society, for example, digital fingerprinting or digital signatures. Let H be a hash function, and x and y be an input and output of H. Hash functions are required to satisfy following three properties.

- **Preimage resistance:** For given y, it must be hard to find x $s.t.$ $H(x) = y$.
- **2nd-preimage resistance:** For given x, it must be hard to find x' $s.t.$ $H(x') = y, x \neq x'$.
- **Collision resistance:** It must be hard to find a pair of (x, x') $s.t.$ $H(x) = H(x'), x \neq x'$.

Let the length of y be n-bit. In preimage resistance, it is obvious that if hash values of 2^n different messages are computed, one of them will be matched with given y with high probability. Therefore, a method computing a preimage faster than 2^n is a threat for hash functions. Such a method is called preimage attack.

HAS-160 is a hash function developed in Korea, and was standardized by Korean government in 2000 [14]. The design of HAS-160 is similar to MD5 [11]

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 302–317, 2009.

Table 1. Comparison of attacks on HAS-160

Attack type	Reference	Step number	Complexity	
			Pseudo-preimage	Preimage
Collision Attack	[16]	45	2^{12}	
	[4]	53	2^{55}	
	[7]	53	2^{35}	
	[7]	59	2^{55}	
Preimage Attack	Ours (Approach 1)	48	2^{128}	2^{145}
	Ours (Approach 2)	52	2^{144}	2^{153}

and SHA-1 [15], in particular, the step function of HAS-160 is very similar to SHA-1. However, the message expansion is a mixture of designs of MD5 and SHA-1, which is faster than that of SHA-1. Since several attacks on MD5 and SHA-1 have already been known, the security analysis of HAS-160 is interesting to know the security contribution of its design.

So far, only a few papers analyze the security of HAS-160. The first cryptanalysis on HAS-160 was presented by Yun et al. [16] at ICISC 2005. They found that a collision for HAS-160 reduced to 45 steps could be generated in a very small complexity. This was improved by Cho et al. [4] at ICISC 2006, which reported that a collision attack could be theoretically applied until 53 steps. This was further improved by Mendel and Rijmen [7] at ICISC 2007, where a real collision until 53 steps was generated and a differential path yielding a 59-step collision was shown. As far as we know, no paper analyzed the preimage resistance of HAS-160 so far. On the other hand, preimage resistance of other MD4-based hash functions have been analyzed recently, for example, preimage attacks on full-round MD4 [6,1], step-reduced MD5 [5,12,2,1], full-round HAVAL-3 [2,13], full-round HAVAL-4 and step-reduced HAVAL-5 [13], and step-reduced SHA-0 and step-reduced SHA-1 [3]. There is another hash function named HAS-V [10], which has the similar structure to HAS-160. Preimage attack on HAS-V has also been discovered by Mendel and Rijmen at ICISC 2007 [8]. Therefore, the preimage resistance of HAS-160 needs to be evaluated by recent attack techniques.

Our results

In this paper, we propose two approaches to generate a preimage of step-reduced HAS-160. The first approach is a simple application of previously known techniques, which are so-called splice-and-cut ant partial-matching techniques. To improve the number of steps that can be attacked, we develop another approach which is similar to the local-collision technique. 52-step of HAS-160 are attacked with the second approach. The complexity of our attacks is summarized in Table 1. For comparison, we also list the previous collision attacks in Table 1. To the best of our knowledge, this is the first paper that analyzes the preimage resistance of HAS-160.

Organization of this paper is as follows. In Section 2, we describe the specification of HAS-160. In Section 3, we summarize the techniques in previous preimage attacks. In Section 4, we propose the preimage attack on 48-step HAS-160 by approach 1. In Section 5, we consider a variant of the local-collision technique to attack 52-step HAS-160. Finally, we conclude this paper in Section 6.

2 Description of HAS-160

HAS-160 [14] is a hash function that takes an arbitrary length message as input and outputs a 160-bit hash value. The design of HAS-160 is similar to those of SHA-1 [15] and MD5 [11].

HAS-160 has the Merkle-Damgård structure, which uses 160-bit (5-word) chaining variables and a 512-bit (16-word) message block to compute a compression function.

First, an input message M is processed to be a multiple of 512 bits by the padding procedure. A single bit 1 is appended followed by 0s until the length becomes 448 modulo 512. Finally, the binary representation of the length of M is appended at the last 64 bits.

Padded message is separated into 512-bit messages $(M_0, M_1, \ldots, M_{n-1})$. Let $CF : \{0,1\}^{160} \times \{0,1\}^{512} \to \{0,1\}^{160}$ be the compression function of HAS-160. A hash value is computed as follows.

1. $H_0 \leftarrow IV$,
2. $H_{i+1} \leftarrow CF(H_i, M_i)$ for $i = 0, 1, \ldots, n-1$,

where H_i is a 160-bit value and IV is the initial value defined in the specification. Finally, H_n is output as a hash value of M.

Table 2. Computation of m_{16} to m_{19} in each round

	Round 1	Round 2	Round 3	Round 4
m_{16}	$m[0,1,2,3]$	$m[3,6,9,12]$	$m[12,5,14,7]$	$m[7,2,13,8]$
m_{17}	$m[4,5,6,7]$	$m[15,2,5,8]$	$m[0,9,2,11]$	$m[3,14,9,4]$
m_{18}	$m[8,9,10,11]$	$m[11,14,1,4]$	$m[4,13,6,15]$	$m[15,10,5,0]$
m_{19}	$m[12,13,14,15]$	$m[7,10,13,0]$	$m[8,1,10,3]$	$m[11,6,1,12]$

$m[i,j,k,l]$ denotes $m_i \oplus m_j \oplus m_k \oplus m_l$.

Table 3. Message order in each step

Round 1: $X_0, X_1, \ldots, X_{19}$	18	0	1	2	3	19	4	5	6	7	16	8	9	10	11	17	12	13	14	15
Round 2: $X_{20}, X_{21}, \ldots, X_{39}$	18	3	6	9	12	19	15	2	5	8	16	11	14	1	4	17	7	10	13	0
Round 3: $X_{40}, X_{41}, \ldots, X_{59}$	18	12	5	14	7	19	0	9	2	11	16	4	13	6	15	17	8	1	10	3
Round 4: $X_{60}, X_{61}, \ldots, X_{79}$	18	7	2	13	8	19	3	14	9	4	16	15	10	5	0	17	11	6	1	12

Step	Message index															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0									∗	∗	∗	∗				
1	∗															
2		∗														
3			∗													
4				∗												
5													∗	∗	∗	∗
6					∗											
7						∗										
8							∗									
9								∗								
10	∗	∗	∗	∗												
11									∗							
12										∗						
13											∗					
14												∗				
15					∗	∗	∗	∗								
16													∗			
17														∗		
18															∗	
19																∗
20		∗			∗							∗			∗	
21				∗												
22							∗									
23										∗						
24													∗			
25	∗							∗			∗			∗		
26																∗
27			∗													
28						∗										
29									∗							
30				∗			∗			∗			∗			
31												∗				
32															∗	
33		∗														
34					∗											
35			∗			∗			∗							∗
36								∗								
37											∗					
38														∗		
39	∗															
40					∗		∗							∗		∗
41													∗			
42						∗										
43															∗	
44								∗								
45		∗		∗					∗		∗					
46	∗															
47										∗						
48			∗													
49												∗				
50						∗		∗					∗		∗	
51					∗											
52														∗		
53							∗									
54																∗
55	∗		∗							∗		∗				
56									∗							
57		∗														
58											∗					
59				∗												
60	∗					∗					∗					∗
61								∗								
62			∗													
63														∗		
64									∗							
65		∗					∗					∗	∗			
66				∗												
67															∗	
68										∗						
69					∗											
70			∗					∗	∗					∗		
71																∗
72											∗					
73						∗										
74	∗															
75				∗	∗					∗					∗	
76												∗				
77							∗									
78		∗														
79													∗			

$*$ denotes message words used in each step. For m_{16} to m_{19}, we write original four messages producing them in the table.

Fig. 1. A figure describing message expansion of HAS-160

Compression Function

HAS-160 iteratively computes a step function 80 times to compute a hash value. Steps 0-19, 20-39, 40-59, and 60-79 are called the first, second, third, and fourth rounds, respectively. Let (H_i, M_i) be the input of the compression function.

Message expansion. First, M_i is divided into sixteen 32-bit message-words $m_0, \ldots, m_{15}$. The message expansion of HAS-160 is a permutation of 20 message words in each round, which consists of $m_0, \ldots, m_{15}$ and four additional messages $m_{16}, \ldots, m_{19}$ computed from $m_0, \ldots, m_{15}$. The computation of $m_{16}, \ldots, m_{19}$ is shown in Table 2.

Let $X_0, X_1, \ldots, X_{79}$ be the message used in each step. The message m_j assigned to each X_j is shown in Table 3. The message expansion is also schematically written in Figure 1.

Step update function. The output of the compression function H_{i+1} is computed as follows.

1. $p_0 \leftarrow H_i$.
2. $p_{j+1} \leftarrow R_j(p_j, X_j)$ for $j = 0, 1, \ldots, 79$,
3. Output $H_{i+1}(= p_{80} + H_i)$, where "+" denotes 32-bit word-wise addition. In this paper, we similarly use "−" to denote 32-bit word-wise subtraction.

R_j is the step function for Step j. Let a_j, b_j, c_j, d_j, e_j be 32-bit values that satisfy $p_j = (a_j, b_j, c_j, d_j, e_j)$. R_j for HAS-160 is defined as follows:

$$\begin{cases} a_{j+1} = (a_j \qquad s1_j) + f_j(b_j, c_j, d_j) + e_j + X_j + k_j \\ b_{j+1} = a_j \\ c_{j+1} = b_j \qquad s2_j \\ d_{j+1} = c_j \\ e_{j+1} = d_j \end{cases}$$

where f_j, k_j, and $\quad s2_j$ are bitwise Boolean function, constant, and $s2_j$-bit left rotation defined in Table 4, and $\quad s1_j$ is $s1_j$-bit left rotation defined in the specification.

Table 4. Function f, constant k, and rotation $s2$ of HAS-160

Round	Function $f_j(X, Y, Z)$	Constant k_j	Rotation $s2_j$
Round 1	$(x \wedge y) \vee (\neg x \wedge z)$	`0x00000000`	10
Round 2	$x \oplus y \oplus z$	`0x5a827999`	17
Round 3	$y \oplus (x \vee \neg z)$	`0x6ed9eba1`	25
Round 4	$x \oplus y \oplus z$	`0x8f1bbcdc`	30

We show a graph of the step function in Figure 2. Note that $R_j^{-1}(p_{j+1}, X_j)$ can be computed in almost the same complexity as that of R_j.

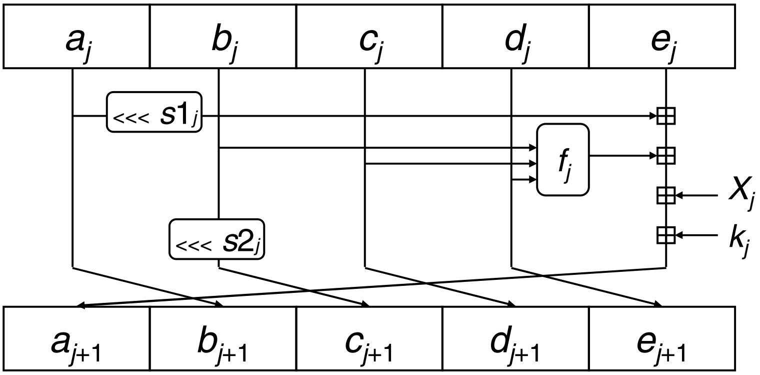


Fig. 2. Step function of HAS-160

3 Related Work

3.1 Converting Pseudo-preimages to a Preimage

For a given hash value y, pseudo-preimage is a pair of $(x, M), x \neq IV$ such that $CF(x, M) = y$. For the Merkle-Damgård hash functions, there is a generic algorithm that converts a pseudo-preimage attack to a preimage attack [9, Fact9.99]. Let the complexity of a pseudo-preimage attack be 2^k. The procedure of this attack when the hash value is n-bit long is as follows.

1. Generate $2^{(n-k)/2}$ pseudo-preimages at the complexity of $2^k \cdot 2^{(n-k)/2}$.
2. Generate $2^{(n+k)/2}$ messages that start from the IV, and compute their hash values.

One of these hash values are expected to be matched. The complexity of this attack is $2^k \cdot 2^{(n-k)/2} + 2^{(n+k)/2} = 2^{1+(n+k)/2}$. Therefore, a pseudo-preimage attack with a complexity less than 2^{n-2} can be converted to a preimage attack.

3.2 Splice-and-Cut, Partial-Matching, and Partial-Fixing Techniques

Splice-and-cut, partial-matching, and partial-fixing techniques were developed by Aoki and Sasaki [1] to launch preimage attacks on MD4 and MD5. These techniques enabled them to attack 63 steps of MD5 with a complexity of 2^{121} and full-round of the compression function of MD4 with a complexity of 2^{107}.

The splice-and-cut technique is a kind of meet-in-the-middle attack. They first consider the first and last steps as consecutive steps, and divide the attack target into two *chunks* of steps so that each chunk includes independent message words from the other chunk. Such message words are called *neutral words*. Then, a pseudo-preimage is computed by the meet-in-the-middle attack.

The partial-matching technique enables an attacker to skip several steps of an attack target when they search for good chunks. Assume that one of divided chunks provides the value of p_i, where $p_i = (a_i, b_i, c_i, d_i)$ and the other chunk provides the value of p_{i+3}, where $p_{i+3} = (a_{i+3}, b_{i+3}, c_{i+3}, d_{i+3})$, and $a_i = d_{i+3}$. p_i and p_{i+3} cannot be directly compared to be matched, however, a part of these values, that is, a_i and d_{i+3} can be compared immediately. In such a case, we can ignore messages used in Steps $i, i+1$, and $i+2$ when we run the meet-in-the-middle attack.

The partial-fixing technique enables an attacker to skip more steps. The idea is fixing a part of neutral words, for example, fixing lower 16 bits of neutral words. By this effort, the attacker can partially compute a chunk even if a neutral word for the other chunk appears.

3.3 Combination of Splice-and-Cut and Local-Collision Techniques

A combination of the meet-in-the-middle and local collision was first proposed by Aumasson et al. [2]. This was further improved by Sasaki and Aoki by combining the splice-and-cut and the local-collision techniques instead of the simple meet-in-the-middle attack [13], and attacked the full-round of HAVAL-3, HAVAL-4, and the step-reduced HAVAL-5 based on this technique.

The key idea is selecting two neutral words that can form a local collision at the beginning of chunks. This means that the attacker chooses two neutral words in suitable positions so that changes of these neutral words will be offset each other without affecting other chaining variables. To prevent the difference from propagating to the other chaining variables, values of other chaining variables are fixed so that input differences of f-function can be ignored in the output value. Such properties are called *absorption properties.*

Since a local collision of HAVAL can be formed by only changing two message words and f-function has many absorption properties, the local-collision technique can be effectively applied to HAVAL.

4 Approach 1: Simple Application of Previous Techniques

In this section, we propose a preimage attack on HAS-160 by applying the splice-and-cut, partial-matching, and partial-fixing techniques. First, we analyze how well each technique can be applied to HAS-160. Second, we show the best selection of chunks discovered by a machine experiment.

Splice-and-cut technique. Different from MD5 and HAVAL, HAS-160 uses expanded messages m_{16} to m_{19} to compute a step function. However, the application of splice-and-cut technique for HAS-160 is straightforward. For the steps where m_{16} to m_{19} are used, we consider that four message words used to compute m_{16} to m_{19} appear at the same time.

Partial-matching technique. In the step update function of HAS-160, four chaining variables out of five chaining variables are just the copy of previous chaining variables or rotation of themselves. Therefore, essentially, only one chaining variable is updated in every step. Due to this property, the partial-matching technique can skip up to four steps.

Partial-fixing technique. In the forward computation and backward computation of step update function of HAS-160, a part of updated chaining variables can be computed even if a part of a message word and chaining variables are not known. Hence, we can apply the partial-fixing technique to skip more steps. To estimate how many steps can be skipped, we need to check details of the step function. Let the symbol $A_j^{(n)}$ represent the variable A where the lower n bits are known and upper $(32-n)$ bits are unknown. We can compute following expressions.

Forward computation:

$$(a_j^{(32)}, b_j^{(32)}, c_j^{(32)}, d_j^{(32)}, e_j^{(32)}), X_j^{(n)} \xrightarrow{R_j} (a_{j+1}^{(n)}, b_{j+1}^{(32)}, c_{j+1}^{(32)}, d_{j+1}^{(32)}, e_{j+1}^{(32)})$$

Inverse computation:

$$(a_{j+1}^{(32)}, b_{j+1}^{(32)}, c_{j+1}^{(32)}, d_{j+1}^{(32)}, e_{j+1}^{(32)}), X_j^{(n)} \xrightarrow{R_j^{-1}} (a_j^{(32)}, b_j^{(32)}, c_j^{(32)}, d_j^{(32)}, e_j^{(n)})$$

$$(a_{j+1}^{(32)}, b_{j+1}^{(32)}, c_{j+1}^{(32)}, d_{j+1}^{(32)}, e_{j+1}^{(n)}), X_j^{(n)} \xrightarrow{R_j^{-1}} (a_j^{(32)}, b_j^{(32)}, c_j^{(32)}, d_j^{(n)}, e_j^{(n)})$$

$$(a_{j+1}^{(32)}, b_{j+1}^{(32)}, c_{j+1}^{(32)}, d_{j+1}^{(n)}, e_{j+1}^{(n)}), X_j^{(n)} \xrightarrow{R_j^{-1}} (a_j^{(32)}, b_j^{(32)}, c_j^{(n)}, d_j^{(n)}, e_j^{(n)})$$

$$(a_{j+1}^{(32)}, b_{j+1}^{(32)}, c_{j+1}^{(n)}, d_{j+1}^{(n)}, e_{j+1}^{(n)}), X_j^{(n)} \xrightarrow{R_j^{-1}} (a_j^{(32)}, b_j^{(n-s2_j)}, c_j^{(n)}, d_j^{(n)}, e_j^{(n-s2_j)})$$

Finally, we can conclude that nine steps in total can be skipped by combining the partial-matching and partial-fixing techniques. Note, a few more steps can be skipped with a lower probability if the number of $s2_j$ is suitable. However, we omit details of this analysis since any suitable application could not be found.

Preimage attacks on 48-step HAS-160 by approach 1. By considering all of the techniques described in this section, we searched for the best selection of chunks. As a result, we found two selections of chunks that can attack 48 steps out of 80 steps. These selections are shown in Figures 4 and 5 at the last part of the paper.

We explain attack details, where the corresponding selection of chunks is shown in Figure 4. We choose Steps 1-48 as an attack target and divide it so that Steps 1-13 and 32-48 are the first chunk with a neutral word m_8, Steps 14-28 are the second chunk with a neutral word m_{11}, and Steps 29-31 are skipped. Our attack first finds pseudo-preimages, and convert them to a preimage. Therefore, our attack finds a 2-block preimage, so first, the appropriate padding strings for 2-block messages are set in m_{13}, m_{14}, and m_{15}. For a given hash value H_2, an attack procedure is as follows. Note, since the number of skipped steps is only three, we do not use the partial-fixing technique.

Attack procedure

1. Fix $m_i (i \notin \{8, 11, 13, 14, 15\})$ and p_{14} to randomly chosen values.
2. For all m_{11}, do the following:

$$p_{j+1} \leftarrow R_j(p_j, X_j) \qquad \text{for } j = 14, 15, \ldots, 28.$$

3. Make a table of (m_{11}, p_{29})s which are computed in the last step.
4. For all m_8, do the following:

$$\begin{cases} p_j \;\leftarrow R_j^{-1}(p_{j+1}, X_j) & \text{for } j = 13, 12, \ldots, 1, \\ p_{49} \leftarrow H_2 - p_1, & \\ p_j \;\leftarrow R_j^{-1}(p_{j+1}, X_j) & \text{for } j = 48, 47, \ldots, 32. \end{cases}$$

   and check whether $(d_{32} \qquad s2_{30})$ and $(e_{32} \qquad s2_{29})$ are matched with a_{29} and b_{29} in the table.
5. If matched, we compute p_{31} to p_{29} by the corresponding m_i, and check whether all values are matched.
6. If matched, the corresponding message and p_1 is a pseudo-preimage.

The computational complexity of the above attack procedure is about 2^{32} $(= 2^{32}\frac{15}{48} + 2^{32}\frac{30}{48})$, and the success probability is about 2^{-96} $(= 2^{32} \cdot 2^{32}/2^{160})$. Thus, by iterating the above procedure 2^{96} times, we expect to find a pseudo-preimage (H_1, M_1), and its complexity is about 2^{128} $(= 2^{96} \cdot 2^{32})$. By applying the technique in Section 3.1, a preimage of 48-step HAS-160 can be computed in $2^{145}(= 2^{1+(160+128)/2})$.

To generate a pseudo-preimage, a memory is used to store 2^{32} (m_{11}, p_{29})s in step 3 of the above procedure. To generate a preimage, 2^{16} pseudo-preimages are stored. Therefore, the memory complexity of this attack is $2^{32} \times 6$ words.

5 Approach 2: Extension of Local-Collision Technique

As we considered in Section 4, no more than 48 steps can be attacked only with the previous techniques. Therefore, we need to consider another approach to extend the attack.

The idea is using the local-collision technique, which was considered by Sasaki and Aoki [13] to attack HAVAL. Their success lies in the specific structure of HAVAL, where a local collision can be formed with a straightforward method and f-function has many absorption properties. However, making a local collision in HAS-160 is more difficult than HAVAL since three message words are needed to make a local collision in HAS-160 and absorption properties are available only in the first round and a part of the third round. In fact, we searched for good chunks under such strong limitations, but no good chunk was discovered. However, we found that by selecting X_{29} and X_{31} as neutral words, Steps 29-31 can be considered to have the same properties as a local collision.

In this section, we first explain the overall strategy of our attack. Then, we explain why Steps 29-31 have the same properties as a local collision. Finally, we explain the attack procedure using the property of steps 29-31 and complexity estimation.

5.1 Overall Strategy

We show the overall strategy by using Figure 3 showing Steps 29-31. Note, in Step 30, we applied an equivalent transformation that changes the order of addition of $e_{30} + f_{30} + (a_{30} \qquad s1_{30})$.

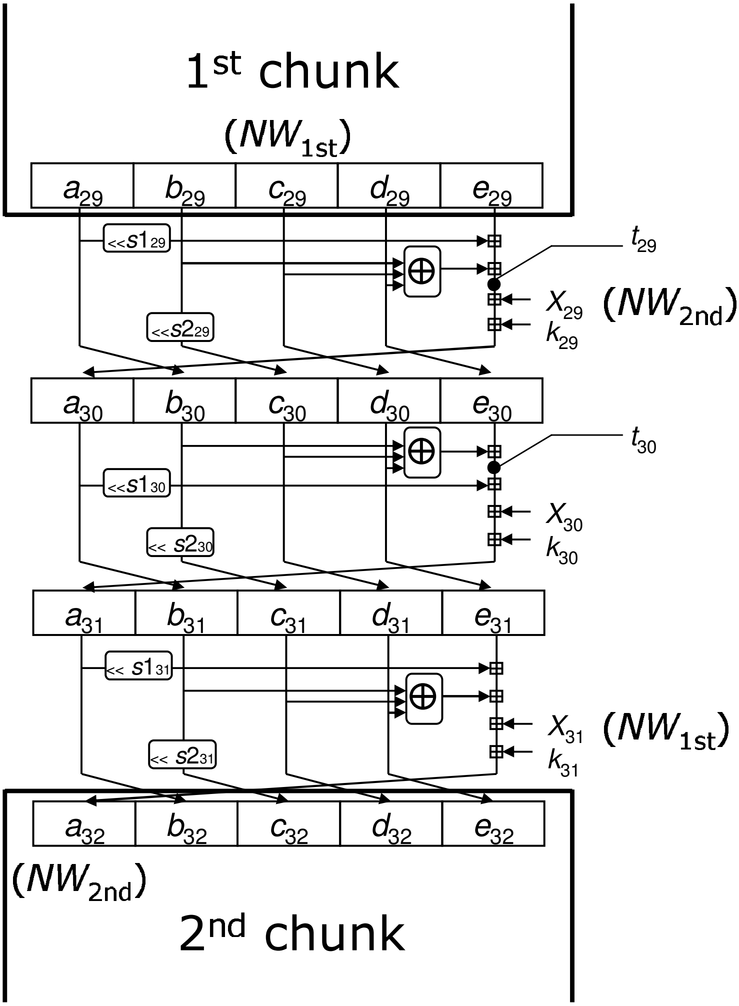


Fig. 3. Steps 29-31 forming a structure similar to local collision

1. We choose c_{29} and X_{31} as neutral words for the first chunk and a_{32} and X_{29} as neutral words for the second chunk. Therefore, both chunks have 2^{64} free bits.
2. For each choice of (c_{29}, X_{31}), compute $a_{29}, b_{29}, d_{29}, e_{29}$ so that the choice of (c_{29}, X_{31}) does not give any influence to the value of the second chunk, namely, the values of $b_{32}, c_{32}, d_{32}, e_{32}$.
3. For each choice of (X_{29}, a_{32}), compute $b_{32}, c_{32}, d_{32}, e_{32}$ so that the choice of (X_{29}, a_{32}) does not give any influence to the value of the first chunk, namely, the values of $a_{29}, b_{29}, d_{29}, e_{29}$.
4. Compute the first chunk for all free bits of (c_{29}, X_{31}) and compute the second chunk for all free bits of (X_{29}, a_{32}). Finally, perform the meet-in-the-middle attack to search for pairs where results of the chunk computation are matched and $(c_{29}, X_{31}, X_{29}, a_{32})$ can correctly satisfy their relationship in Steps 29-31.

Step	Message index																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	*																first chunk ←
2		*															
3			*														
4				*													
5													*	*	*	*	
6					*												
7						*											
8							*										
9								*									
10	*	*	*	*													
11									○								
12										*							
13											*						
14												●					→second chunk
15					*	*	*	*									
16													*				
17														*			
18															*		
19																*	
20		*			*							●			*		
21				*													
22							*										
23										*							
24													*				
25	*							*			*			*			
26																*	
27			*														
28						*											
29									○								skip
30				*			*			*			*				
31												●					
32															*		first chunk
33		*															
34					*												
35			*			*			○							*	
36								*									
37											*						
38														*			
39	*																
40					*		*							*		*	
41													*				
42						*											
43															*		
44								*									
45		*		*					○		*						
46	*																
47										*							
48			*														

Fig. 4. Selected chunks by approach 1 (Type 1)

Therefore, we need to realize the independency of $a_{29}, b_{29}, d_{29}, e_{29}$ and neutral words for the second chunk, and the independency of $b_{32}, c_{32}, d_{32}, e_{32}$ and neutral words for the first chunk.

5.2 Computation for Steps 29 to 31

To make the first and second chunk independent each other, we need to guarantee that a_{29} and d_{32}, b_{29} and e_{32}, d_{29} and b_{32}, and e_{29} and c_{32} are always consistent regardless of the values of neutral words. This can be achieved as follows.

Step	Message index																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
21				*													first chunk
22							*										
23										*							
24													*				
25	*							*			*			*			
26																*	
27			*														
28						*											
29									*								
30				*			*			*			*				
31												●					
32															*		
33		*															←
34					○												→second chunk
35			*			*			*							*	
36								*									
37											*						
38														*			
39	*																
40					○		*							*		*	
41													*				
42						*											
43															*		
44								*									
45		*		*					*		*						
46	*																
47										*							
48			*														
49												●					skip
50						*		*					*		*		
51					○												
52														*			first chunk
53							*										
54																*	
55	*		*							*		●					
56									*								
57		*															
58											*						
59				*													
60	*					*					*					*	
61								*									
62			*														
63														*			
64									*								
65		*					*					●	*				
66				*													
67															*		
68										*							

Fig. 5. Selected chunks by approach 1 (Type 2)

- We fix a_{29} and b_{29} to a randomly chosen value. Then, we fix d_{32} to $(a_{29} \quad s2_{30})$ and e_{32} to $(b_{29} \quad s2_{29})$ to guarantee the consistency of these variables.
- To guarantee that the relationship of d_{29} and b_{32} are satisfied, we fix the value of t_{30}, which is indicated in Figure 3. Since b_{30} and c_{30} are fixed values, the value of $f_{30}(b_{30}, c_{30}, d_{30})$ are uniquely determined by the selection of $c_{29}(= d_{30})$. Therefore, for each selection of c_{29}, we compute $e_{30} \leftarrow t_{30} - f_{30}$ so that t_{30} is fixed. Note, when we compute a_{31} for the second chunk, we use the expression $a_{31} \leftarrow (a_{30} \quad s1_{30}) + t_{30} + X_{30} + k_{30}$.

Step	Message index																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
12										*							first chunk
13											*						
14												●					
15					*	*	*	*									
16													*				
17														*			
18															*		
19																*	
20		*			*							●			*		
21				*													
22							*										
23										*							
24													*				
25	*							*			*			*			
26																*	
27			*														
28						*											↑
29									○								skip
30				*			*			*			*				
31												●					
32															*		↓ second chunk
33		*															
34					*												
35			*			*			○							*	
36								*									
37											*						
38														*			
39	*																
40					*		*							*		*	
41													*				
42						*											
43															*		
44								*									
45		*		*					○		*						
46	*																
47										*							
48			*														
49												●					skip
50						*		*					*		*		
51					*												
52														*			
53							*										
54																*	
55	*		*							*		●					
56									○								
57		*															first chunk
58											*						
59				*													
60	*					*					*					*	
61								*									
62			*														
63														*			

Fig. 6. Selected chunks by approach 2 (52 steps)

– To guarantee that the relationship of e_{29} and c_{32} are satisfied, we fix the value of t_{29}, which is indicated in Figure 3. Considering a_{29}, b_{29} are fixed, and d_{29} is uniquely computed for each c_{29}, for each selection of c_{29}, we compute $e_{29} \leftarrow t_{29} - f_{29} - (a_{29} \quad s1_{29})$ so that t_{29} is fixed. Note, when we compute a_{30} for the second chunk, we use the expression $a_{30} \leftarrow (a_{29} \quad s1_{29}) + t_{29} + X_{29} + k_{29}$.

The relationship of $(c_{29}, X_{31}, X_{29}, a_{32})$ can be written as follows. Neutral words for the first and second chunks are stressed by $\underline{}_{\text{1st}}$ and $\underline{}_{\text{2nd}}$, respectively.

$$\begin{aligned} \underline{a_{32}}_{\text{2nd}} &\overset{?}{=} \underline{c_{29}}_{\text{1st}} + (a_{31} \qquad s1_{31}) + f_{31}(b_{31}, c_{31}, d_{31}) + \underline{X_{31}}_{\text{1st}} + k_{31}, \qquad (1) \\ \text{where, } a_{31} &= (a_{30} \qquad s1_{30}) + t_{30} + X_{30} + k_{30}, \\ a_{30} &= (a_{29} \qquad s1_{29}) + t_{29} + \underline{X_{29}}_{\text{2nd}} + k_{29}. \end{aligned}$$

This equation is satisfied with a probability of 2^{-32} for a randomly chosen $(c_{29}, X_{31}, X_{29}, a_{32})$.

5.3 Preimage Attack on 52-Step HAS-160

The entire structure of our attack on 52-step HAS-160 is shown in Figure 6. This attack finds a pseudo-preimage of 52 steps (Steps 12-63) of HAS-160 with a complexity of 2^{144}, and a preimage with a complexity of 2^{153}. The attack is a 2-block attack. The procedure for a given target hash value H_2 is as follows.

Attack procedure

1. Fix $m_i, (i \notin \{8, 11, 13, 14, 15\})$ and lower 16 bits of m_8 to randomly chosen values and fix m_{11} in bit positions 7-22 to be 0. Fixed bit positions are optimized for matching decision, namely, they are fixed so that their positions become lower 16 bits after $s2_{51} (= 25)$ left rotation.
2. Fix chaining variables in steps 29-31 as described in Section 5.2.
3. For all 32 bits of a_{32} and higher 16 bits of m_8,
   (a) Compute a_{30} and a_{31} as described in Section 5.2.
   (b) Compute followings:
   $$p_{j+1} \leftarrow R_j(p_j, X_j) \qquad \text{for } j = 32, 33, \ldots, 48.$$
   (c) Compute 16 bits of a_{50} by using partially fixed m_{11}. This computation is an addition operation. Since we know only bits 7-22 of m_{11}, we cannot know the carry effect from bit 6 to bit 7. Therefore, we compute 16 bits of a_{50} for both cases.
   (d) When a_{32} and higher 16-bits of m_8 are selected, equation 1 can be written as
   $$\underline{c_{29}}_{\text{1st}} + \underline{X_{31}}_{\text{1st}} \overset{?}{=} Const_{\text{2nd}}, \qquad (2)$$
   for a uniquely computed $Const_{\text{2nd}}$. Compute the value of such a $Const_{\text{2nd}}$.
   (e) Make a table of $(m_8, p_{49}, (16\text{-bits of } a_{50} \qquad s2_{51}), Const_{\text{2nd}})$s. Since we have 48 free bits in neutral words, and 2 candidates of partial a_{50}, we have 2^{49} items in the table.
4. For all 32-bits of c_{29} and not-fixed 16-bits of m_{11}, do the following:
   (a) Compute e_{30} and e_{29} as described in Section 5.2.

(b) Do the following:

$$\begin{cases} p_j & \leftarrow R_j^{-1}(p_{j+1}, X_j) \quad \text{for } j = 28, 27, \ldots, 12, \\ p_{64} & \leftarrow H_2 - p_{12}, \\ p_j & \leftarrow R_j^{-1}(p_{j+1}, X_j) \quad \text{for } j = 63, 62, \ldots, 57. \end{cases}$$

(c) Compute the lower 16-bits of e_{56}, e_{55}, and e_{54} with the partially fixed m_8.

(d) Compute $Const_{1\text{st}} \leftarrow \underline{c_{29}}_{1\text{st}} + \underline{X_{31}}_{1\text{st}}$.

(e) For each item in the table, check whether the lower 16-bits of e_{54} and $Const_{1\text{st}}$ are matched with 16-bits of $(a_{50} \quad s2_{51})$ and $Const_{2\text{nd}}$, respectively.

(f) If matched, compute p_{50} to p_{57} by the corresponding message word, and check whether all values from both chunks are matched. This is performed step by step.

(g) If all bits are matched, the corresponding message and p_{12} is a pseudo-preimage.

The computational complexity of steps 3a, 3b, 3c, and 3d of the procedure is $2^{48}\frac{2}{52} + 2^{48}\frac{17}{52} + 2^{48}\frac{1}{52} + 2^{48}\frac{3}{52}$, and the complexity of steps 4a, 4b, 4c, and 4d is $2^{48}\frac{2}{52} + 2^{48}\frac{24}{52} + 2^{48}\frac{3}{52} + 2^{48}\frac{1}{52}$. In step 4e, $2^{97}(= 2^{49} \cdot 2^{48})$ pairs are tested to be matched and $2^{49}(= 2^{97} \cdot 2^{-48})$ pairs will be matched. In step 4f, computation of p_{50} costs $2^{44}(\approx 2^{49}\frac{1}{52})$, and 2^{48} pairs out of 2^{49} pairs will be left by checking the correctness of the carry in a_{50}. Then, computation of p_{51} costs $2^{43}(\approx 2^{48}\frac{1}{52})$, and 2^{32} pairs out of 2^{48} pairs will be left by partially matching a_{51} and e_{55}. Then, computation of p_{52} costs $2^{32}\frac{1}{52}$, and 2^{16} pairs will be left by partially matching a_{52} and e_{56}. Complexity for computing of p_{53} and p_{57} are negligible, and finally 2^{-96} pair will be matched for all bits. Therefore, by repeating the above procedure 2^{96} times, a pair will be matched for all bits, in other words, a pseudo-preimage will be found. The complexity of the attack is approximately $2^{144}(= 2^{48} \times 2^{96})$. At the last, this pseudo-preimage attack is converted to a preimage attack with a complexity of 2^{153} by using the conversion algorithm described in Section 3.1.

In this attack, we use a memory to store 2^{49} $(m_8, p_{49}, (\text{16-bits of } a_{50} \quad s2_{51}), Const_{2\text{nd}})$s in step 3e. This memory size can be reduced by storing both a_{50} with carry and a_{50} without carry in the same address of the table. Therefore, the memory complexity of this attack is $2^{48} \times 9$ words.

6 Conclusion

In this paper, we proposed two approaches for the preimage attacks on the hash function HAS-160. The first approach is a simple application of previously known techniques, which generates a pseudo-preimage of 48 steps of HAS-160 with a complexity of 2^{128} and a preimage with a complexity of 2^{145}. To further improve the number of steps that can be attacked, we considered a variant of the local-collision technique to skip several steps at the beginning of the chunks. This enabled us to find a pseudo-preimage of 52 steps of HAS-160 with a complexity of 2^{144} and a preimage with a complexity of 2^{153}. To the best of our knowledge, this is the first paper that analyzes the preimage resistance of HAS-160.

References

1. Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Workshop Records of SAC 2008, Sackville, Canada, pp. 82–98 (2008)
2. Aumasson, J.-P., Meier, W., Mendel, F.: Preimage attacks on 3-pass HAVAL and step-reduced MD5. In: Workshop Records of SAC 2008, Sackville, Canada, pp. 99–114 (2008); ePrint version is available at IACR Cryptology ePrint Archive: Report 2008/183, `http://eprint.iacr.org/2008/183.pdf`
3. De Cannière, C., Rechberger, C.: Preimages for Reduced SHA-0 and SHA-1. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 179–202. Springer, Heidelberg (2008); slides on preliminary results were appeared at ESC 2008 seminar, `http://wiki.uni.lu/esc/`
4. Cho, H.-S., Park, S., Sung, S.H., Yun, A.: Collision search attack for 53-step HAS-160. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 286–295. Springer, Heidelberg (2006)
5. De, D., Kumarasubramanian, A., Venkatesan, R.: Inversion attacks on secure hash functions using SAT solvers. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 377–382. Springer, Heidelberg (2007)
6. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
7. Mendel, F., Rijmen, V.: Colliding message pair for 53-step HAS-160. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 324–334. Springer, Heidelberg (2007)
8. Mendel, F., Rijmen, V.: Weaknesses in the HAS-V Compression Function. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 335–345. Springer, Heidelberg (2007)
9. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press, Boca Raton (1997)
10. Park, N.K., Hwang, J.H., Lee, P.J.: HAS-V: A new hash function with variable output length. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 202–216. Springer, Heidelberg (2001)
11. Rivest, R.L.: Request for Comments 1321: The MD5 Message Digest Algorithm. The Internet Engineering Task Force (1992), `http://www.ietf.org/rfc/rfc1321.txt`
12. Sasaki, Y., Aoki, K.: Preimage attacks on step-reduced MD5. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 282–296. Springer, Heidelberg (2008)
13. Sasaki, Y., Aoki, K.: Preimage attacks on 3, 4, and 5-pass HAVAL. In: Pieprzyk, J.P. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 253–271. Springer, Heidelberg (2008)
14. Telecommunications Technology Association. Hash Function Standard Part 2: Hash Function Algorithm Standard, HAS-160 (2000)
15. U.S. Department of Commerce, National Institute of Standards and Technology. Announcing the SECURE HASH STANDARD (Federal Information Processing Standards Publication 180-3) (2008), `http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf`
16. Yun, A., Sung, S.H., Park, S., Chang, D., Hong, S.H., Cho, H.-S.: Finding collision on 45-step HAS-160. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 146–155. Springer, Heidelberg (2006)

Essentially Optimal Universally Composable Oblivious Transfer

Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi

BRICS, Department of Computer Science, Aarhus University,
Åbogade 34, 8200 Århus, Denmark
{ivan,jbn,orlandi}@cs.au.dk

Abstract. Oblivious transfer is one of the most important cryptographic primitives, both for theoretical and practical reasons and several protocols were proposed during the years. We propose a protocol which is simultaneously optimal on the following list of parameters: *Security:* it has universal composition. *Trust in setup assumptions:* only one of the parties needs to trust the setup (and some setup *is* needed for UC security). *Trust in computational assumptions:* only one of the parties needs to trust a computational assumption. *Round complexity:* it uses only two rounds. *Communication complexity:* it communicates $\mathcal{O}(1)$ group elements to transfer one out of two group elements. The Big-O notation hides 32, meaning that the communication is probably not optimal, but is essentially optimal in that the overhead is at least constant. Our construction is based on pairings, and we assume the presence of a key registration authority.

Keywords: Oblivious Transfer, Universally Composable Security.

1 Introduction

An oblivious transfer (OT) involves two parties, a sender and a receiver. The sender has two secret messages. The receiver selects to retrieve one of them, without disclosing which one. At the same time the receiver is not allowed to learn more than one secret. Oblivious transfer was first introduced by Wiesner [Wie83] in the late seventies under the name of *conjugate coding*. However, the importance of this primitive in the cryptographic field was first pointed out by Rabin in [Rab81].

OT is the base for many secure multiparty computation (SMC) protocols [Yao86, GMW87], where several instances of OT are run at the same time. However, many of the proposed OT protocols do not give any guarantee on the security under composition. Our protocol is secure in the universally composable [Can01] model, using just two rounds of communication and having only a constant overhead, i.e. $\mathcal{O}(k)$ bits are communicated to do an OT of k bits.

In addition we do not need to assume a common reference string. Instead, the receiver R once and for all has to register a public key with a key registration authority KR and prove to KR that he knows the corresponding secret key. After this any sender S can retrieve the public key and perform an OT to

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 318–335, 2009.

R — in particular, S does not need a public key or trust any common reference string, giving our construction the same flavor as a PKI for public-key encryption, where also just R registers a public key, after which all S can transfer messages securely to R. The public-key flavor of our protocol makes it ideal for asymmetric settings, where e.g. one party is a server, which can afford the time and cost of registering a public key for the given application. Clients then only need to retrieve the public key of the server to perform UC OT with the server.

As a final feature, our protocol is perfectly secure for the sender. It can therefore be viewed as optimal in four aspects:

1. It is secure under general composition.
2. It uses two rounds. Clearly there exists no one-round OT protocol.
3. Only one party has to trust the setup — S has to trust that KR checks that R knows its secret key. Since OT is impossible in the plain UC model, some setup must be trusted, and having only one party do so is optimal.
4. Only one party has to trust a computational assumption. In particular, S has perfect security. No OT for the classical model can have perfect security for both parties.

The communication complexity is probably not optimal. Under the decisional linear (DLIN) assumption, we send 32 group elements to transfer one out of two group elements. This gives a constant overhead, but is probably far from optimal. This seems, so far, to be the unfortunate price to pay for the other fully optimal properties.

Our OT protocol is primarily based on homomorphic encryption in pairing friendly groups, which we use to give a new instantiation of the notion of mixed commitments from [DN02]. This instantiation is constructed to work well with the efficient non-interactive witness indistinguishable (NIWI) proofs by Groth, Ostrovsky and Sahai [GOS06, GS08], which are in turn based on paring-based cryptography. We put ourselves in the hybrid UC model, where all parties have access to secure and authenticated channels and to a key registration authority (KR). This model was presented and motivated in [BCNP04].

Related Work. Examples of two-round OT can be found in [NP01, AIR01, Kal05]. However, none of these protocols achieve UC security. If we consider UC security, OT protocols are known, but they require more rounds of interaction [Gar04, JS07] or other parties helping the computation [Fis06].

As a witness that a secure and efficient OT is of primary importance, several attempts were made in the last years. Lindell [Lin08] has a very general construction that achieves full simulation, based on the existence of homomorphic encryption solely. Camenisch, Neven and shelat [CNS07] built a protocol for adaptive k-out-of-n OT, providing full simulation with specific number-theoretic assumptions. Upon this work Green and Hohenberger [GH07] constructed another adaptive OT, which requires weaker assumptions. Recently the same authors proposed another adaptive OT [GH08], this time achieving UC security.

Independently from our work, Peikert, Vaikuntanathan and Waters [PVW07] presented a two round UC OT protocol. However, their protocol works in the common reference string model (CRS), and uses different computational assumptions. Therefore, these two works can be seen as complementary.

2 Main Ideas

In this section we are going to give the main ideas, leaving all the details to the following sections. We first present an attack that motivates the need of a composable OT, then we sketch the protocol.

We have two players called the sender and the receiver. The sender has two secrets x_0, x_1, while the receiver has a selection bit b. At the end of the protocol R gets x_b while S gets nothing.

2.1 Insecurity of OT Composition

We can describe a round optimal OT (i.e. 2 round OT) in the following way:

Choose: R computes a message $c = \text{Choose}(b)$, and sends c to S.
Transfer: S computes a message $t = \text{Transfer}(c, x_0, x_1)$ and sends it to R.
Retrieve: R retrieves $x_b = \text{Retrieve}(t)$.[1]

The security of such a protocol is usually stated like:

Receiver's privacy: the output of the Choose phase, c, does not reveal any information about b to S.

Sender's privacy: the output of the Transfer phase, t, does not reveal anything about x_{1-b} to R.

This kind of security definition works in the case of a stand-alone OT execution, but fails dramatically in the case of even a sequential composition, and even for very generic reasons. We consider the following composed protocol to illustrate it: R and S run a first OT protocol. R inputs b, S inputs x_0, x_1, and R gets x_b. Then R and S run a second OT protocol. R inputs b', S inputs x'_0, x'_1, and R gets $x'_{b'}$. Finally R sends $x'_{b'}$ to S. Now instantiate this protocol with a two-message OT which is secure according to the previous definition:

1. (a) R computes $c = \text{Choose}(b)$, and sends c to S.
   (b) S computes $t = \text{Transfer}(c, x_0, x_1)$ and sends it to R.
   (c) R retrieves $x_b = \text{Retrieve}(t)$.
2. (a) R computes $c' = \text{Choose}(b')$, and sends c' to S.
   (b) S computes $t' = \text{Transfer}(c', x'_0, x'_1)$ and sends it to R.
   (c) R retrieves $x'_{b'} = \text{Retrieve}(t')$.
3. R sends $x'_{b'}$ to S.

[1] Note that this is the only possible order of the messages. If we build a protocol where S sends the first message and then R computes x_b from this message, then clearly R can choose to learn both x_0 and x_1.

A cheating S could use the first Choose message in the second Transfer phase, i.e. he could compute $t' = \text{Transfer}(c, x'_0, x'_1)$. Therefore, R retrieves x'_b instead of $x'_{b'}$ without noticing it. In the 3rd step, he will send x'_b to S, clearly revealing information about b, which an ideal implementation would not.

Note that, despite the fact that the protocol presented is an ad-hoc constructed counterexample, this vulnerability is actually quite important and has many consequences: when parties run more OT instances, the receiver cannot be sure that the Transfer messages contain his choice. A protocol that is not secure against this attack is the one in [AIR01].

Intuitively this problem arises from the fact that the security of the sender and the security of the receiver are analyzed separately, and therefore there is no "link" between the Choose phase and the Transfer phase[2]. Another common definition for the security of OT protocols is the half-simulation, as in [NP05]. In this scenario we usually require strong (simulation) security against the receiver, but just stand-alone privacy against the sender. Note that this would not protect against the attack sketched above. This relaxation is usually justified by saying that the sender is commonly a server or a service provider, and therefore it can be controlled better or more than the receiver, who represents any user. As the above example shows, this motivation assumes that the server chooses not to learn information, which he could in fact learn by deviating only so slightly from the implementation.[3] Under such an assumption (essentially that the server is at most passively corrupted) things become much simpler. Here we want active security for both parties.

2.2 Our Protocol

We are going to present the main intuition behind our protocol in 5 steps.

Step 1: OT based on Homomorphic Encryption. Assume an additively homomorphic cryptosystem is available, i.e. a cryptosystem that satisfies the following:

[2] A way to fix this problem, as some OT protocols do, is to change the structure of the protocol, allowing Choose to output also a piece of trapdoor information k that will be later used during the Retrieve phase. In this case, the protocol will be of the form: $(c, k) = \text{Choose}(b); t = \text{Transfer}(c, x_0, x_1); x_b = \text{Retrieve}(t, k)$. We prefer, instead of fixing just this problem, to develop our protocol in the UC framework, for it provides us stronger guarantees. In particular, UC security protects against ill effects of composition as those described above, while still allowing us to analyze the protocol in isolation.

[3] This motivation is also not so strong given that in several applications the role of the sender and the receiver can be swapped. Moreover, in some applications like authentication, it is the server that plays the role of the receiver, while the user plays the role of the sender. This could in principle be handled by using that OT is symmetric: an OT from S to R can be turned into an OT from R to S without further assumptions. This transformation however, adds another round of communication, and it always produces a one-bit OT. For applications where two-round OT or string-OT is needed, "turning the OT around" is therefore not a practical solution.

$D(E(x)E(y)) = x + y$, where E, D represent the encryption and the decryption functions[4]. Then the following is a simple OT construction, if the parties are semi-honest:

Choose: The receiver encrypts $c_1 = E(b)$ and sends it to the sender.
Transfer: The sender computes $c_0 = E(1)c_1^{-1} = E(1 - b)$ and $d = c_0^{x_0} c_1^{x_1}$ and sends d it to the receiver.
Retrieve: The receiver decrypts $x_b = D(d)$.

The idea is that the receiver lets (c_0, c_1) be an encryption of the vector $(1, 0)$, if he wants to get the first secret or $(0, 1)$, if he wants the second one. The sender computes, exploiting the homomorphic property: $c_0^{x_0} c_1^{x_1}=E((1–b)x_0+bx_1)=E(x_b)$.

Step 2: Managing a Malicious Receiver. The protocol is intuitively correct and private, if both parties follow the protocol. However, in the case of malicious adversaries there are many evident security flaws. First of all, if the receiver is not honest, he could send an encryption of $b \notin \{0, 1\}$. If for instance he sends an encryption of $b = 2$, at the end of the protocol he will get $2x_1 - x_0$, which leaks both x_0 and x_1 if they are bits. Therefore the receiver has to prove that the message is well formed, by using a NIWI proof.

Step 3: Managing a Malicious Sender. A sender, even behaving maliciously, cannot break the privacy of the receiver without breaking the security of the underlying encryption scheme. In fact, he just sees a ciphertext. The receiver, however, has no way to check if the sender is inputting a "fresh" ciphertext, obtained through the expected computation, or just a chosen ciphertext that came from somewhere else. The actual problem is that we do not check if the sender knows what he is inputting or not, and so the output may not even depend at all on the Choose message. This kind of problem does not allow the sender to learn more than what he is supposed to in a single execution of the protocol. But, as described in Section 2.1, if more than one instance of the protocol is run, the loss of security is dramatic. Therefore, we have to ask also the sender to prove that the message is well formed, using a NIWI proof.

Step 4: Achieving UC. To achieve UC we need to be able to simulate the view of the parties in the real protocol, by having access just to the ideal functionality. It is well known that it is impossible to achieve UC OT in the plain model [CF01], and therefore we will assume to have access to a KR authority [BCNP04]. The protocol then consists of two phases. In the registration phase the receiver once and for all registers an encryption key ek at the KR and proves that he knows the corresponding decryption key dk. In the communication phase the parties then perform the actual OTs using the encryption scheme $E = E_{ek}$. In the simulation it is the simulator who simulates the KR authority, which allows it to extract the decryption key dk known by the receiver. This allows the simulator to compute from c_1 the choice bit used by a corrupted receiver.

[4] Note that by repeating this operation we can also achieve that $D(E(x)^a) = ax$.

In the case of a malicious sender things are more difficult. In fact, to be able to simulate, we need to extract both x_0, x_1 from the message d. But a well formed d contains no information at all about one of the two secrets. To deal with this we need the receiver to register another key ck with the KR, where ck is a public key for a commitment scheme. Then the sender commits to x_0 and x_1 under ck and gives a NIWI proof that the commitments indeed contain messages used to compute the reply d from c_0, c_1. By using a perfectly hiding commitment scheme, the receiver will not be able to learn anything from these commitments in the real protocol. In the simulation we will, however, let the simulator cheat and use an extractable commitment scheme, which allows it to extract x_0 and x_1 from the commitments.

Step 5: The Final Protocol. The UC functionality we want to implement is:

Choose: R inputs $(\texttt{choose}, otid, b)$ to the ideal functionality, where *otid* must be a fresh OT identifier.

Transfer: The ideal functionality outputs $(\texttt{chosen}, otid)$ to S. S can then any number of times input a message of the form $(\texttt{transfer}, otid, x_0, x_1)$ into the ideal functionality.

Retrieve: Each time the ideal functionality outputs $(\texttt{retrieve}, S, otid, x_b)$ to R.

This ideal functionality slightly generalizes the standard one in that S can transfer several times using the same Choose message from R. This e.g. allows OT of longer strings efficiently. Collecting the above five steps we get the following protocol:

Key-Registration: R registers an encryption key *ek* and a commitment key *ck* and proves that he knows the decryption key *dk* (corresponding to *ek*), and that *ck* is a key for a perfect hiding commitment scheme. If so, the public keys *ek* and *ck* are given to S.

Choose: R gets his selection bit b. He encrypts $c_1 = E_{ek}(b)$. He computes a NIWI proof π_{choose} that c_1 contains either 0 or 1, and sends (c_1, π_{choose}) to S.

Transfer: S gets his two inputs x_0, x_1. He gets c_1, π_{choose} and he aborts if π_{choose} is invalid. If not, he computes $c_0 = E(1)c_1^{-1}$ and $d = c_0^{x_0} c_1^{x_1}$. Then he computes $C_0 = \mathrm{Comm}_{ck}(x_0), C_1 = \mathrm{Comm}_{ck}(x_1)$. He finally computes a NIWI $\pi_{transfer}$ that proves that d is computed correctly from c_1 and the values inside the commitments C_0, C_1. He sends everything to R.

Retrieve: R gets $d, C_0, C_1, \pi_{transfer}$ and he aborts if the check on $\pi_{transfer}$ fails. If not, he decrypts $D_{dk}(d) = x_b$ and outputs it.

There are a number of technical issues which we solved to make the above approach work: The "encryption scheme" used is in fact a mixed commitment scheme, that is built on top of Boneh, Boyen and Shacham [BBS04] cryptosystem. Next, this mixed commitment scheme has no efficient decryption, i.e. the receiver gets an element of the form g^{x_b}, with the message x_b that the sender inputs in the OT. We will therefore start with the description of a bit OT protocol, where this is clearly not an issue. If we want to transmit more data, we can think of our protocol as a random OT, where the sender picks x_0 and x_1 at random and the receiver receives $K_0 = g^{x_0}$ or $K_1 = g^{x_1}$. Even though the sender cannot choose K_0 and K_1 as he desires, he can compute them on his side. Together with the second message the sender then sends $E_{K_0}(m_0)$ and $E_{K_1}(m_1)$, where (m_0, m_1) are the actual messages of the OT and $E_{K_b}(m_b)$ is an encryption of m_b under the key K_b. In this way we avoid the discrete logarithm problem. Another issue is that the NIWI proofs from [GOS06, GS08] work in the CRS model, while we prefer to put ourselves in the KR model to have just one party trust the setup. We deal with this by noting that the NIWI proofs can be instantiated with either perfect soundness or perfect WI, depending on how the CRS is created. We let R register two CRSs, one of each flavor, as part of his public key. When R proves, he uses the scheme with perfect soundness. When S proves, he uses the scheme with perfect witness indistinguishable.

3 Preliminaries

3.1 Pairing-Based Cryptography

In the last years pairing-based cryptography gained more and more interest. Since its introduction in [Jou00][5], pairings were used in several applications and allowed to achieve strong goals, like IBE [BF01].

In pairing-based cryptography we can define bilinear maps between groups of points on elliptic curves as follows:

Definition 1. *Let $\mathbb{G}, \mathbb{G}_1$, be two multiplicative cyclic groups of finite order n, and let g be a generator for $\mathbb{G}$. Then we say that $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map if:*

Bilinear: *e is bilinear, i.e., for all $x, y \in \mathbb{G}$, $a, b \in \mathbb{Z}$ we have that $e(x^a, y^b) = e(x, y)^{ab}$.*

Non-Degenerate: *For all $x \in \mathbb{G}$, $x \neq 1$, $e(x, x)$ generates $\mathbb{G}_1$.*

Computable: *For all $x, y \in \mathbb{G}$, the pairing $e(x, y)$ can be computed efficiently.*

There are several computational assumptions in the world of pairing-based cryptography. In this paper we will reduce the security of our protocol to the following:

Definition 2 (Decisional Linear (DLIN) Assumption [BBS04]). *Let $\mathbb{G}, \mathbb{G}_1$ be groups of prime order p with a bilinear map e as defined above. The decisional linear assumption states that given three random generators f, h, g and f^r, h^s, g^t, it is hard to distinguish the case $t = r + s$ from a random t.*

3.2 Universally Composable Security Framework

If we want to prove that a protocol is secure, we first need to define what secure means. The universally composable security framework, defined by Canetti

[5] In fact, pairings were used even before in cryptography, in order to break the discrete logarithm problem (the MOV attack [MOV93]).

[Can01], is becoming a standard definition if one wants proper security guarantees. The strength of this framework relies in the universally composable theorem, which states that if a protocol is secure in the UC model, then this protocol will preserve the same security even if composed with an arbitrary number of copies of itself or with other protocols.

The price to pay for such a result is the impossibility of constructing any nontrivial protocol that is secure in the UC model[6]. In order to develop interesting protocols in the UC model we need some kind of setup assumptions. We put ourselves in the key registration authority scenario, first introduced in [BCNP04]. In this model, that has the flavor of a public-key infrastructure, we assume that there exists a trusted registration authority where parties can register public keys associated with their identities, while demonstrating that they have access to the corresponding secret keys. Alternatively, parties can let the authority choose public keys for them, in scenarios where the corresponding secret keys need not be revealed, even to the owners of the public keys. Then, parties can query the authority for a party identity and obtain the registered public key for that identity. Any ideal functionality can be UC realized by interactive protocols in the KR model under standard computational hardness assumptions.

An advantage of the KR assumption, in respect to other setup assumptions like the common reference string model, is that it is trivial to ensure that all trust is not concentrated in one single entity. Namely, the receiver can register his key to several KRs, and the sender retrieves it from all of them. Now the sender only has to trust that one of the KRs does its job properly to be convinced that the receiver knows its secret key. A full comparison of the KR model against the CRS model is out of the scope of this paper, and we refer to [BCNP04] for more details.

There are several ways to implement a KR in the real world. In particular, as discussed in [BCNP04], it is possible to implement it with a stand-alone zero-knowledge proof of knowledge, if we have the guarantee that the proofs are run in a isolated trusted environment — maybe the registrant shows up at the KR with the prover on a smartcard, and then the KR runs the smartcard in an isolated setting. If perfect isolation is not available, it is still possible to run an UC setup in the case of partial isolation [DNW08], where the parties are allowed to communicate with the environment, but just a limited amount of data.

We note that our protocol has *gracefully degradation*, as defined in [BCNP04]: if the proof of secret key given by the receiver is not UC (maybe because the assumption that the proof was running in an isolated setting failed), but it is at least a stand-alone proof of knowledge, then our OT protocol too will be a stand-alone secure implementation (of the multi-OT functionality) — the simulator will rewind the stand-alone proof of knowledge from the receiver to get the secret key, and then proceed as the UC simulator for all the OTs. Even if the proof fails to be just a stand-alone proof of knowledge, but it is at least a proof of membership, we will have some security, as the key being well-formed gives unconditional security for the sender, though without any composition guarantees. The implementation

[6] Actually, it is possible to implement symmetric protocols like secure channels [CK02].

therefore in some sense delivers the best possible security level given the quality of the trusted setup.

4 Underlying Primitives

4.1 Mixed Commitment Scheme

We use a special kind of commitment called mixed commitment [DN02], which is a commitment scheme that can be instantiated with two kinds of key, giving two kinds of security. The first kind is perfectly binding and extractable, while the second is perfectly hiding and equivocal. A key which produces perfectly binding commitments will be called an extraction key (X-key) while a key that produces perfectly hiding commitments will be called an equivocal key (E-key). These two kinds of keys have to be computationally indistinguishable, in order for the commitment scheme to be called mixed.

We note that if we always instantiate the commitment scheme with X-keys, we end up with a commitment scheme that allows extraction, i.e. a public-key encryption scheme, where some keys (the E-keys) ensure that the "ciphertexts" contain no information about the plaintexts. We use this as an essential ingredient in our construction.

DLIN based Encryption Scheme. When we build our mixed commitment scheme, we start from the DLIN based cryptosystem from Boneh, Boyen and Shacham [BBS04], described now.

Let G be an algorithm that takes a security parameter as input and outputs $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ such that p is prime, $\mathbb{G}, \mathbb{G}_1$ are descriptions of groups of order p, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is an admissible bilinear map, and g is a generator of $\mathbb{G}$. Those are the public parameters of the cryptosystem that works as follows:

Key Generation: Select x, y randomly in $\mathbb{Z}_p^*$, then compute $(f, h) = (g^x, g^y)$. The encryption key is $ek = (f, h)$ and the decryption key is $dk = (x, y)$.
Encryption: To encrypt a message $M \in \mathbb{G}$, select two random values $r, s \in \mathbb{Z}_p^*$. Then compute the encryption as $E_{ek}(M; r, s) = (\alpha, \beta, \gamma) = (f^r, h^s, g^{r+s}M)$.
Decryption: The message can be efficiently decrypted as $D_{dk}(\alpha, \beta, \gamma) = M = \alpha^{-1/x}\beta^{-1/y}\gamma$.

This encryption scheme is clearly IND-CPA secure under the DLIN assumption. Note that until now we did not use the pairing at all. The pairing will be used to prove statements about encryptions.

Mixed Commitment Scheme. We now describe the mixed commitment scheme. This is the commitment scheme under which the sender commits to x_0 and x_1 (under an E-key), and when instantiated with an X-key, it is the encryption scheme used by the receiver to encrypt b. This is an essential trick, as it will make commitments and encryptions work together nicely.

The keys for the commitment scheme are going to be the ciphertexts of DLIN cryptosystem. I.e., a commitment key is of the form $ck = E_{ek}(M; r, s)$. A commitment to $m \in \mathbb{Z}_p$ is then of the form

$$\mathrm{Comm}_{ek,ck}(m;t,u) = ck^m E_{ek}(1;t,u)\ .$$

This is clearly homomorphic in the sense that

$$E_{ek}(M_0;t_0,u_0)E_{ek}(M_1;t_1,u_1) = E_{ek}(M_0 M_1;t_0+t_1,u_0+u_1)\ .$$

The basic idea of the mixed commitment scheme is then that if ck is an encryption of 1, then it follows that $ck^m E_{ek}(1;t,u)$ is a random encryption of 1. And, it is possible to efficiently open this commitment to any m given t and u and the randomness used to compute ck. On the other hand, if ck is the encryption of a generator (say g), then

$$ck^m E_{ek}(1;t,u)$$

is a random encryption of g^m, and it is therefore perfectly binding. In addition, it is possible to extract g^m from the commitment if the decryption key dk is known. Thanks to the properties of the cryptosystem, it is computationally infeasible to decide whether ck is an encryption of 1 or g, which is why it is a mixed commitment scheme.

This leads to the construction of our scheme as follows:

General key generation: There is a key pair (ek, dk), where $ek = (g, f, h) = (g, g^x, g^y)$ is an encryption key and $dk = (x, y)$ the decryption key. A full public key for the system is of the form (ek, ck), where ck is an encryption under ek.

Extraction key (X-key): For an X commitment key we have $ck = ck_X = E_{ek}(g; r, s) = (f^r, h^s, g^{r+s+1})$, where r and s are random. We will denote by $ck_X \leftarrow \mathrm{KG}_X$ the algorithm that produces an X-key, and we use $\mathcal{K}_X$ to denote the set of X-keys. The extraction trapdoor is $t_X = dk$.

Equivocal key (E-key): For an E commitment key we have $ck = ck_E = E_{ek}(1; r, s) = (f^r, h^s, g^{r+s})$, where r and s are random and $t_E = (r, s)$ is the equivocation trapdoor. We will denote by $ck_E \leftarrow \mathrm{KG}_E(r, s)$ the algorithm that produces an E-key, and we use $\mathcal{K}_E$ to denote the set of E-keys.

Committing: To commit to a message $m \in \mathbb{Z}_p$ under the general key ek and the commitment key ck, select $t, u \in_R \mathbb{Z}_p^*$, and compute $\mathrm{Comm}_{ek,ck}(m; t, u) = ck^m E_{ek}(1; t, u)$.

Opening: To open a commitment C, the committer releases (m, t, u). The receiver checks that $C = ck^m E_{ek}(1; t, u)$.

The general secret key dk allows to efficiently determine if the commitment key ck is an X-key or an E-key, otherwise, they will be indistinguishable from the properties of DLIN cryptosystem. Note that this commitment is homomorphic with respect to addition.

It is clear that the E-keys produce perfectly hiding commitments and that the X-keys produce perfectly binding commitments. Here is how equivocation and extraction work:

Equivocation: A random commitment to m under an E-key $ck \in \mathcal{K}_E$ is of the form

$$\begin{aligned}\mathrm{Comm}_{ek,ck}(m;t,u) &= ck^m E_{ek}(1;t,u)\\ &= (f^r, h^s, g^{r+s})^m (f^t, h^u, g^{t+u}) = (f^{rm+t}, h^{sm+u}, g^{rm+sm+t+u})\end{aligned}$$

for uniformly random t and u. Given any m' and letting $t' = r(m-m')+t$ and $u' = s(m-m')+u$, it follows that t' and u' are uniformly random and that

$$\begin{aligned}\mathrm{Comm}_{ek,ck}(m';t',u') &= (f^{rm'+t'}, h^{sm'+u'}, g^{rm'+sm'+t'+u'})\\ &= (f^{rm+t}, h^{sm+u}, g^{rm+sm+t+u}) = \mathrm{Comm}_{ek,ck}(m;t,u)\ .\end{aligned}$$

I.e., given the randomness used to compute $\mathrm{Comm}_{ek,ck}(m;t,u)$ and the equivocation trapdoor $t_E = (r,s)$ of ck, one can open $\mathrm{Comm}_{ek,ck}(m;t,u)$ to any value.

Extraction: For $ck \in \mathcal{K}_X$ and $c = \mathrm{Comm}_{ek,ck}(m;t,u) = ck^m E_{ek}(1;t,u)$ we have that $D_{dk}(c) = D_{dk}(ck)^m \cdot D_{dk}(E_{ek}(1;t,u)) = g^m$, from which m can be retrieved by exhaustive searching, if it is from a small known set.

Note that the extraction has a limit on the size of m that can be extracted. In our first construction, our message set will be just $\{0,1\}$ when we need extraction of m. In our second construction we consider $K = g^m$ to be a key and m just a value used to generate the key. In that case we can extract the key K for any m.

4.2 Efficient NIWI Proofs

Both during the Choose and the Transfer phase we need some non-interactive (NI) proofs. It turns out that these proofs do not have to be fully zero-knowledge. It is sufficient that they are witness indistinguishable (WI). Still, using standard WI proofs will result in increasing dramatically the number of rounds of the protocol. But also general NIWI proofs, even without increasing the number of rounds, will let the protocol be impractical. We use instead new NIWI constructions [GOS06, GS08] which allow to prove algebraic relations in bilinear groups. In particular, we use the following WI proofs:

Proof of Bit: In [GOS06] a composable NIWI to prove that the content of a commitment is either 0 or 1 is given. We will denote with $\pi_{0\vee 1}(c)$ the proof for the following relations: $R_{bit} = \{((ek,ck,c),(m,t,u)) | c = \mathrm{Comm}_{ek,ck}(m;t,u) \wedge m \in \{0,1\}\}$. The proof consists of 6 group elements.

Proof of Multi-Exponent: In [GS08] a composable NIWI to prove the relation between the content of a number of commitments and the exponents of a multi-exponentiation is given. The proof consists of 2 group elements. We use it for 3 exponents and denote with $\pi_{MX}(c, g_1, g_2, g_3, C_1, C_2, C_3)$ the proof for the following relations:

$$\begin{aligned}R_{MX} = \{((ek,ck,c,g_1,g_2,g_3,C_1,C_2,C_3),(x_1,x_2,x_3,t_1,t_2,t_3,u_1,u_2,u_3))|\\ c = g_1^{x_1} g_2^{x_2} g_3^{x_3}, \forall i = 1,2,3 : C_i = \mathrm{Comm}_{ek,ck}(x_i;t_i,u_i)\}\ .\end{aligned}$$

All the proofs are for the CRS model, where the proof assumes that a random common reference string crs has been honestly generated and it is known by the prover and the verifier. More precisely crs is sampled as $crs \leftarrow \mathrm{CRS}(r_{crs})$ for a poly-time algorithm CRS and uniformly random r_{crs}. In fact, there exist two different such generators CRS_S and CRS_Z. When crs is generated by CRS_S, then the proofs have *perfect* soundness and computational WI (under DLIN). When crs is generated by CRS_Z, then the proofs are *perfect* WI and computationally sound (under DLIN). The outputs of the two generators are in addition computationally indistinguishable. We can exploit this to avoid the CRS model at all. We let the receiver generate two common reference strings $crs_S = \mathrm{CRS}_S(r_{crs,S})$ and $crs_Z = \mathrm{CRS}_Z(r_{crs,Z})$ and use the KR authority to verify that crs_S and crs_Z were generated in this way. Then the proofs from the sender to the receiver are done under crs_Z, and the sender is guaranteed WI even if crs_Z was not generated at random, as the WI is perfect. Proofs from the receiver to the sender are done under crs_S, and the sender is guaranteed soundness even if crs_S was not generated at random, as the soundness is perfect.

5 Final Protocol

Since the mixed commitment scheme that we use does not have efficient extraction for arbitrary messages, we at first use it to construct *bit* OT. Later, we are discussing how to achieve *string* OT.

5.1 Parameter Agreement

We assume that all parties agree on the finite groups underlying the encryption scheme. In practice this would probably happen by the groups being described in some standard and hard-coded into the software for the OT module. In the UC model we model it using an ideal functionality which simply outputs a description of the groups to all parties. This ideal functionality can be thought of as the standardization body, and be activated with a message (`get groups`). It generates a DLIN group by running $(p, \mathbb{G}, \mathbb{G}_1, e, g) \leftarrow G(1^k)$, and outputs $param = (p, \mathbb{G}, \mathbb{G}_1, e, g)$ to all parties.

5.2 Key Registration Authority

Next we describe the registration phase. In the registration phase all parties which later want to act as receivers have to register a public key with a KR authority and prove knowledge of the corresponding secret key. Later all parties which want to act as senders can retrieve the public keys of the receivers from KR. Following [BCNP04] we model this simplistically by having one KR, by letting the registrants show knowledge of their secret keys by showing them directly to the KR and letting the KR broadcast the corresponding public keys.

In more detail, the KR is parametrized by some poly-time relation R and accepts messages of the form $(\texttt{register}, pid, pk, sk)$ from some party P_i. It checks that $(pk, sk) \in R$ and if so sends (P_i, pk) to all parties. The relation R is chosen such that (pk, sk) being in R ensures that pk is a well-formed public key and that sk is the secret key.

In our protocol we use a public key of the form $pk = (param, ek, ck_X, ck_E, crs_Z, crs_S)$, where $param = (p, \mathbb{G}, \mathbb{G}_1, e, g)$ and $ek = (f, h)$, and we use a secret key of the form $sk = (dk, r_X, t_E, r_{crs,Z}, r_{crs,S})$, where $dk = (x, y)$. The relation R checks that $f = g^x$, $h = g^y$, $ck_X = \mathrm{KG}_X(r_X)$, $ck_E = \mathrm{KG}_E(t_E)$, $crs_Z = \mathrm{CRS_Z}(r_{crs,Z})$ and $crs_S = \mathrm{CRS_S}(r_{crs,S})$. I.e., it checks that ek is a well-formed public key for DLIN cryptosystem (and that the receiver knows the decryption key) and that ck_X is an X-key for our mixed commitment scheme and that ck_E is an E-key for our mixed commitment scheme (and that the receiver knows the equivocation trapdoor) and that crs_Z is a well-formed common reference string for the NIWI system giving perfect WI (and that the receiver knows how to simulate proofs), and that crs_S is a well-formed common reference string for the NIWI system giving perfect soundness.

The receiver P_i lets $param$ be the public parameters agreed upon by all parties, and he generates ek, ck_X, ck_E, crs_Z and crs_S at random, thereby learning the sk expected by KR. After key registration the receiver deletes r_X, t_E, $r_{crs,S}$ and $r_{crs,Z}$, as they are not needed in the protocol, and constitute a security risk if leaked. When the sender receives $(P_i, (param', ek, ck_X, ck_E, crs_Z, crs_S))$ he checks that $param'$ is equal to the parameters $param$ agreed upon earlier. If so, he remembers that the public key of receiver P_i is $(ek, ck_X, ck_E, crs_Z, crs_S)$.

5.3 1-Out-of-2 Bit Oblivious Transfer

We now describe the communication phase. Here the parties can perform an unbounded number of OTs using the established PKI.

Choose: The receiver is given a bit b and an OT identifier $otid$. He computes a commitment $c_1 = (\alpha_1, \beta_1, \gamma_1) = \mathrm{Comm}_{ek,ck_X}(b)$. He sends $otid, c_1$ to the sender. He also sends $\pi_{0\vee 1}(c_1)$, computed under crs_S.

Transfer: The sender is given two secrets x_0, x_1 and $otid$. He waits for a message of the form $otid, c_1, \pi_{0\vee 1}$ from the receiver and checks the receiver's proofs and, if he accepts, he computes and sends to the receiver $otid, d = c_0^{x_0} c_1^{x_1} E_{ek}(1; r, s)$ with r and s chosen at random and with $c_0 = (\alpha_0, \beta_0, \gamma_0) = E_{ek}(1)c_1^{-1}$.[7] Note that when the sender is honest, then $d = (\alpha, \beta, \gamma) = (\alpha_0^{x_0}\alpha_1^{x_1} f^r, \beta_0^{x_0}\beta_1^{x_1} h^s, \gamma_0^{x_0}\gamma_1^{x_1} g^{r+s})$. In addition, the sender sends commitments $C_0 \leftarrow \mathrm{Comm}_{ek,ck_E}(x_0)$, $C_1 \leftarrow \mathrm{Comm}_{ek,ck_E}(x_1)$, $C_2 \leftarrow \mathrm{Comm}_{ek,ck_E}(r)$, $C_3 \leftarrow \mathrm{Comm}_{ek,ck_E}(s)$, and proofs $\pi_{MX}(\alpha, \alpha_0, \alpha_1, f, C_0, C_1, C_2)$, $\pi_{MX}(\beta, \beta_0, \beta_1, h, C_0, C_1, C_3)$ and $\pi_{MX}(\gamma, \gamma_0, \gamma_1, g, C_0, C_1, C_2C_3)$, to prove that C_0, C_1, C_2, C_3 commit to values x_0, x_1, r, s used to compute d. The NIWI proofs are performed under crs_Z.

Retrieve: The receiver checks the proofs, and, if he accepts, he extracts d using dk and obtains $g^{x_b} = D_{dk}(d)$. If $g^{x_b} = 1$ he outputs 0, otherwise he outputs 1.

[7] Here $E_{ek}(1)$ is some fixed encryption of 1 so that also R can compute c_0. This just saves the sending of c_0.

The protocol sends the 6 commitments $c_1, d, C_0, C_1, C_2, C_3$, each consisting of 3 group elements. Besides this a proof of size 6 is sent and 3 proofs of size 2 are sent, for a total of 30 group elements.

Theorem 1. *Under the assumption that the DLIN problem is hard, the protocol in Section 5.3 securely realizes F_{OT} in the UC-hybrid model for static corruptions.*

Proof: As we deal with static security, the adversary needs to choose which party to corrupt before the protocol starts. We can therefore address the case of the malicious sender and the malicious receiver separately.

Security Against Malicious Receiver: The simulator runs the KR phase. If any of the keys registered by a corrupted receiver are not well formed, it ignores the registration, as would the real KR. Otherwise, it extracts from the receiver the secret key $(dk, r_X, t_E, r_{crs,Z}, r_{crs,S})$.

In the communication phase, when it receives a message $otid, c_1, \pi_{0 \vee 1}$ from a corrupted receiver, it checks for the proof to be valid. If yes, it extracts $g^b = D_{dk}(c_1)$. If $g^b = 1$, it inputs 0 to the ideal functionality on behalf of the corrupted receiver. Otherwise it inputs 1. If the proofs fail, the simulator just ignores the message.

When the simulator is given x_b by the ideal functionality it behaves like an honest sender, after choosing a random x'_{1-b}.

This transfer message is perfectly indistinguishable from a protocol one. In fact d will be exactly the same compared to a real protocol run, being an encryption of the same message x_b. All the commitments C_0, C_1, C_2, C_3 have the same distribution, as ck_E is an E-key. The three proofs sent by the sender have the same distribution, as the proofs are perfect WI when performed under crs_Z — in fact the receiver could have simulated the proofs himself. Note that no complexity assumption is required here, so the simulation is perfect against a malicious receiver.

Security Against Malicious Sender: The simulator runs the KR phase. For all honest receivers it uses a random key $(ek, ck_X, ck_E, crs_Z, crs_S)$, where ck_X is a random E-key, ck_E is a random X-key, crs_Z is generated using CRS_S, and crs_S is generated using CRS_Z. These four are indistinguishable for the corrupted sender as E-keys and X-keys are indistinguishable and the output distributions of CRS_Z and CRS_S are indistinguishable if DLIN is hard.

In the communication phase, when it has to send a Choose message for some $otid$ from the honest receiver to a corrupted sender, it does not know the real choice bit b. Instead it uses $b' = 0$.

When it receives $otid, d, C_0, C_1, C_2, C_3$ from the corrupted sender, along with the three proofs, it checks the proofs as in the protocol. If the proofs are valid, it extracts $g^{x_0} = D_{dk}(C_1)$ and $g^{x_1} = D_{dk}(C_2)$, and recovers x_0, x_1 by letting $x_i = 0$ if $g^{x_i} = 1$ and $x_i = 1$ otherwise, with $i = 0, 1$. Then it inputs them to the ideal functionality on behalf of the corrupted sender. Otherwise, it just ignores the message.

The Hybrid Argument. The above just sketches the simulator. Showing that the simulation with this simulator is indistinguishable from the real execution is as usual done using a hybrid argument.

That the simulation for a malicious receiver is perfect was already argued above. We therefore focus on the case of a malicious sender. We here sketch the sequence of hybrids used to go from the simulation to the real protocol.

In the first step the only change is to use $b' = b$ instead of $b' = 0$. Here b is the real choice bit, which is obtained by inspecting the ideal functionality — this the simulator cannot do, but we can do it as a mind spiel to define a hybrid distribution. Note that this step actually does not change the distribution as ck_X is an E-key in the simulation and the distribution of c_1 therefore does not depend on b' at all — it can be opened to both 0 and 1. Furthermore, the proof is perfect WI when crs_S is generated using CRS_Z, and the distribution of the proof therefore does not depend on which opening of c_1 is used.

In the second step we change ck_X to be a random X-key instead of a random E-key and generate crs_S using CRS_S. This change is indistinguishable to the corrupted parties as dk is kept secret by the honest receiver and the output distributions of CRS_S and CRS_Z are computationally indistinguishable if DLIN is hard. Note that after these two steps, the messages $c_1, \pi_{0 \vee 1}$ have the same distribution as in the real protocol.

In the third step, the simulator computes the output x_b of the receiver, not by extracting x_0 and x_1 from C_1 and C_2 and inputting (x_0, x_1) to the ideal functionality (to make it output x_b), but by extracting a value x'_b from d and letting the ideal functionality output x'_b. When both ck_E and ck_X are X-keys, as they are at this point in the sequence of hybrids, it is straightforward to verify that if all three statements proved by the sender are true, it will always hold that $x'_b = x_b$. Since the NIWI proof system has perfect soundness when crs_Z is generated using CRS_S, as it is at this point in the sequence of hybrids, it follows that the third step actually makes no difference in the distribution.

In the last step ck_E is changed to be a random E-key instead of a random X-key and crs_Z is generated using CRS_Z. Again this is indistinguishable. Now the distribution is identical to the real protocol.

5.4 1-Out-of-2 String Oblivious Transfer

In this section we present a protocol for string OT that is more efficient than achieving string OT by standard composition of bit OT.

The reason why we cannot just OT an n-bit value is that the receiver will not be able to decrypt the answer anymore. Recall that the value d sent from S to R is a commitment of the form $\mathrm{Comm}_{ek,ck_X}(x_b)$. This allows the receiver to efficiently compute g^{x_b} from d using the decryption key dk, as d is a commitment under the X-key ck_X. The problem is that if we let x_b be arbitrary, then computing x_b from g^{x_b} cannot be done efficiently.

Our idea is to consider our protocol a random OT, where the sender inputs two random values x_0, x_1, and the receiver gets a random group element $K_b = g^{x_b}$. Let us note that the sender cannot choose the values of K_0 and K_1, as the discrete logarithm problem is assumed to be hard here. However, he can compute himself the two elements K_0, K_1, as he knows g, x_0, x_1. At the end, S and R share a random group element, that the sender can use as a key to encrypt his messages M_0, M_1, encoded as group elements, i.e. the sender computes and sends $X_0 = g^{x_0}M_0, X_1 = g^{x_1}M_1$ to the receiver. The receiver can retrieve $M_b = X_b K_b^{-1}$. Since K_{1-b} is a uniformly random group element completely unknown by the receiver, he gets no information on M_{1-b}. The only price paid is that we send two more group elements, increasing the total communication up to 32 group elements.

Though intuitively clear, it remains to verify that the security is maintained. It turns out that the only non-trivial part is to check that the simulator can still extract the commitments from a malicious S — recall that ck_X is an E-key in the simulation and that the simulator therefore cannot extract d, but must extract the commitments given by the malicious S. We focus on this part, and leave the easier details to the reader.

Recall that e.g. the first component of d, being $\alpha_0^{x_0}\alpha_1^{x_1}\alpha^r$, is accompanied by three commitments $C_0 \leftarrow \mathrm{Comm}_{ek,ck_E}(x_0), C_1 \leftarrow \mathrm{Comm}_{ek,ck_E}(x_1), C_2 \leftarrow \mathrm{Comm}_{ek,ck_E}(r)$ and a proof that they commit to values used to compute the first component of d. Recall that in the simulation ck_E is an X-key. This means that the simulator can use dk to extract g^{x_0} from C_0 and g^{x_1} from C_1, which allows to compute $K_0 = g^{x_0}$ and $K_1 = g^{x_1}$, as desired. The proofs therefore, so to say, do not prove that S knows x_0 and x_1, but that S knows K_0 and K_1, which is sufficient to ensure that he knows M_0 and M_1.

Acknowledgments

We thank the anonymous reviewers from CRYPTO 2008 for the useful comments.

References

[AIR01] Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)

[BBS04] Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

[BCNP04] Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS, pp. 186–195. IEEE Computer Society, Los Alamitos (2004)

[BF01] Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)

[Can01] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)

[CF01] Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)

[CK02] Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)

[CNS07] Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor [Nao07], pp. 573–590

[DN02] Damgård, I., Nielsen, J.B.: Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (2002)

[DNW08] Damgård, I., Nielsen, J.B., Wichs, D.: Isolated proofs of knowledge and isolated zero knowledge. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 509–526. Springer, Heidelberg (2008)

[Fis06] Fischlin, M.: Universally composable oblivious transfer in the multi-party setting. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 332–349. Springer, Heidelberg (2006)

[Gar04] Garay, J.A.: Efficient and universally composable committed oblivious transfer and applications. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 297–316. Springer, Heidelberg (2004)

[GH07] Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)

[GH08] Green, M., Hohenberger, S.: Universally composable adaptive oblivious transfer. In: Pieprzyk, Y. (ed.) ASIACRYPT 2008. LNCS vol. 5350, pp. 179–197. Springer, Heidelberg (2008)

[GMW87] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229. ACM, New York (1987)

[GOS06] 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)

[GS08] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008), http://eprint.iacr.org/2007/155

[Jou00] 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)

[JS07] Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: Naor [Nao07], pp. 97–114

[Kal05] Kalai, Y.T.: Smooth projective hashing and two-message oblivious transfer. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 78–95. Springer, Heidelberg (2005)

[Lin08] Lindell, A.Y.: Efficient fully-simulatable oblivious transfer. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 52–70. Springer, Heidelberg (2008), http://eprint.iacr.org/2008/035

[MOV93] Menezes, A., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory 39(5), 1639–1646 (1993)

[Nao07] Naor, M. (ed.): EUROCRYPT 2007. LNCS, vol. 4515. Springer, Heidelberg (2007)

[NP01] Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: SODA, pp. 448–457 (2001)

[NP05] Naor, M., Pinkas, B.: Computationally secure oblivious transfer. J. Cryptology 18(1), 1–35 (2005)

[PVW07] Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. Cryptology ePrint Archive, Report 2007/348 (2007), `http://eprint.iacr.org/`

[Rab81] Rabin, M.O.: How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory (1981)

[Wie83] Wiesner, S.: Conjugate coding. SIGACT News 15(1), 78–88 (1983)

[Yao86] Yao, A.C.-C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167. IEEE, Los Alamitos (1986)

Generalized Universal Circuits for Secure Evaluation of Private Functions with Application to Data Classification

Ahmad-Reza Sadeghi[*] and Thomas Schneider[**]

Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Germany
{ahmad.sadeghi,thomas.schneider}@trust.rub.de

Abstract. Secure Evaluation of Private Functions (PF-SFE) allows two parties to compute a private function which is known by one party only on private data of both. It is known that PF-SFE can be reduced to Secure Function Evaluation (SFE) of a Universal Circuit (UC). Previous UC constructions only simulated circuits with gates of $d = 2$ inputs while gates with $d > 2$ inputs were decomposed into many gates with 2 inputs which is inefficient for large d as the size of UC heavily depends on the number of gates.

We present generalized UC constructions to efficiently simulate any circuit with gates of $d \geq 2$ inputs having efficient circuit representation. Our constructions are non-trivial generalizations of previously known UC constructions.

As application we show how to securely evaluate private functions such as neural networks (NN) which are increasingly used in commercial applications. Our provably secure PF-SFE protocol needs only one round in the semi-honest model (or even no online communication at all using non-interactive oblivious transfer) and evaluates a generalized UC that entirely hides the structure of the private NN. This enables applications like privacy-preserving data classification based on private NNs without trusted third party while simultaneously protecting user's data and NN owner's intellectual property.

Keywords: universal circuits, secure evaluation of private functions, neural networks, private data classification, privacy.

1 Introduction

Today, a variety of new business models can be provided as electronic services where customers post their requests to a remote provider who performs specific knowledge based operations on their data and provides customers with the results. Examples are expert systems for heath diagnostics, remote data bases, multimedia data processing, or data classification tools (e.g., for spam). From

[*] The first author was supported by the European Union under FP6 project SPEED.
[**] The second author was supported by the European Union under FP7 project CACE.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 336–353, 2009.

security targets point of view customers may send sensitive and security critical data and hence require the protection (confidentiality and integrity) of their data, while the service providers may require the protection of their Intellectual Property (IP), i.e., their expertise embedded in their system.

Hence, the problem can be formulated as follows: two parties, a service requester R (client) and a service provider P (server), are involved in the computation of a function f (belonging to P) on data x (input by R) where P should not obtain any information about x and R should not get any useful information about f besides the result $f(x)$[1].

For arbitrary functions f, this is tackled by *Secure Function Evaluation* (SFE) of *Private Functions* (PF-SFE). SFE protocols [27,12,14,13,9,5] allow two parties to securely evaluate a common function on private data. Based on this, PF-SFE [20,16,10] evaluates a *Universal Circuit* (UC) [25,10] as common function which is programmed with the private function f. As UC can be programmed to simulate any function, it entirely hides f while SFE ensures privacy of data x.

Previous UC constructions can simulate circuits with gates of $d = 2$ inputs only. For example, circuits performing arithmetic operations like addition, number comparison, or multiplication are most efficiently implemented from chains of $d = 3$ input gates (full adders, and full comparers). When these types of circuits need to be simulated with known UC constructions (to hide which arithmetic operations are performed), the large gates must be decomposed into many $d = 2$ input gates, e.g., five gates per 3-input gate using Shannon's expansion theorem: $f(a, b, c) = (c \wedge f(a, b)|_{c=1}) \vee (\bar{c} \wedge f(a, b)|_{c=0}))$.

To overcome this overhead, we generalize previous UC constructions to $d \geq 2$ as non-trivial extensions of previous work together with new constructions that are especially suited to simulate small circuits. Our constructions are much more efficient than using the straight-forward solution of evaluating $\lceil d/2 \rceil$ known UCs (where $d = 2$) in parallel. The overhead of our best Generalized UC (GUC) construction is by a factor of two smaller than the best known UC construction for the practical example given in §5.4.

As application of our GUC constructions we show how to securely evaluate private Neural Networks (NN) where each neuron has d inputs. Amongst others, NNs are very useful tools for data classification including pattern/sequence recognition, and sequential decision making. NNs are increasingly becoming important for deployment in commercial applications like spam filtering [3], speech recognition [24], and many more [23] where the "expertise" of the provider is embedded in NN. Neural networks allow to model any function [7], and are robust against noise. Previous work [15,17] is based on straight forward use of homomorphic encryption in multiple rounds that cannot hide NN's structure completely and is not provably secure. In contrast we show that our solution (i) is efficient w.r.t. the size of the circuit needed for a reasonable NN, (ii) requires no or only one round to evaluate a GUC that (iii) hides the underlying NN and its topology entirely and (iv) is provably secure in the semi-honest model.

[1] Clearly, evaluating f on different inputs x_i allows R to learn some information on f. Thus, P should restrict the maximum number of evaluations of f by other means.

1.1 Related Work

Two-Party Secure Function Evaluation (SFE) protocols [27,12,9] allow two parties to evaluate any function represented as boolean circuit which is known to both of them on their respective private inputs. All of these protocols are provably secure against semi-honest adversaries and can be extended to be secure against malicious adversaries via cut-and-choose [14,13,5]. SFE in semi-honest model is based on Oblivious Transfer (OT) requiring one round of communication [19,1,8] or a non-interactive implementation based on an extension of trusted computing modules [6].

NNs can be represented as boolean circuits - threshold NNs with backpropagation [21] or evolutionary learning algorithms [18] were implemented in hardware, but were not considered to be evaluated within SFE protocols yet.

SFE can be extended to private functions (PF-SFE) where the private function is known to one party only and hidden entirely from the other party. PF-SFE is reduced to SFE of a Universal Circuit (UC) that is programmed with the private function and entirely hides its structure [20,16,10]. The SFE protocol of [9] which allows very efficient evaluation of UCs can also be used to improve evaluation of GUCs of this paper (by a factor between two- and fourfold).

Currently known UC constructions [25,10] can simulate gates with $d = 2$ inputs only. Gates with more than two inputs can be simulated by decomposing them into multiple gates with two inputs.

Oblivious training and evaluation of NNs was studied in the context of Oblivious Polynomial Evaluation (OPE) [2]. The OPE protocol reduces oblivious polynomial evaluation to OT. Based on this protocol, they show how to train and evaluate NNs in multiple rounds without hiding the structure of the NN. Nonlinear activation functions are either evaluated using a circuit based SFE protocol or piece-wise approximated as polynomials evaluated via OPE protocol.

Oblivious NN evaluation using homomorphic encryption [15,17] requires multiple rounds as well - one per layer of the NN. The protocol allows evaluation of NNs with several activation functions like threshold or sigmoid function. To hide the structure of the NN, dummy neurons are introduced but this does not entirely hide the structure (e.g., maximal number of neurons per layer and maximal number of layers are revealed as outlined in their paper). Also, these protocols are not provably secure as blinding an additively homomorphic encrypted value with a randomly chosen factor reveals information on the magnitude of the value.

1.2 Our Contributions

In §4 we present practical *Generalized Universal Circuit (GUC) constructions* to efficiently simulate circuits of gates with $d \geq 2$ inputs having efficient circuit representations. Former UCs are restricted to $d = 2$ and decompose larger gates into multiple gates with two inputs resulting in much more overhead than our GUC constructions (cf. Table 1 in §5.4). Our constructions are non-trivial generalizations of known UC constructions that are special cases of ours for $d = 2$.

Based on these GUC constructions we present *protocols to securely evaluate private Neural Networks* (NN) with activation functions implemented as small circuits (such as threshold function) in §5. Unlike previous work, our protocols are provably secure, need a constant number of rounds (one round in the semi-honest model or even no online communication at all using non-interactive OT), hide NN's structure entirely (besides number of in- and outputs, maximum degree d and number of neurons k) and are still practical.

1.3 Basic Idea and Outline

A *Generalized Universal Circuit* (GUC) is a circuit which can be programmed to simulate an arbitrary circuit that consists of gates with d inputs each. These gates are required to have an efficient circuit representation which is the case in our example for neurons in §5 or if the size of their function table 2^d is small.

A GUC can be thought of as a kind of processor (here: programmable circuit) that takes as input some data x and a program p_f corresponding to a function (here: input circuit) and evaluates the program on the data (here: input circuit on data): $UC(p_f, x) = f(x)$. As GUC can be programmed with *any* function it does not reveal anything about the function. This allows to evaluate arbitrary functions privately. In contrast to previous UC constructions we relax the restriction that input circuits need to consist of gates of $d = 2$ inputs only.

In §4 we give different methods to construct GUCs in an iterative (§4.1), modular (§4.2), or graph based (§4.3) way and compare them in §4.4. Definitions are given in §2, new building blocks in §3.

As practical application where simulation of d input gates is advantageous, we show how neural networks (NN), described in §5.1, can be expressed as circuits consisting of d input gates in §5.2. Their structure can be entirely hidden inside a GUC which allows secure evaluation of private NNs as shown in §5.3. Finally, we compare our GUC construction and its application to securely evaluate private NNs to previously known constructions and protocols in §5.4.

2 Definitions and Preliminaries

The following definitions generalize those of [10] to gates with multiple inputs.

A *gate* G is the implementation of a boolean function $\{0,1\}^d \to \{0,1\}$ with d *inputs* and one *output*. The *size* of a gate G, denoted by $|G|$, is the multiple of function table entries needed to implement the gate w.r.t. a 2 input gate, namely $|G| = 2^{d-2}$ (e.g., $|B|_{d=1} = 0.5$, $|B|_{d=2} = 1$, etc.).

We consider acyclic *circuits* consisting of connected gates with arbitrary fan-out, i.e., the output of each gate can be used as input to arbitrary many gates. Further, each output of circuit C is the output of a gate and not a redirected input of C. The *size* of a circuit is the sum of the sizes of its gates. Communication and computation complexity of SFE protocols is linear in the size of the circuit.

A *programmable gate* is a gate with an unspecified function table. To *program* it, a specific function table with 2^d entries for each input combination is given.

To simplify presentation we group gates into functional *blocks* as follows:

A *block* B^u_v is a sub-circuit with u inputs $in_1, .., in_u$ and v outputs $out_1, .., out_v$. B^u_v computes a function $f_B : \{0,1\}^u \to \{0,1\}^v$ that maps the input values to the output values. For simplicity, we identify B_v^u with f_B and write: $B(in_1, .., in_u) = (out_1, .., out_v)$. Blocks consist of connected gates and other sub-blocks. The size of block B, denoted by $|B|$, is the sum of the sizes of its sub-elements.

A *programmable block* is a block consisting of programmable gates or programmable blocks. It is programmed by programming each of its sub-elements.

A *generalized Universal Circuit* (GUC) $\mathrm{UC}_{k,u,v\times d}$ is a programmable block with u inputs and v outputs that can be programmed (denoted by $\mathrm{UC}^C_{k,u,v\times d}$) to simulate any circuit C with up to u inputs, v outputs and k gates with d inputs, i.e., $\forall (in_1, .., in_u) \in \{0,1\}^u : \mathrm{UC}^C_{k,u,v\times d}(in_1, .., in_u) = C(in_1, .., in_u)$.

We use programmable block constructions from [10] with the given number of in- and outputs and the following informal functionalities (see [10] for exact definitions and constructions): Y switching block (Y_1^2 programmable as left: $out_1 = in_1$ or right: $out_1 = in_2$), X switching block (X_2^2 programmable as pass: $(out_1, out_2) = (in_1, in_2)$ or cross: $(out_1, out_2) = (in_2, in_1)$), S_v^u selection block (programmable to select for each of the v outputs any of the u inputs including duplicates). Their sizes and properties are summarized in Table 3 in §B.

Our constructions use different compositions of wires:
A (single) *wire* has value either 0 or 1 and is drawn as thin arrow (⟶).

A *multi wire* W consist of ω wires with fixed ordering. The single wires can be indexed by $W[1], .., W[\omega]$. The value of W is the unsigned integer value $w = \sum_{i=1}^{\omega} 2^{i-1} W[i]$. Multi wires are drawn as filled thick arrows (➡).

A *bundle* consists of wires with irrelevant ordering and no duplicates (no two wires are the output of the same gate). $u \times d$ denotes u bundles of d wires each.

Exact calculations of the sizes of our constructions and building blocks with all intermediate steps are given in the full version of this paper.[2]

3 Bundle Blocks for GUC Constructions

The main difference of our efficient GUC constructions compared to previous UC constructions is to switch bundles of d wires instead of single wires only. To do this, we construct efficient bundle blocks that are used as the fundamental building blocks of the GUC constructions described in §4.

C_d^u Choice Block. is a programmable block that can be programmed to choose from the u inputs a bundle of d distinct values as outputs (without recurrence) where the order of the outputs does not matter. More formally, given a subset $\Gamma \subseteq \{1, .., u\}, |\Gamma| = d$, the choice block computes $C(in_1, .., in_u) = (in_{\gamma_1}, .., in_{\gamma_d})$ where $\Gamma = \{\gamma_1, .., \gamma_d\}$ (note, the set equality implies irrelevance of ordering and no duplicate inputs). Of course the definition of a choice block makes only sense for $u \geq d$ as Γ is undefined for $u < d$.

A simple implementation of a C_d^u choice block, $C_d^{u,\mathbf{simple}}$, is to use d selection blocks S_1^{u-d+1} in parallel: $out_i = S_1^{u-d+1}(in_i, .., in_{u-d+i})$; $i = 1, .., d$, i.e., the first selection block can be programmed to select any of the inputs

[2] The full version of this paper is available at `http://eprint.iacr.org/2008/453`.

$in_1, .., in_{u-d+1}$, the second any of $in_2, .., in_{u-d+2}$ and so on. This results in $|C_d^{u,\mathbf{simple}}| = d \cdot |S_1^{u-d+1}| = du - d^2$. The equation also holds true for $u = d$ where the choice block consists of wires only and its size is 0. The straight-forward programming algorithm works as follows: sort Γ ascendingly; for $i = 1, .., d$ do: let k be the smallest element in Γ; program the i-th selection block to select in_k; remove k from Γ; next i. Correctness and efficiency are easy to verify.

Alternatively, a choice block, $C_d^{u,\mathbf{sublin}}$, which is much more efficient for larger d can be derived as a special case of bundle permutation block described next.

$\boldsymbol{BP_{v\times d}^{u\geq vd}}$ **Bundle Permutation Block.** is a programmable block that can be programmed to permute the u inputs to v bundled outputs of d wires each (without duplicates). We define bundle permutation blocks to have at least as many inputs as outputs: $u \geq vd$. More formally, let $S \subseteq 1, .., u; |S| = vd$ be the subset of inputs that are chosen as outputs and $\Phi = (\Phi_1, .., \Phi_v)$ be an ordered partition of S with $\bigcup\limits_{i=1}^{v} \Phi_i = S; |\Phi_i| = d$, the block computes $BP(in_1, .., in_u) = (in_{\varphi_{1,1}}, .., in_{\varphi_{1,d}}, .., in_{\varphi_{v,1}}, .., in_{\varphi_{v,d}})$, with $\Phi_i = \{\varphi_{i,1}, .., \varphi_{i,d}\}; i = 1, .., v$.

Our efficient implementation of $BP_{v\times d}^{u\geq vd}$ bundle permutation block is based on the *truncated permutation block* construction of [10]. A $TP_v^{u\geq v}$ truncated permutation block is a programmable block that can be programmed to permute its $u \geq v$ inputs to its v outputs (u and v need not be equal or powers of two as in [26]). Remaining $u - v$ inputs are discarded (truncated permutation).

TP_v^u block is constructed recursively from two $TP_{v/2}^{u/2}$ sub-blocks and $u/2 - 1$ X switching blocks on top that distribute the inputs to the sub-blocks and $v/2$ X switching blocks on bottom to distribute their outputs to the outputs of TP_v^u block as shown in Fig. 1(a). W.l.o.g. we assume u and v are even at each recursion step (otherwise we introduce an unused dummy input or output with small overhead). Note, that our construction is upside-down compared to the original construction of [10] to have the saved X blocks on top instead (the programming algorithm remains the same using the inverse permutation instead). This modification implies that our GUC construction $M3$ in special case $d = 2$ is more efficient than the original UC construction of [10].

To obtain an efficient $BP_{v\times d}^{u\geq vd}$ bundle permutation block, a $TP_{vd}^{u\geq vd}$ truncated permutation block is constructed without the lowest $\log d$ layers of X switching blocks which can be replaced with wires as the order within each bundle of d wires is irrelevant. Hence, $|BP_{v\times d}^u| = |TP_{vd}^{u\geq vd}| - |X| \cdot \log d \cdot dv/2 = (u + dv) \log v + (\log d + 1)u - 3dv + 2$. An efficient programming algorithm for bundle permutation blocks can easily be derived from the one given in [10].

Our efficient construction of the bundle permutation block is indeed a generalization of the truncated permutation block of [10] which is a special case for $d = 1$ of our construction ($BP_{v\times 1}^{u\geq v} \equiv TP_v^{u\geq v}$) with exactly the same size $|BP_{v\times 1}^{u\geq v}| = |TP_v^{u\geq v}| = (u + v) \log v + u - 3v + 2$.

By fixing the other parameter $v = 1$, we obtain a more efficient (sub-linear in the number of outputs d) construction for choice blocks, $C_d^{u,\mathbf{sublin}} := BP_{1\times d}^{u\geq d} \equiv C_d^u$, with size $|C_d^{u,\mathbf{sublin}}| = |BP_{1\times d}^{u\geq d}| = (\log d + 1)u - 3d + 2$.

4 Generalized Universal Circuits

A *generalized universal circuit* (GUC) $UC_{k,u,v\times d}$ is a boolean circuit that can be programmed to simulate any circuit with u inputs, v outputs and k gates with $d \geq 2$ inputs each. Existing UC constructions [25,10] can simulate gates with two inputs only and can be seen as a special case of our corresponding generalized constructions for $d = 2$. $UC_{k,u,v\times d}$ has exactly u inputs and v outputs.

Each d input gate of the simulated circuit is simulated within a *gate simulation block*, i.e., a programmable block which can be programmed to simulate the functionality of the gate and has d inputs and 1 output. Gates are simulated in topologic order which can be computed efficiently by topologic sorting in $O(k)$.

In the following, we assume that the order of the inputs of a gate simulation block is irrelevant and no inputs are duplicated. This is the case for gate simulation blocks implemented as a d input programmable gate that is programmed with the same function table as the simulated d input gate of exponential size $|G| = 2^{d-2}$. The entries of the function table can be swapped according to an arbitrary input ordering and duplicate inputs can easily be eliminated. Also for gate simulation blocks that implement neurons as a circuit the order of inputs is irrelevant and duplicate inputs can be eliminated as we will explain in §5. The irrelevance of the input ordering without duplicates is reflected by bundles of d wires as input into a gate simulation block.

If the order of inputs of the simulated gates is relevant or duplicate inputs are needed, the following GUC constructions can be extended by replacing the bundle blocks from §3 with their corresponding non-bundled counterparts where the order of outputs does matter at the cost of a small overhead: $C_d^{u,\mathbf{simple}} \mapsto S_d^u, C_d^{u,\mathbf{sublin}} \mapsto S_d^{u\geq d}, BP_{v\times d}^{u\geq vd} \mapsto TP_{vd}^{u\geq vd}$.

We stress that the sizes of all building blocks and the GUCs presented in the remainder of this section only depend on the parameters u, v, k, d but neither on the input data nor the simulated circuit. Hence, dynamic choice of the smallest implementation for each building block in so called *combined constructions* respectively choosing smallest GUC construction reveals nothing about the input data nor the simulated circuit.

4.1 Iterative GUC Constructions

A simple GUC is constructed iteratively by choosing for the i-th gate simulation block G_i any of the u inputs of the circuit or the output of a previous gate simulation block $G_1, .., G_{i-1}$ with C_d^{u+i-1} choice block. The v outputs of the GUC can be selected to be any of the outputs of the k gate simulation blocks using $S_v^{k\geq v}$ selection block.

Using simple $C_d^{u,\mathbf{simple}}$ choice blocks results in a total size of

$$|UC_{k,u,v\times d}^{\mathbf{I1}}| = 0.5dk^2 + (du - d^2 - 0.5d + |G| + 1)k + (k + 3v)\log v - 4v + 3$$
$$\sim 0.5d \cdot k^2 + d \cdot uk.$$

With sub-linear $C_d^{u,\mathbf{sublin}}$ choice blocks instead the construction has size

$$|UC^{\mathbf{I2}}_{k,u,v\times d}| = (0.5\log d + 0.5)k^2 + (u\log d + u - 0.5\log d - 3d + |G| + 2.5)k \\ + (k+3v)\log v - 4v + 3 \sim 0.5\log d \cdot k^2 + \log d \cdot uk.$$

A combination of both approaches chooses the smallest implementation for each choice block (simple or sub-linear) dynamically.

$$|UC^{\mathbf{I}}_{k,u,v\times d}| \leq min(|UC^{\mathbf{I1}}_{k,u,v\times d}|, |UC^{\mathbf{I2}}_{k,u,v\times d}|).$$

All iterative GUC constructions are only practical for a small number of simulated gates k and few inputs u.

4.2 Modular GUC Constructions

Another approach to construct a GUC is to separate the inputs and the outputs from the simulation of the gates. This results in modular GUC constructions which are a generalization of the modular UC construction of [10]. The *modular GUC* is composed out of three programmable blocks as shown in Fig. 1(b). The *generalized Universal Block (UB)*, $U_{k\times d}$, simulates the k gates, the *input selection block* chooses the corresponding inputs and the *output selection block* chooses the outputs of the simulated gates as outputs of the modular GUC. The construction has size

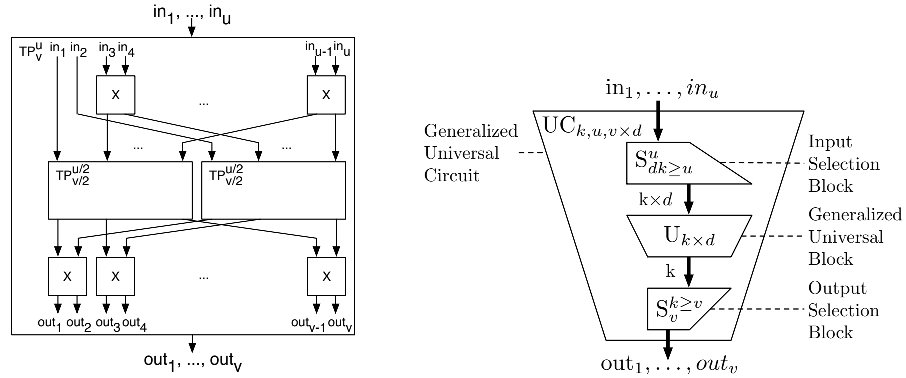
$$|UC^{\mathbf{Mi}}_{k,u,v\times d}| = |S^u_{dk\geq u}| + |U^{\mathbf{Mi}}_{k\times d}| + |S^{k\geq v}_v| \sim 2dk\log k + dk\log u + |U^{\mathbf{Mi}}_{k\times d}|.$$

The overall complexity is determined by the complexity of the generalized UB $U_{k\times d}$ that only depends on the number of simulated gates k and no longer on the number of inputs u or outputs v. The generalized UB $U_{k\times d}$ has k bundles of d inputs where the i-th bundle, $in_{di}, .., in_{di+d-1}$, can be switched to the i-th gate simulation block G_i; the output of G_i is connected to out_i of $U_{k\times d}$.

Next, we give different constructions for generalized UBs that can be plugged into the modular GUC construction. The iterative constructions (generalized from [22, Section 5.3.1]) grow like $d{\cdot}k^2$ and $\log d{\cdot}k^2$ and the recursive construction (generalized from [10]) grows like $dk\log^2 k$.

Iterative Generalized Universal Block Construction. An iterative construction for a generalized UB is similar to the iterative GUC construction described in §4.1 but without the dependancy on the inputs that are handled efficiently by the input selection block of the modular GUC construction. For each gate simulation block G_i, a C_d^{d+i-1} choice block can be programmed to choose any of the d wires of the i-th input bundle of the generalized UB and the $i-1$ outputs of the previous gate simulation blocks $G_1, .., G_{i-1}$ as input to G_i. Using simple $C_d^{u,\mathbf{simple}}$ choice blocks this results in a total size of

$$|U^{\mathbf{M1}}_{k\times d}| = 0.5dk^2 - 0.5dk + k\cdot|G| \sim 0.5d\cdot k^2.$$



(a) Truncated Permutation Block

(b) Modular GUC Construction

Fig. 1. Building blocks for GUCs

With sub-linear $C_d^{u,\mathbf{sublin}}$ choice blocks instead the size is

$$|U_{k\times d}^{\mathbf{M2}}| = (0.5\log d + 0.5)k^2 + (d\log d - 0.5\log d - 2d + |G| + 1.5)k$$
$$\sim 0.5\log d \cdot k^2.$$

Both modular iterative constructions still grow like k^2 but are more efficient than the iterative GUC constructions from §4.1 for circuits with many inputs due to the efficient handling of inputs with the input selection block.

Recursive Generalized Universal Block Construction. A generalization of the recursive UB construction of [10] yields a generalized UB of size

$$|U_{k\times d}^{\mathbf{M3}}| = (0.625d + 0.25)k\log^2 k + (0.5d\log d - 0.625d - 1.25)k\log k$$
$$+ (|G| + 3)k - 3 \sim 0.625dk\log^2 k.$$

The detailed description of the construction is in the full version of this paper. Compared to the constructions presented before, the recursive construction grows like $k\log^2 k$ instead of k^2 which is clearly much slower for larger circuits.

Combined Generalized Universal Block Construction. A combination of these generalized UB constructions uses the smallest generalized UB implementation (M1, M2 or M3) dynamically. *Dynamic Programming* avoids recalculation of the smallest construction for given parameters by caching it in a table.

$$|U_{k\times d}^{\mathbf{M}}| \leq min(|U_{k\times d}^{\mathbf{M1}}|, |U_{k\times d}^{\mathbf{M2}}|, |U_{k\times d}^{\mathbf{M3}}|)$$

4.3 Universal Graph Based GUC Construction

A generalization of Valiant's UC construction [25] which is based on Universal Graphs results in a GUC construction of size

$$|UC_{k,u,v\times d}^{\mathbf{UG}}| \sim 4.75d(2k + u + \frac{v}{d-1})\log k.$$

This GUC construction is asymptotically better than those shown before for large circuits and is based on the following theorem.

Theorem 1. *Each circuit of arbitrary fan-out with v outputs and k gates of d inputs each can be converted into an equivalent circuit with fan-out $\leq d$ by adding at most $k + \frac{v}{d-1}$ gates.*

For lack of space, the detailed description of this construction and the proof of the theorem are given in §A.

4.4 Comparison of GUC Constructions

The sizes of the different GUC constructions for several practical parameters are shown in Fig. 4 of §B. Which construction is the smallest (and hence should be used for least overhead) depends on the parameters d, k, u, and v only. Quadratic constructions ($I1, I2, M1, M2$) are suitable for small, whereas recursive construction ($M3$) is better for mid-size and universal graph based construction (UG) for large circuits.

5 Secure Evaluation of Private NNs with GUCs

5.1 Structure of NNs

A *neural network* (NN) is an acyclic directed graph of several neurons. The neurons are arranged in multiple layers (Fig. 2) in topologic order. Each neuron has d input bits, one output bit, internal precision of s bits and is structured as shown in Fig. 3(a). Each input bit in_i of a neuron is multiplied with a s-bit constant weight w_i. The Σ block computes the sum σ of these weighted inputs. The threshold function τ compares σ with a threshold value t and sets the output: if $\sigma := \sum_{i=1}^{d} w_i \cdot in_i < t$, then $out = 0$, else $out = 1$.

The input neurons of the first layer in the NN have only 1 input with multiple bits m. They can also be realized with a neuron as shown in Fig. 3(a) by setting the weights correspondingly.

Neurons fulfill the restrictions for d input gates of §4: Inputs can be permuted arbitrarily by permuting their weights in the same way. Duplicate inputs into a neuron with the same source can be merged into one input by adding their weights. Implementation of neurons presented next guarantees that the weights of neurons remain hidden from requester and hence these modifications are not detectable for him.

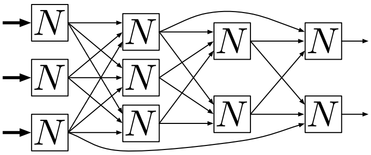
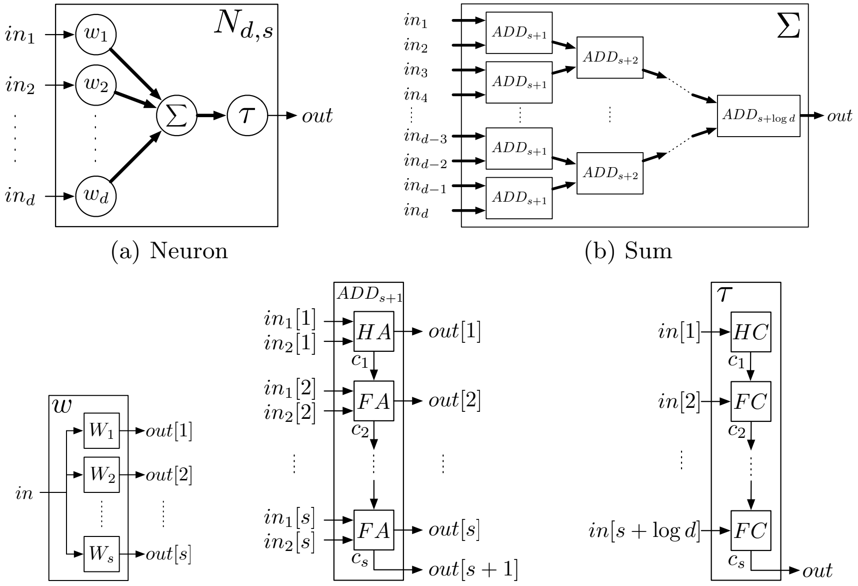


Fig. 2. NN with 4 layers of $k = 10$ neurons with degree $d = 3$ including $u = 3$ input neurons and $v = 2$ outputs

5.2 Circuit Implementation of Neurons

If the number of inputs d is small, each neuron can be implemented as programmable d input gate of size $|N_{d,s}^{\mathbf{gate}}| = 2^{d-2}$ (for arbitrary activation function), otherwise as programmable d input block (Fig. 3(a)). The neuron can be programmed depending on the weights $\omega_1, .., \omega_d$ and the threshold value t.



(a) Neuron

(b) Sum

(c) Weights

(d) Adder

(e) Threshold

Fig. 3. Circuit implementation of a neuron

As before we give directly the total size of the building blocks. Exact calculations with all intermediate steps are given in the full version of this paper.

The w block multiplies its input bit with constant ω of s bits (Fig. 3(c)). The bits $\omega[i]$, $i = 1, .., s$, determine the programming of the programmable gate W_i: if $\omega[i] = 0$, then $W_i = 0$, else $W_i = in$. The size is $|W_i| = 0.5$, $|w| = s \cdot |W_i| = 0.5s$.

The Σ block sums up the d input values of s bits each to a $s + \log d$ bit value by pairwise adding them in a tree (Fig. 3(b)). An adder ADD_{s+1} to add two s bit values to an $s + 1$ bit value is composed as usual from a half adder HA and $s - 1$ full adders FA (Fig. 3(d)). The size is $|HA| = 2$, $|FA| = 4$, $|ADD_{s+1}| = 4(s + 1) - 6$, $|\Sigma| < 4ds + 2d - 4s + 6$.

The τ block compares the $s + \log d$ bit input with an $s + \log d$ bit constant t (Fig. 3(e)). The carry $c_i = 0$, $i = 1, .., s+\log d$, tells that the i least significant bits of in are less than the i least significant bits of t: if $((in \mod 2^i) < (t \mod 2^i))$, then $c_i = 0$, else $c_i = 1$. Depending on $t[1]$, the programmable half comparer block HC is programmed: if $t[1] = 0$, then $HC = 1$, else $HC = in_1$. The remaining bits $t[i]$, $i = 2, .., s+\log d$, determine the program of the programmable full comparer blocks FC_i: if $t[i] = 0$, then $FC_i = in_i \vee c_{i-1}$, else $FC_i = in_i \wedge c_{i-1}$. The size is $|HC| = 0.5$, $|FC| = 1$, $|\tau| = s + \log d - 0.5$.

The total size of a neuron implemented as programmable block is $|N^{\mathbf{block}}_{d,s}| < 4.5ds + 2d - 3s + \log d + 5.5$.

5.3 Protocol for Oblivious Evaluation of NNs Using GUCs

Oblivious evaluation of NNs is reduced to SFE of GUCs similar to the reduction for PF-SFE [10]. A GUC is programmed to simulate the structure of the NN. Each gate simulation block G is instantiated with a programmable circuit for a neuron $N_{d,s}$ programmed with the coefficients of the neuron it simulates: $UC_{NN} = UC_{k,u,v \times d}|_{G=N_{d,s}}$. Programmed GUC simulates the NN and entirely hides its structure (besides size and number of inputs and outputs): $\forall (in_1, .., in_u) \in \{0,1\}^u : \mathrm{UC}_{NN}(in_1, .., in_u) = NN(in_1, .., in_u)$.

When requester evaluates the programmed UC_{NN} with a SFE protocol, he learns no more about NN than the maximal number of neurons k, maximal degree of neurons d, internal precision s, inputs u and outputs v.

The protocol needs one round in the semi-honest model (using interactive OT such as [19,1,8]) or is non-interactive (using non-interactive OT of [6]). Recall, the reduction from PF-SFE to SFE using UC (resp. GUC in our case) is non-cryptographic and the security of the PF-SFE protocol is exactly that of the underlying SFE protocol which is provably secure against semi-honest adversaries (e.g., [27,12,9]). This can be extended to be provably secure against malicious adversaries by using correspondingly secure SFE protocols (e.g., [14,13,5]) which need more than one but still a constant small number of rounds.

5.4 Comparison with Previous Work

We compare our GUC constructions with existing UC constructions before comparing our protocol for secure evaluation of NNs with existing protocols. As example for both comparisons we use the practical NN to classify sonar targets from [4] for which performance results are given in [15]. However, we use threshold instead of sigmoid as activation function as performance of [15] is almost the same independent of the used activation function. Besides this we use exactly the same parameters for the neural network, namely $u = 60$ inputs, $v = 2$ outputs, $k = 12$ hidden neurons with an internal resolution $s = 20$ (corresponding to their quantization factor $Q = 10^{-6}$). In order to obfuscate the structure of NN they propose to embed NN into a grid of 5 layers with 15 neurons each. Hence, the in-degree of each neuron is $d = 15$ while the total number of neurons is hidden to be less than $k = 75$. This results in neurons of size $|N| = |N^{\mathbf{block}}_{d=15,s=20}| \leq 1,330 < |N^{\mathbf{gate}}_{d=15,s=20}| = 8,192$.

We compare three possible alternatives that protect the internal weights and thresholds of the neurons and incrementally protect the structure of the NN:

(A) Embed NN into a 5 × 15 grid to obfuscate its structure (same as [15]): evaluate $75 \cdot |N| \leq 99,750$ gates.

(B) Hide the structure of the NN entirely:
simulate NN of $k = 12$ neurons ($12 \cdot |N| \leq 15{,}960$ gates) in GUC ($|UC^{\mathbf{M1}}_{k=12,u=60,v=2\times d=15}| \leq 2{,}304$ gates, cf. Fig. 4(g) in §B).

(C) Additionally hide the number of neurons to be less than 75:
simulate NN of $k = 75$ neurons ($75 \cdot |N| \leq 99{,}750$ gates) in GUC ($|UC^{\mathbf{M2}}_{k=75,u=60,v=2\times d=15}| \leq 27{,}371$ gates, cf. Fig. 4(h) in §B).

Comparison of GUC Construction. As shown in Table 1, the overhead of our GUC introduced in case (B) and (C) is very moderate compared to known UC constructions. Applying *UC* directly results in an overhead which is by a factor of $> 10^3$ times bigger, while using $\lceil d/2 \rceil = 8$ parallel UCs still is by a factor of more than two times bigger than our solution.

Table 1. Comparison of UC overhead (in number of gates)

	UC		Parallel UCs		GUC
	[25]	[10]	[25]	[10]	§4
(B): $k = 12$	4,242,114	5,556,431	23,432	6,577	2,304
(C): $k = 75$	31,482,358	47,478,158	100,359	58,618	27,371

Comparison of Protocol for Secure Evaluation of NNs. We used Fairplay SFE system [14] implemented in Java as well without cut and choose step to evaluate a circuit with $u = 60$ inputs, $v = 2$ outputs and the given number of gates within two processes on a notebook with 2.16 GHz Intel Core 2 processor and 2 GB memory. As [15] do not specify the exact hardware used ("two mid-level notebooks, connected on a LAN network"), the results of the comparison shown in Table 2 are qualitative but not necessarily quantitative:

Unlike the protocol in [15], ours are provably secure and approaches (B) and (C) hide the structure of NN better than just obfuscating it.

The total amount of communication overhead of [15] is by a factor of at least 10 times better than our solutions, however they need multiple rounds (for each layer of the NN) whereas our solutions need only one round in the semi-honest model or even no online-communication at all (using non-interactive OT). On analyzing the communication complexity separately for server and client we see that in our solutions the amount of data sent by the client is much smaller than that of the server and only depends on the number of inputs but not on the size of NN. The amount of data sent by client is by a factor of five less than that in [15] (as the client in their symmetric protocol sends approximately half of the total data). This asymmetry in the communication exactly corresponds to modern communication networks such as mobile networks or the internet, where the upstream of the client is much slower than its downstream. Downloading the maximum amount of 5.4 MB in (C) is realistic for today's mobile networks.

The total time for executing the protocol of (A) and (C) is almost the same as that of [15] while (B) is almost three times faster. While in [15] client has to do only 20% of the work, in our protocols, server and client need approximately the same amount of computation for creating and evaluating the garbled

Table 2. Comparison of protocols for secure evaluation of NNs

Protocol	[15]	(A)	(B)	(C)
Level of privacy	obfuscate	obfuscate	hide structure	hide structure&size
Provably secure	no	yes		
Communication (Total)	76 kB	4.2 MB	0.79 MB	5.4 MB
Server (send)	$\approx$ 38 kB	4.2 MB	0.78 MB	5.4 MB
Client (send)	$\approx$ 38 kB	7.5 kB		
Rounds	5	1 (0)		
Computation (Total)	11.7 s	11.3 s	4.0 s	13.4 s
Server	9.3 s	$\approx$ 5.7 s	$\approx$ 2.0 s	$\approx$ 6.7 s
Client	2.4 s	$\approx$ 5.7 s	$\approx$ 2.0 s	$\approx$ 6.7 s

circuit. Online-computation of our server can be avoided almost completely by constructing the garbled circuit in advance while server is idle. This reduces total execution time of our protocols by half.

Using [9] as underlying SFE protocol or a high-speed SFE implementation such as [11] written in C with elliptic curve based OT would further improve communication and computation complexity of our protocols.

Acknowledgments. We thank Vladimir Kolesnikov and anonymous reviewers for their helpful comments.

References

1. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)
2. Chang, Y.-C., Lu, C.-J.: Oblivious polynomial evaluation and oblivious neural learning. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 369–384. Springer, Heidelberg (2001)
3. Drewes, R.: An artifical neural network spam classifier (August 2002), `http://www.interstice.com/drewes/cs676/spam-nn/`
4. Gorman, R.P., Sejnowski, T.J.: Analysis of hidden units in a layered network trained to classify sonar targets. Neural Networks 1(1), 75–89 (1988)
5. Goyal, V., Mohassel, P., Smith, A.: Efficient two party and multi party computation against covert adversaries. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 289–306. Springer, Heidelberg (2008)
6. Gunupudi, V., Tate, S.R.: Generalized non-interactive oblivious transfer using count-limited objects with applications to secure mobile agents. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 98–112. Springer, Heidelberg (2008)
7. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2(5), 359–366 (1989)
8. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending Oblivious Transfers Efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)

9. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
10. Kolesnikov, V., Schneider, T.: A practical universal circuit construction and secure evaluation of private functions. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 83–97. Springer, Heidelberg (2008), `http://thomaschneider.de/FairplayPF`
11. Lindell, Y., Pinkas, B., Smart, N.: Implementing two-party computation efficiently with security against malicious adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
12. Lindell, Y., Pinkas, B.: A proof of Yao's protocol for secure two-party computation. Cryptology ePrint Archive, Report 2004/175 (2004)
13. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
14. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay — a secure two-party computation system. In: USENIX (2004), `http://www.cs.huji.ac.il/project/Fairplay/fairplay.html`
15. Orlandi, C., Piva, A., Barni, M.: Oblivious neural network computing via homomorphic encryption. European Journal of Information Systems (EURASIP) 2007(1), 1–10 (2007)
16. Pinkas, B.: Cryptographic techniques for privacy-preserving data mining. SIGKDD Explor. Newsl. 4(2), 12–19 (2002)
17. Piva, A., Caini, M., Bianchi, T., Orlandi, C., Barni, M.: Enhancing privacy in remote data classification. In: New Approaches for Security, Privacy and Trust in Complex Environments (SEC 2008) (2008)
18. Plagianakos, V.P., Vrahatis, M.N.: Parallel evolutionary training algorithms for hardware-friendly neural networks. Natural Computing 1(2-3), 307–322 (2002)
19. Rabin, M.O.: How to exchange secrets with oblivious transfer. Technical report, Harward University, Available at Cryptology ePrint Archive, Report 2005/187 (1981)
20. Sander, T., Young, A., Yung, M.: Non-interactive cryptocomputing for NC^1. In: Proc. 40th IEEE Symp. on Foundations of Comp. Science, FOCS 1999, New York, pp. 554–566. IEEE, Los Alamitos (1999)
21. Sato, K., Hikawa, H.: Implementation of multilayer neural network with threshold neurons and its analysis. Artificial Life and Robotics 3(3), 170–175 (1999)
22. Schneider, T.: Practical secure function evaluation. Master's thesis, University of Erlangen-Nuremberg (2008), `http://thomaschneider.de/theses/da/`
23. StatSoft, Inc. STATISTICA Automated Neural Networks (2008), `http://www.statsoft.com/products/stat_nn.html`
24. Tebelskis, J.: Speech Recognition using Neural Networks. PhD thesis, School of Computer Science, Pittsburgh (1995)
25. Valiant, L.G.: Universal circuits (preliminary report). In: STOC 1976, pp. 196–203. ACM Press, New York (1976)
26. Waksman, A.: A permutation network. J. ACM 15(1), 159–163 (1968)
27. Yao, A.C.: How to generate and exchange secrets. In: FOCS 1986, Toronto, pp. 162–167. IEEE, Los Alamitos (1986)

A Universal Graph Based GUC Construction

To construct an asymptotically better UC than the practical constructions presented before we generalize Valiant's UC construction [25]. Valiant shows how to construct a UC by embedding the given circuit into a universal graph $\Gamma_d(k')$ with $|\Gamma_d(k')| \sim 4.75dk' \log k'$, where d is the fan-in and fan-out of the simulated graph and k' is the number of simulated nodes. A circuit with u inputs and k gates having fan-in and fan-out maximum d can be represented as such a simulateable graph with $k' = k + u$ nodes and embedded into this universal graph. We generalize Valiant's construction (that uses $d = 2$ and can simulate circuits with gates having 2 inputs only) to a GUC that simulates circuits with arbitrary fixed in-degree d and arbitrary fan-out of size

$$|UC^{\mathbf{UG}}_{k,u,v\times d}| \sim 4.75d(2k + u + \frac{v}{d-1}) \log k.$$

The circuit is converted from arbitrary fan-out to a circuit with fan-out at most d which can be embedded into the universal graph $\Gamma_d(k')$. This is done by replacing each gate with fan-out $x > d$ by a binary tree of $\lceil \frac{x}{d-1} \rceil + 1$ gates with fan-out $\leq d$. At most $e = k + \frac{v}{d-1}$ extra gates are needed as described in §A.1. Setting $k' = k + u + e$ results in the stated complexity for the generalized Valiant's UC construction. By setting $d = 2$ we obtain exactly the asymptotic complexity of Valiant's original UC construction.

A.1 Converting Circuit to Fan-Out $\leq d$

As described in §A, a circuit with s gates of arbitrary fan-out and m outputs can be converted into one having gates with fan-out $\leq d$ only, by replacing each gate with fan-out $x > d$ by a binary tree of $\lceil \frac{x}{d-1} \rceil + 1$ gates with fan-out $\leq d$ each.

Theorem 1 in §4.3 gives an upper bound for the maximal number of extra gates e added which we prove similar to [25, Fact 3.1 and Corollary 3.1]:

Fact: Suppose that a graph G with fan-in d has among the nodes of in-degree zero, n' nodes of nonzero out-degree, and amongst the rest v' nodes of out-degree zero, f_i of out-degree i, $i = 1, .., d-1$, and g of out-degree greater than $d-1$. Suppose also that $e' = \sum(out - degree - d) = \sum x$ over the set of nodes with out-degree greater than d. Then $e' \leq \sum_{i=1}^{d-1}(d-i)f_i + dv' - n'$.

Proof. The total of the out-degrees must equal the total of the in-degrees. The former is $\geq n' + \sum_{i=1}^{d-1}(if_i) + dg + e'$, and the latter is $\leq d(v' + \sum_{i=1}^{d-1} f_i + g)$. □

Corollary: $e \leq k + \frac{v}{d-1}$

Proof. Any gate with fan-out $x + d$, $x > 0$, can be replaced by a binary tree with $\lceil \frac{x}{d-1} \rceil + 1 \leq \frac{x}{d-1} + 2$ gates. Hence for any circuit there is an equivalent one having fan-out d and at most $e \leq \frac{e'}{d-1} + g$ more gates. But the total number of gates $k = \sum_{i=1}^{d-1} f_i + g + v'$ and in any minimal circuit $v' \leq v$. Hence at most

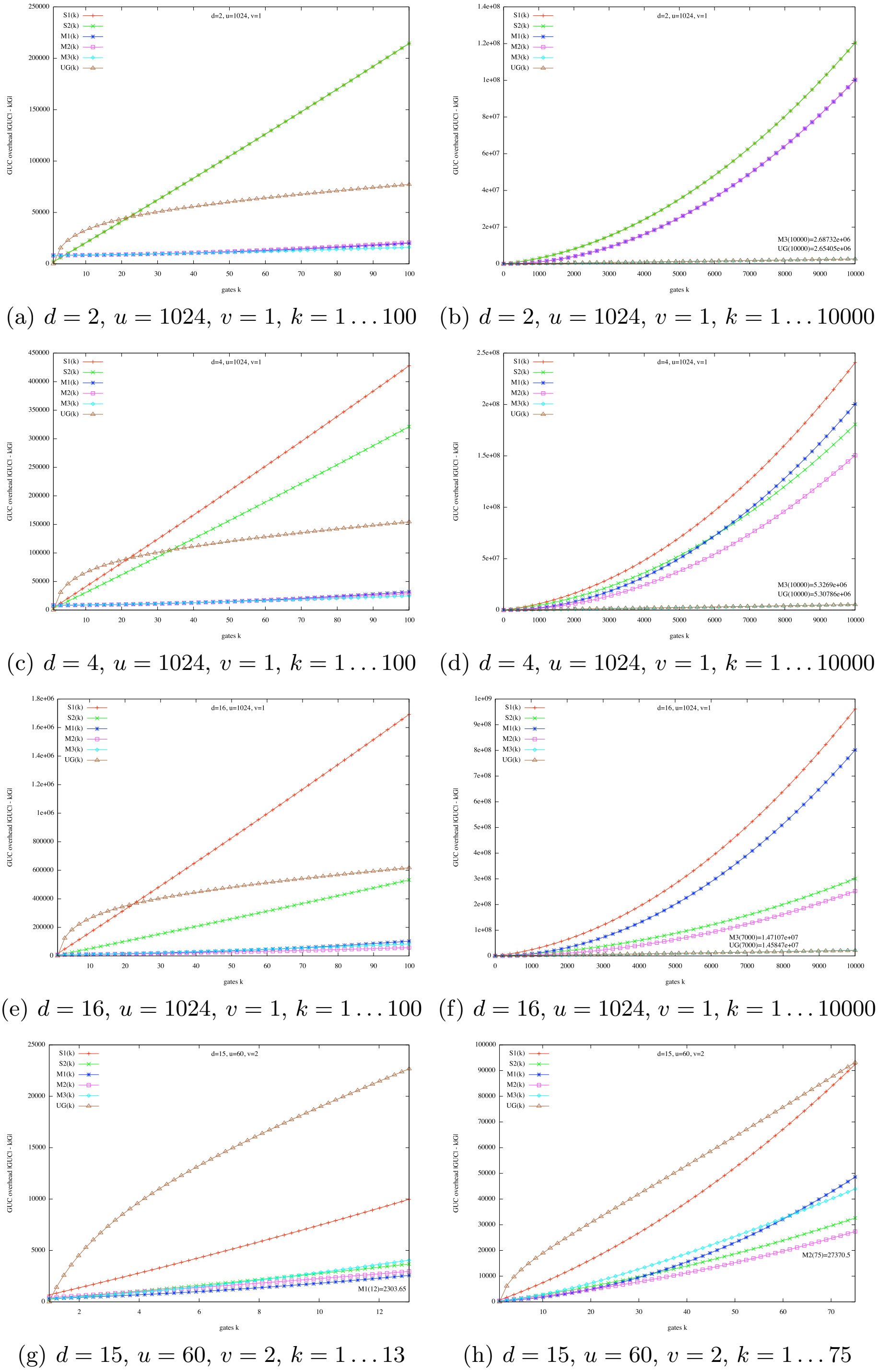
$$e \leq \frac{e'}{d-1} + g \leq \frac{d}{d-1}v' + \sum_{i=1}^{d-1} \frac{d-i}{d-1} f_i + g \leq s + \frac{d}{d-1}v' - v' \leq k + \frac{v}{d-1}$$

extra gates are needed which completes the proof of the corollary and Theorem 1. □

B Tables and Figures

Table 3. Programmable switching blocks

Block	Name of Block	Size	Duplicates	Order
S_1^u [10]	Selection	$u - 1$	X	X
S_v^u [10]	Selection (simple)	$v(u-1)$	X	X
$S_{v \geq u}^u$ [10]	Selection (efficient)	$(u+v)\log u + 2v \log v - 2u - v + 3$	X	X
$S_v^{u \geq v}$ [10]	Selection (efficient)	$(u+3v)\log v + u - 4v + 3$	X	X
S_{2u}^u [10]	Selection (improved)	$6u \log u + 3$	X	X
P_u^u [26]	Permutation	$2u \log u - 2u + 2$	-	X
$EP_{v \geq u}^u$ [10]	Permutation (expanded)	$(u+v)\log u - 2u + 2$	-	X
$C_d^{u,\mathbf{simple}}$ (§3)	Choice (simple)	$du - d^2$	-	-
$C_d^{u,\mathbf{sublin}}$ (§3)	Choice (sub-linear)	$(\log d + 1)u - 3d + 2$	-	-
$BP_{v \times d}^{u \geq vd}$ (§3)	Bundle Permutation	$(u + dv)\log v + (\log d + 1)u - 3dv + 2$	-	-



(a) $d = 2,\ u = 1024,\ v = 1,\ k = 1 \ldots 100$ (b) $d = 2,\ u = 1024,\ v = 1,\ k = 1 \ldots 10000$

(c) $d = 4,\ u = 1024,\ v = 1,\ k = 1 \ldots 100$ (d) $d = 4,\ u = 1024,\ v = 1,\ k = 1 \ldots 10000$

(e) $d = 16,\ u = 1024,\ v = 1,\ k = 1 \ldots 100$ (f) $d = 16,\ u = 1024,\ v = 1,\ k = 1 \ldots 10000$

(g) $d = 15,\ u = 60,\ v = 2,\ k = 1 \ldots 13$ (h) $d = 15,\ u = 60,\ v = 2,\ k = 1 \ldots 75$

Fig. 4. Comparison of GUC constructions for different parameters

Proving a Shuffle Using Representations of the Symmetric Group[*]

Soojin Cho[1] and Manpyo Hong[2]

[1] Department of Mathematics, Ajou University,
San5 Woncheon-Dong, Youngtong-Gu, Suwon, 443-749 Korea
chosj@ajou.ac.kr

[2] Department of Information and Computer Engineering, Ajou University
San5 Woncheon-Dong, Youngtong-Gu, Suwon, 443-749 Korea
mphong@ajou.ac.kr

Abstract. Shuffling protocol proposed in Crypto 2005 by Peng et al. is improved so that the number of communication rounds for the verification is reduced. We use an idea of linear representations of the symmetric group and a property of the incidence matrices of 1-subsets and 2-subsets of a finite set. The proposed protocol is valid for mix networks implemented with Paillier encryption schemes with which we can apply some known zero-knowledge proofs following the same line of approaches of Peng et al. [24]. The overall cost for the verification, *if* we fully implement our idea, is more expensive than that of the original protocol by Peng et al. We, *however*, can control the level of computation cost for the verification by using the idea of λ-*designs* properly.

1 Introduction

Since D. Chaum proposed the scheme of mix-net [3] as a primitive for anonymous communications, enormous amount of research and great improvement on mix-net has been made in many different perspectives: New encryption schemes were employed [23,21,14], weaknesses pointed out through many analyses [27,26,18] of the early construction of mix-net has produced more advanced and securer mix networks [21,6]. For many different purposes for anonymity, various systems were developed; for web services, real time systems were developed [9], and for mailing services non real time systems like babel, mixmaster and mixminion were cultivated [5]. Network topology and mixing mechanism are some of other concerns in constructing mix-nets [4,8]. Measuring the anonymity of mix-nets is another important fundamental work [29,7].

One of the most important matters at present is on the proof of correctness of the mix net. Roughly speaking, there are two kinds of proof system of mix-nets; one is optimistic and the other is verifiable proof system. The correctness

[*] This research is supported by Foundation of ubiquitous computing and networking project(UCN) Project, the Ministry of Knowledge Economy(MKE) 21st Century Frontier R&D Program in Korea and a result of subproject UCN 08B3-B-30S.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 354–367, 2009.

of the shuffling of the whole mix-net is verified after the mix-net outputs the shuffling results in plain texts in *optimistic* proof system [15], while in *verifiable* proof system each mix server provides proofs of correctness of the shuffling [25,12,1,16,19,20,24,30].

Techniques of *verification* for mix-nets vary according to the mix-net implementation and vice versa. Permutation network is the framework of the verification protocol proposed by Abe [1], ElGamal re-encryption scheme and Paillier re-encryption scheme are used for the proof system of Furukawa-Sako [12] and Nguyen et al. [20] respectively. A more general scheme than ElGamal is allowed to apply the verification by Neff [19]. In the proof system by Groth [16], additively homomorphic re-encryption scheme is necessary. In sender verifiable mix-net based on ElGamal encryption by Wikström, no re-encryption is needed. In [25], Peng et al. applied the idea of Abe [1] to design a proof system for mix-nets employing ElGamal re-encryption system and a very carefully designed proof system by Peng et al. [24] assumes the additively homomorphic (re-)encryption schemes. A recent work by Groth and Lu [17] is based on the homomorphic encryption scheme also, in which computational complexity has been dramatically reduced.

The main concern of this article is on the universally verifiable proof system of re-encryption mix-nets implemented with Paillier encryption scheme [22] as the work by Peng et al. was done in the same line: We extend the idea of [24] for the verification of a shuffle by considering pairs of messages as well as single messages. This enables us to do the verification in *two* communication rounds rather than *four* rounds as in [24]. Computationally improved protocol is also suggested using the idea of λ-designs.

1.1 Related Works

In their system of proof, Furukawa-Sako [12] describes an equivalent condition for a matrix to be a permutation matrix which involves quadratic and cubic relations among the entries of a given matrix. The proof of the shuffle for ElGamal encryption scheme and Paillier encryption scheme have been implemented in [12] and [20] respectively.

In [25], Peng et al. restrict the set of available permutations and employ batch verifications of knowledge to reduce the computational cost of the proof. Recently, Groth and Lu suggested very efficient schemes extending the idea of previously known works on the verification [17].

In [24], Peng et al. proposed a very carefully designed, efficient proof system on mix-nets with additively homomorphic re-encryption scheme: Let $E(m, r)$ be the encryption of the message m with randomizer r and $D(c)$ be the decryption of ciphertext c. When $c_1, c_2, \ldots, c_n$ is the list of cipher texts a shuffling party receives from the previous shuffling party, let $c'_1, c'_2, \ldots, c'_n$ be the list of outputs of the current shuffling party that is supposed to be passed to the next shuffling party. Then, when N is the modulus of the message space, the basic idea of the proof is to let the shuffling party do the zero-knowledge proof of the followings; for random integers s_i, s'_i, $i = 1, 2, \ldots, n$, (s)he knows t_i, t'_i, $i = 1, 2, \ldots, n$, such that

$$\sum_{i=1}^{n} s_i D(c_i) = \sum_{i=1}^{n} t_i D(c'_i) \pmod N\,, \tag{1}$$

$$\sum_{i=1}^{n} s'_i D(c_i) = \sum_{i=1}^{n} t'_i D(c'_i) \pmod N\,, \tag{2}$$

$$\sum_{i=1}^{n} s_i s'_i D(c_i) = \sum_{i=1}^{n} t_i t'_i D(c'_i) \pmod N\,. \tag{3}$$

For the proof of (1), (2) and (3), at least *four* communication rounds are necessary. The main goal of our work is to reduce the number of rounds in the same framework of Peng et al. while maintaining the reasonable computational complexity.

1.2 Motivation and the Outline of Our Work

For a positive integer n, permutations of n objects form a group S_n called *symmetric groups* of order $n!$. *Group representations* are representations of group elements as matrices so that the structure of the given group is preserved. To represent every permutation in S_n as an $n \times n$ permutation matrix as in [12] is a representation of S_n called *defining representation* [28]. It is known, in representation theory of symmetric groups[1], that every representation of S_n is isomorphic to a direct sum of irreducible representations indexed by the partitions[2] of n. From this, extending the work of Furukawa-Sako, we may consider some natural ways to represent the given permutation as a matrix of various dimensions. Since the representation Furukawa-Sako used is basically corresponding to the partition $(n-1,1)$, we may consider the partition $(n-2,2)$ as a second simplest case. The representation of type $(n-2,2)$ describes how a given permutation permutes 2-subsets of $\{1,2,\ldots,n\}$, while the one of type $(n-1,1)$ is to look at how the given permutation permutes elements (or 1-subsets) of $\{1,2,\ldots,n\}$. This means that the representation of a permutation of type $(n-2,2)$ is an $\binom{n}{2} \times \binom{n}{2}$ matrix. We somehow expect that this type of (linear) representation would let us explain non-linear relations that are indispensable to prove that a given matrix is a permutation matrix in *zero-knowledge* manner: Furukawa-Sako [12] used quadratic and cubic relations of the entries of a given matrix, and Peng et al. used a quadratic relation of the entries of a transformation matrix between two sets of decrypted messages, that a shuffling party used.

Using the idea explained above, we implement a proof system for a shuffling with only *two* communication rounds. We have to pay more computational cost for the verification in general instead. We, however, can control the level of computation cost by adopting the idea of λ-*designs*.

[1] The theory of representations of group varies depending on the ground field used. We, however, do not take this issue seriously, since in our case, n can be assumed to be relatively prime to the modulus in use.

[2] *A partition* of a positive integer n is a non-increasing finite sequence of positive integers, whose sum is n. For example, $(4,3,3,1)$ is a partition of 11.

Our method of proof is to let each shuffling party prove the followings, instead of equations (1),(2) and (3); for given random integers s_{ij}, $1 \leq i \neq j \leq n$, he knows t_{ij}'s, such that

$$\sum_{i<j} s_{ij}(D(c_i) + D(c_j)) = \sum_{i<j} t_{ij}(D(c'_i) + D(c'_j)) \pmod N\,,$$

$$\sum_{i<j} s_{ij} D(c_i) D(c_j) = \sum_{i<j} t_{ij} D(c'_i) D(c'_j) \pmod N\,.$$

In Section 2, the basics for our mix-net and shuffling is set up and some basic necessary results are presented. In Section 3, the protocol for the proof is given, while the proofs of its correctness and privacy is given in Section 4. In Section 5, we describe an (computationally) improved version of our protocol, and in Section 6 computational cost of our protocol is calculated. We conclude with some final remarks on our protocol and Peng et al.'s.

2 Preliminaries

2.1 Encryption System

As the proof system of Peng et al. [24] does, our proof system relies on the property of homomorphic encryption. We, therefore, take an additively homomorphic encryption scheme for the underlying encryption and decryption scheme: We have $E(m_1 + m_2, r') = E(m_1, r_1)E(m_2, r_2)$ when $E(m, r)$ denote the encrypted message of m with randomizer r. We also assume that our encryption algorithm is semantically secure so that the permutation of the encrypted messages by shuffling party is not revealed. We let N be the modulus of the message space, whose smallest factor is comparatively larger than the number of messages, where $N = p^l q^m$ is a product of powers of two large primes $p < q$. As usual, p, q are one of the security factors of the scheme, so the probability of knowing a factorization of N is negligible. We let α be an integer such that $2^\alpha < p$, as Peng et al. did in [24], that is necessary for the arguments of the protocol security, since we use composite integers for the modulus of the message space.

We use $E(m)$ for the encryption of m with some random number when there is no need to specify the random number for the encryption. The re-encryption of a cipher-text c is denoted by $RE(c, r) = cE(0, r)$, and the decryption of a cipher text c is denoted by $D(c)$.

For the implementation of the main idea, we let $\widetilde{E}$ be another additively homomorphic encryption which use the same method of encryption for E (same message space and the ciphertext space) but with different parameters (for example, different bases of the ciphertext space in case of Paillier scheme). Then the corresponding decryption $\widetilde{D}$ is multiplicatively homomorphic:

$$\widetilde{D}(e_1 e_2) = \widetilde{D}(e_1) + \widetilde{D}(e_2)\,. \tag{4}$$

We make another assumption on E and $\widetilde{D}$ for simpler implementation:

$$\widetilde{D}(E(0,r)) = 0 \text{ for any } r\,. \tag{5}$$

Paillier Encryption Scheme

`Key generation` The modulus of the message space is $N = pq$ for two large primes $p < q$. A base $g \in \mathbb{Z}^*_{N^2}$ is an element of order $N\ell$ for some $\ell \in \{1, \dots, \lambda\}$, where λ is the least common multiple of $p-1$ and $q-1$.

`Encryption` For a message $m \in Z_N$, select a random $r \in \mathbb{Z}^*_N$. Then the ciphertext is computed by

$$c = g^m \cdot r^N \pmod{N^2}\,.$$

`Decryption` For a given ciphertext $c \in \mathbb{Z}^*_{N^2}$, the plaintext m is computed by

$$m = \frac{L(c^\lambda \bmod N^2)}{L(g^\lambda \bmod N^2)} \pmod{N}\,, \text{where}$$

L is a function defined as $L(u) = \frac{u-1}{N}$ for $u \in \mathbb{Z}^*_{N^2}$ such that $u = 1 \,(\mathrm{mod}\ N)$.

Following proposition is very basic but useful for us to prove the correctness of our protocol.

Proposition 1. *Let E be a Paillier encryption and D be the corresponding decryption. Then the followings are satisfied when $0 \in \mathbb{Z}_N$ is the additive identity and $1 \in \mathbb{Z}^*_{N^2}$ is the multiplicative identity:*

1. *$D(1) = 0$.*
2. *$D(c_1 c_2^{-1}) = D(c_1) - D(c_2)$ for all $c_1, c_2 \in G_2$.*

Proof. Since $E(0,1) = 1$ and the decryption is one-to-one, $D(1) = 0$ must hold. For any $c \in \mathbb{Z}^*_{N^2}$, $D(cc^{-1}) = D(1) = 0$ by the first part. On the other hand, $D(cc^{-1}) = D(c) + D(c^{-1})$ by (4). Therefore, we have $D(c^{-1}) = -D(c)$ and the second part follows. □

Useful properties of Paillier encryption scheme is stated in the following proposition. (See Lemma 5 in [22].)

Proposition 2. *The decrypted message of $c \in \mathbb{Z}^*_{N^2}$ is $0 \in \mathbb{Z}_N$ if and only if c is an Nth residue modulo N^2, that is $c = x^N \pmod{N^2}$ for some $x \in \mathbb{Z}^*_{N^2}$. Moreover, the property (5) is satisfied for any choice of bases for E and $\widetilde{D}$.*

2.2 Mix-Net and Shuffling

We let $\{m_1, m_2, \dots, m_n\}$ be the set of original messages and $\{c_1, c_2, \dots, c_n\}$, $c_i = E(m_i)$, $1 \le i \le n$, be a set of encrypted messages. Then a mix-net contains many rounds of shufflings defined as follows.

Shuffling party receives a set $\{c_1, c_2, \dots, c_n\}$ of encrypted messages from the previous shuffling party and outputs another set of encrypted messages $\{c'_1, c'_2, \dots, c'_n\}$ that is obtained as follows: for any i, $c'_i = RE(c_{\pi(i)}, r_i)$ for some permutation $\pi \in S_n$ and randomizers r_i.

A *proof of a shuffle* is a process to verify, without revealing any information, that a shuffling party did the shuffling in an honest way; that is *there is a permutation $\pi \in S_n$ such that $D(c'_i) = D(c_{\pi(i)})$*.

Assumption. We make an assumption that the *linear ignorance condition* for the set $\{m_1, \ldots, m_n\}$ of messages and the set $\{m_i m_j \mid 1 \leq i < j \leq n\}$ of products of messages are satisfied, where the linear ignorance condition is defined as follows. Linear ignorance condition is assumed in the first protocol of [24]:

Definition 1. *A set of messages $\{m_1, m_2, \ldots, m_n\}$ satisfies the linear ignorance condition if given a set of cipher-texts $\{c_1, c_2, \ldots, c_n\}$ of $\{m_1, m_2, \ldots, m_n\}$, the possibility for an adversary to find a non-trivial linear relation of $\{m_1, \ldots, m_n\}$ is negligible.*

Remark 1. We may drop the assumption on the linear ignorance condition if we use the method of the second protocol of Peng et al. in [24]. We do not deal with that matter in the present article though.

2.3 An Important Result by Peng et al.

The following proposition is proved in [24], and we state it for our later use. (See Lemmas 1, 2, 3 and 4 in [24].) For a given matrix A, we use A^t for the *transpose matrix of A*.

Proposition 3. *Suppose that $\{m_1, m_2, \ldots, m_n\}$ satisfies the linear ignorance condition, and let $\{c'_1, c'_2, \ldots, c'_n\}$ be the corresponding output of $\{c_1, c_2, \ldots, c_n\}$ by a shuffling party. For random numbers $s_1, s_2, \ldots, s_n$ from $\mathbb{Z}_N$, if the shuffling party can find $t_1, t_2, \ldots, t_n$ in $\mathbb{Z}_N$ such that*

$$\sum_i s_i D(c_i) = \sum_i t_i D(c'_i) \pmod N \tag{6}$$

with a probability larger than $2^{-\alpha}$, then the shuffling party can find an $n \times n$ invertible matrix P such that

$$[D(c'_1), D(c'_2), \ldots, D(c'_n)]^t = P\,[D(c_1), D(c_2), \ldots, D(c_n)]^t \pmod N, \tag{7}$$

$$[t_1, t_2, \ldots, t_n]^t = P^{-1}\,[s_1, s_2, \ldots, s_n]^t \pmod N. \tag{8}$$

Corollary 1. *The matrix P in Proposition 3 is unique.*

Proof. If there are two different matrices satisfying Equation (7), then one can find a non-trivial linear relation of $\{m_1, m_2, \ldots, m_n\}$. □

3 Verification Protocol

In this section, we describe our protocol and prove that an honest shuffling party always can pass the verification.

Suppose that a shuffling party receives $\{c_1, c_2, \ldots, c_n\}$ and the corresponding output is $\{c'_1, c'_2, \ldots, c'_n\}$. We let $m_i = D(c_i)$ and $m'_i = D(c'_i)$ for $i = 1, 2, \ldots, n$. We also suppose that $d_i = \widetilde{D}(c_i)$ and $d'_i = \widetilde{D}(c'_i)$ are published by an authorized party. The chosen basis for $\widetilde{E}$ and $\widetilde{D}$ does not have to be published since $\widetilde{D}$ is used only for the implementation of the protocol and we just need its multiplicatively homomorphic property:

Protocol

1. The verifier randomly chooses s_{ij} for $1 \leq i < j \leq n$ from $\{0, 1, \ldots, 2^\alpha - 1\}$ and publishes them.
2. The shuffling party shows, in a zero knowledge manner, that he knows t_{ij} for $1 \leq i < j \leq n$ and r_i, $i = 1, 2, \ldots, n$, such that, in the space of ciphertext,

$$\prod_i c'_j = \prod_i c_i E(0, r_i)\,, \tag{9}$$

$$\prod_{i<j} (c'_i c'_j)^{t_{ij}} = \prod_{i<j} (c_i c_j)^{s_{ij}} (E(0, r_i) E(0, r_j))^{t_{ij}}\,, \tag{10}$$

$$\prod_{i<j} ({c'}_i^{d'_j} {c'}_j^{d'_i})^{t_{ij}} = \prod_{i<j} (c_i^{d_j} c_j^{d_i})^{s_{ij}} (E(0, r_i)^{d'_j} E(0, r_j)^{d'_i})^{t_{ij}}\,. \tag{11}$$

Implementation. The same (non-interactive, zero-knowledge) implementation used in [24] (see section 3) can be adopted for the implementation of our protocol due to the fact that Equations (9), (10), (11) are essentially the same as the equations proved in [24] except the number of terms in each product.

Lemma 1. *If the shuffling party is honest, then he can pass the verification.*

Proof. Suppose $c'_i = RE(c_{\pi(i)}, r_i)$ are obtained using a permutation $\pi \in S_n$. Then by taking $t_{ij} = s_{\pi(i)\pi(j)}$ ($s_{\pi(j)\pi(i)}$ if $\pi(i) > \pi(j)$) the shuffling party can pass the verification: It is easy to check Equations (9) and (10) and we only check Equation (11). Observe first that $d'_i = d_{\pi(i)}$ since

$$d'_i = \widetilde{D}(c'_i) = \widetilde{D}(c_{\pi(i)E(0,r_i)}) = \widetilde{D}(c_{\pi(i)}) + \widetilde{D}(E(0, r_i)) = \widetilde{D}(c_{\pi(i)})\,,$$

where the last equality is from the assumption (5). Now we can finish the proof;

$$\begin{aligned}\prod_{i<j} \left({c'}_i^{d'_j} {c'}_j^{d'_i}\right)^{t_{ij}} &= \prod_{i<j} \left(c_{\pi(i)}^{d_{\pi(j)}} c_{\pi(j)}^{d_{\pi(i)}}\right)^{s_{\pi(i)\pi(j)}} \left(E(0, r_i)^{d'_j} E(0, r_j)^{d'_i}\right)^{t_{ij}} \\ &= \prod_{i<j} \left(c_i^{d_j} c_j^{d_i}\right)^{s_{ij}} \left(E(0, r_i)^{d'_j} E(0, r_j)^{d'_i}\right)^{t_{ij}}\,. \qquad \square\end{aligned}$$

4 Proof of the Correctness of Protocol

In this section, we prove that the proposed protocol is a correct verification *when we adopt Paillier encryption.*

Lemma 2. *When we assume Paillier encryption scheme, equations (9), (10) and (11) imply the following equations :*

$$\sum_i m_i = \sum_i m'_i \pmod N\,, \tag{12}$$

$$\sum_{i<j} s_{ij}(m_i + m_j) = \sum_{i<j} t_{ij}(m'_i + m'_j) \pmod{N}\,, \tag{13}$$

$$\sum_{i<j} s_{ij}(m_i m_j) = \sum_{i<j} t_{ij}(m'_i m'_j) \pmod{N}\,. \tag{14}$$

Proof. The modulus for the equations in the proof vary depending on which space we are working. The modulus is N if we are working on the space of messages, while the modulus must be N^2 if we are working on the space of ciphertexts; and we omit the modulus in the proof.

Equations (12) and (13) are immediate from the additively homomorphic property of our encryption scheme, and we only give a careful proof of Equation (14): Assume Equation (11) is true. Since E is additively homomorphic and $c_i = E(m_i), c'_i = E(m'_i)$ for $i = 1, \ldots, n$, we can rewrite Equation (11) as follows.

$$E\left(\sum_{i<j} s_{ij}(d_i m_j + d_j m_i)\right) = E\left(\sum_{i<j} t_{ij}(d'_i m'_j + d'_j m'_i)\right)$$

Encryption is always one-to-one and we obtain

$$\sum_{i<j} s_{ij}(d_i m_j + d_j m_i) = \sum_{i<j} t_{ij}(d'_i m'_j + d'_j m'_i)\,. \tag{15}$$

Since $d_i = \widetilde{D}(c_i), d'_i = \widetilde{D}(c'_i)$ for $i = 1, \ldots, n$, and $\widetilde{D}$ is multiplicatively homomorphic, we can rewrite Equation (15) as follows;

$$\widetilde{D}\left(\prod_{i<j} c_i^{s_{ij} m_j} c_j^{s_{ij} m_i}\right) = \widetilde{D}\left(\prod_{i<j} {c'}_i^{t_{ij} m'_j} {c'}_j^{t_{ij} m'_i}\right)\,.$$

Proposition 1 now implies that

$$\widetilde{D}\left(\frac{\prod_{i<j} c_i^{s_{ij} m_j} c_j^{s_{ij} m_i}}{\prod_{i<j} {c'}_i^{t_{ij} m'_j} {c'}_j^{t_{ij} m'_i}}\right) = 0\,,$$

and we obtain

$$\frac{\prod_{i<j} c_i^{s_{ij} m_j} c_j^{s_{ij} m_i}}{\prod_{i<j} {c'}_i^{t_{ij} m'_j} {c'}_j^{t_{ij} m'_i}} = x^N \text{ for some } x \in \mathbb{Z}_{N^2}^*$$

by Proposition 2. Let $x = E(y)$ for $y \in \mathbb{Z}_N$ then, again by the additively homomorphic property of E, we have

$$\sum_{i<j} s_{ij}(m_j m_i + m_i m_j) = \sum_{i<j} t_{ij}(m'_j m'_i + m'_i m'_j) + N \cdot y\,.$$

Note that $N \cdot y = 0 \pmod{N}$. We finally show that Equation (14) is true. □

Remark 2. We assume Paillier encryption in Lemma 2, where some special properties held in Paillier scheme (Proposition 2) are essentially used in the proof. In the following argument, we do not assume Paillier encryption. Additively homomorphic property of the encryption scheme and some special properties like Proposition 2 are used only for the zero-knowledge implementation of (12), (13) and (14). Therefore, if one can implement a protocol which guarantees (12), (13) and (14), then the arguments followed show that we have a correct verification with only two rounds of communications.

We let $[n] = \{1, 2, \ldots, n\}$ be the set of integers from 1 to n. Let $W(n)$ be the $n \times \binom{n}{2}$ matrix of 0's and 1's, the rows and columns of which are indexed by 1-subsets and 2-subsets of $[n]$ respectively;

$$(W(n))_{IJ} = \begin{cases} 1 & \text{if } I \subset J\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Throughout the rest of this article, let us fix an order on the set of 2-subsets of $[n]$ so that the first n of them are $\{1, 2\}, \{2, 3\}, \ldots, \{n-1, n\}$ and $\{n-2, n\}$. We always use the fixed order on the set of 1-subsets of $[n]$: $\{1\}, \{2\}, \ldots, \{n\}$.

The following proposition is the main tool that enables us to translate results on the collection of messages to the collection of pairs of messages and vice versa.

Proposition 4. *There is an n by n invertible matrix R over $\mathbb{Z}_N$ such that $RW(n) = [I_n \,|\, B] \pmod N$, where I_n is the $n \times n$ identity matrix, and B is an $n \times (\binom{n}{2} - n)$ matrix over $\mathbb{Z}_N$.*

Proof. Suppose that the rows and columns are indexed in the orders we fixed. Consider the series of row operations that changes the rows of $W(n)$ into $\mathbf{r}_1$, $\mathbf{r}_2 - \mathbf{r}_1$, $\mathbf{r}_3 - \mathbf{r}_2 + \mathbf{r}_1$, …, $\mathbf{r}_n - \mathbf{r}_{n-1} + \cdots + (-1)^{n-1}\mathbf{r}_1$, where $\mathbf{r}_i$ is the ith row of $W(n)$. It is easy to see that for every $i = 1, \ldots, n-1$, the $\{i, i+1\}$th column of the new matrix contains only one 1 at the ith row. Moreover, the $\{n-2, n\}$th column is $[0, \ldots, 0, 1, -1, 2]^t$. Note that 2 is a unit in $\mathbb{Z}_N$ since N is a product of powers of odd primes. This enables us to multiply the inverse of 2 to the last row and add it and its negative to the $(n-1)$st row and the $(n-2)$nd row, respectively. The composition of series of row operations we used makes R in the theorem. □

Remark 3. $W(n)$ is a special case of well known *incidence matrices* of t-subsets and k-subsets of $[n]$, where $t = 1, k = 2$. The rank of incidence matrices are known over $\mathbb{Z}_p$ for a prime p (see [31,11]): Our $W(n)$ has full rank n over $\mathbb{Z}_p$, $p \neq 2$. We, however, use $N = p^l q^m$ for the modulus and that is why we need to give a proof of Proposition 4.

In the rest of the article, every equation is modulo N, the modulus of the message space. In the following theorems and lemmas, we suppose that a shuffling party can provide proofs of Equations (9), (10) and (11) with a probability larger than $2^{-\alpha}$.

The following theorem is immediate from Proposition 3 since we are assuming the linear ignorance condition on $\{m_i m_j \,|\, 1 \leq i < j \leq n\}$. Remember that we fix an order of 2-subsets of $[n]$, and we follow the same order for s_{ij}'s, t_{ij}'s, $m_i m_j$'s and $m'_i m'_j$'s.

Theorem 1. *There is an invertible $\binom{n}{2} \times \binom{n}{2}$ matrix $\widetilde{P}$ over $\mathbb{Z}_N$ such that*

$$[m'_1 m'_2, m'_2 m'_3, \ldots]^t = \widetilde{P}\,[m_1 m_2, m_2 m_3, \ldots]^t \ \text{ and}$$
$$[t_{1\,2}, t_{2\,3}, \ldots]^t = (\widetilde{P})^{-1}\,[s_{1\,2}, s_{2\,3}, \ldots]^t\ .$$

Theorem 2. *The shuffling party knows an invertible $n \times n$ matrix P over $\mathbb{Z}_N$ such that*

$$[m'_1, m'_2, \ldots, m'_n]^t = P\,[m_1, m_2, \ldots, m_n]^t\ .$$

Proof. By Proposition 4, for random $s_1, s_2, \ldots, s_n$, one always can find s_{ij}'s in $\mathbb{Z}_N$ satisfying $\sum_{k\in\{i,j\}} s_{ij} = s_k$ for each k: we just have to solve $W(n)\,X = [s_1, \ldots, s_n]^t$ for $X = [x_{12}, x_{23}, \ldots]^t$. Therefore, the existence of P in the theorem is guaranteed by Proposition 3, since we assume the linear ignorance condition on $\{m_1, m_2, \ldots, m_n\}$. □

We state two Lemmas without proof which can be done by comparing coefficients of m_i's or $m_i m_j$'s in appropriate equations:

Lemma 3. *For any $i < j$ and $k < l$,*

$$\widetilde{P}_{\{ij\}\{kl\}} = P_{ik}P_{jl} + P_{il}P_{jk}\ . \tag{16}$$

Lemma 4. *When t_{ij}'s are the numbers provided by the shuffling party for given s_{ij}'s, the following equations are satisfied for any i,j and k.*

$$\sum_{k\in\{\alpha,\beta\}} s_{\alpha\beta} = \sum_{\alpha<\beta} (P_{\alpha k} + P_{\beta k})t_{\alpha\beta}\,, \tag{17}$$
$$(W(n)\widetilde{P})_{\{k\}\{i\,j\}} = P_{ik} + P_{jk}\ . \tag{18}$$

Theorem 3. *The matrix P given in Theorem 2 satisfies the following equations, and is a permutation matrix with high probability as a consequence.*

$$\text{For all } i,\ \sum_{\alpha=1}^{n} P_{i\alpha} = 1\,, \tag{19}$$
$$\text{for all } i<j \text{ and } k,\ P_{ik} + P_{jk} = P_{ik}\sum_{\alpha\neq k} P_{j\alpha} + P_{jk}\sum_{\alpha\neq k} P_{i\alpha}\ . \tag{20}$$

Proof. Equation (19) is immediate from Equation (12) and Theorem 2:

$$m_1 + \cdots + m_n = m'_1 + \cdots + m'_n = (\sum_\alpha P_{1\alpha})m_1 + \cdots + (\sum_\alpha P_{n\alpha})m_n\ .$$

For any $i < j$ and k, since

$$P_{ik} + P_{jk} = P_{ik}\sum_{\alpha\neq k} P_{j\alpha} + P_{jk}\sum_{\alpha\neq k} P_{i\alpha} = P_{ik}(1 - P_{jk}) + P_{jk}(1 - P_{ik})\,,$$

we have $2P_{ik}P_{jk} = 0$. In our case, this implies that either $P_{ik} = 0$ or $P_{jk} = 0$ since the factorization of modulus N is assumed to be very hard and, it is not known to the shuffling parties or N is a prime. Therefore, there can be at most one non-zero entry in each column of P. Since P is invertible, P can not have a zero column and each column must have exactly one non-zero entry, that is there are exactly n non-zero entries in P. Since there can be no zero row in P there must be exactly one non-zero entry in each row, and Equation (19) proves that each non-zero entry must be 1. □

Through the arguments of Theorem 1, Theorem 2 and Theorem 3, we have shown the following:

Theorem 4. *Suppose that a shuffling party can provide proofs of Equations (9), (10) and (11) with a probability larger than* $2^{-\alpha}$, *and assume the linear ignorance condition for the set* $\{m_1, m_2, \ldots, m_n\}$ *and* $\{m_i m_j \,|\, i < j\}$. *Then there is a permutation matrix* P *such that*

$$[m'_1, m'_2, \ldots, m'_n]^t = P\,[m_1, m_2, \ldots, m_n]^t \ .$$

5 Computationally Improved Protocol

In the previous sections, we proved that the proposed protocol is a valid verification that can be implemented with two communication rounds between the shuffling party and the verifier. But the proposed protocol has a significant drawback in computation complexity: The complexity of our protocol is $O(n^2)$ while the one by Peng et al. has $O(n)$ as its computational complexity. This is because we use all $\binom{n}{2}$ 2-subsets of $[n]$ for the verification. We, however, can restrict the number of nonzero s_{ij}'s so that we obtain a linear complexity: We must be careful to choose nonzero s_{ij}'s so that we do not lose the balance of i's, though. A good method to choose $\{i, j\}$'s for nonzero s_{ij}'s is to use known λ-*designs.* A λ-*design* is, in our case, a collection of 2-subsets of $[n]$ which contains exactly λ subsets containing i for each $i \in [n]$ (see [2]). Note that we can construct λ-designs easily as long as either n or λ is even; using all possible 2-subsets means we use an $(n-1)$-design.

By employing the idea of λ-designs, we still can keep the main idea of our protocol and get a reasonable computation cost also.

6 Computation Cost

The computation cost for a verification process depends on the method of implementation of zero-knowledge proofs and the encryption scheme. Since we can directly employ the implementation of (non-interactive) zero-knowledge proofs proposed in [24] for the proofs of (9), (10) and (11), we can calculate computation cost for the verification as Peng et al. did in [24]. There are more recent works other than [24], where better efficiencies has been achieved ([17,30]) with at least *three* rounds though. In [17], the comparison in efficiency has been made among verification protocols including the one by Peng et al.[24] and the one by Groth and Lu [17]. We, hence, compare the efficiency of our protocol with Peng et al.'s only.

In section 4 of [24], it is assumed that the cost of exponentiation with x-bit exponent is $1.5x$, and the cost of the product of n exponentiations with x-bit exponent is at most $n + 0.5nx$. Assuming that $\alpha = 20$ and N is 1024 bit number, the cost of computing d_i's and d_i''s is about $2n + \frac{n}{256}$ full length exponentiations since they are decryption processes in Paillier scheme. A rough (not sharp) upper bound for the cost of verification is $4\lambda n$ (full length exponentiations). Therefore, the overall cost for one shuffling with verification is about $(3 + 4\lambda)n$ if a λ-design is employed.

7 Final Remarks

Two conditions in Theorem 3 are sufficient for a matrix P to be a permutation matrix if the modulus is a prime as in the case of modified ElGamal. But, when the modulus is a composite integer they do not give sufficient conditions for P to be a permutation matrix. We, in Theorem 3, conclude P is a permutation matrix due to hardness of factorization of N. The following example shows that P doesn't have to be a permutation matrix without this assumption on the modulus.

Example 1. When $N = 1453 \cdot 3019 = 4386607$, the following non-permutation matrix satisfies the conditions in Theorem 3:

$$P = \begin{bmatrix} 271711 & 4114897 & 0 & 0 \\ 4114897 & 271711 & 0 & 0 \\ 0 & 0 & 271711 & 4114897 \\ 0 & 0 & 4114897 & 271711 \end{bmatrix}.$$

By defining $\widetilde{P}$ by $\widetilde{P}_{\{ij\}\{kl\}} = P_{ik}P_{jl} + P_{il}P_{jk}$, as in Equation (16) and providing t_{ij}'s calculated by $(\widetilde{P})^{-1}[s_{12}, s_{23}, \ldots]^t$, a shuffling party can pass the verification while passing contaminated messages: $[m_1', \ldots, m_n']^t = P[m_1, \ldots, m_n]^t$.

Remark 4. The matrix P in Example 1 can also pass all the verification of Peng et al. in [24]. Peng et al., however, did not give a correct explanation how this can happen in [24]: In their proof of the main theorem, they made a wrong reasoning by overlooking the fact that the message space may have composite modulus (the third paragraph in p. 197 of [24]). This can be treated though by taking the same argument we use for the proof of the correctness of our protocol.

References

1. Abe, M.: Mix-networks on permutation networks. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 258–273. Springer, Heidelberg (1999)
2. Cameron, P.J., van Lint, J.H.: Designs, graphs, codes and their links. London Mathematical Society Student Texts, vol. 22. Cambridge University Press, Cambridge (1991)

3. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–88 (1981)
4. Danezis, G.: Mix-networks with restricted routes. In: Dingledine, R. (ed.) PET 2003. LNCS, vol. 2760, pp. 1–17. Springer, Heidelberg (2003)
5. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a type iii anonymous remailer protocol. In: IEEE Symposium on Security and Privacy, pp. 2–15. IEEE Computer Society, Los Alamitos (2003)
6. Desmedt, Y.G., Kurosawa, K.: How to break a practical MIX and design a new one. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 557–572. Springer, Heidelberg (2000)
7. Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Dingledine and Syverson [10], pp. 54–68
8. Dingledine, R., Freedman, M.J., Hopwood, D., Molnar, D.: A reputation system to increase mix-net reliability. In: Moskowitz, I.S. (ed.) IH 2001. LNCS, vol. 2137, pp. 126–141. Springer, Heidelberg (2001)
9. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The second-generation onion router. In: USENIX Security Symposium, pp. 303–320. USENIX (2004)
10. Goldberg, I.: Privacy-enhancing technologies for the internet, II: Five years later. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 1–12. Springer, Heidelberg (2003)
11. Frankl, P.: Intersection theorems and mod p rank of inclusion matrices. J. Combin. Theory Ser. A 54(1), 85–94 (1990)
12. Furukawa, J., Sako, K.: An Efficient Scheme for Proving a Shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
13. Goh, E.-J.: Encryption Schemes from Bilinear Maps, Ph.D. thesis, Department of Computer Science, Stanford University (September 2007)
14. Golle, P., Jakobsson, M., Juels, A., Syverson, P.F.: Universal re-encryption for mixnets. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 163–178. Springer, Heidelberg (2004)
15. Golle, P., Zhong, S., Boneh, D., Jakobsson, M., Juels, A.: Optimistic mixing for exit-polls. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 451–465. Springer, Heidelberg (2002)
16. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2002)
17. Groth, J., Lu, S.: Verifiable Shuffle of Large Size Ciphertexts. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 377–392. Springer, Heidelberg (2007)
18. Mitomo, M., Kurosawa, K.: Attack for flash MIX. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 192–204. Springer, Heidelberg (2000)
19. Andrew Neff, C.: A verifiable secret shuffle and its application to e-voting. In: ACM Conference on Computer and Communications Security, pp. 116–125 (2001)
20. Nguyen, L., Safavi-Naini, R., Kurosawa, K.: Verifiable shuffles: A formal model and a paillier-based efficient construction with provable security. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 61–75. Springer, Heidelberg (2004)
21. Ogata, W., Kurosawa, K., Sako, K., Takatani, K.: Fault tolerant anonymous channel. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 440–444. Springer, Heidelberg (1997)
22. 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)

23. Park, C.-s., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/Nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
24. Peng, K., Boyd, C., Dawson, E.: Simple and efficient shuffling with provable correctness and ZK privacy. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 188–204. Springer, Heidelberg (2005)
25. Peng, K., Boyd, C., Dawson, E., Viswanathan, K.: A correct, private, and efficient mix network. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 439–454. Springer, Heidelberg (2004)
26. Pfitzmann, B., Pfitzmann, A.: How to break the direct RSA-implementation of mixes. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 373–381. Springer, Heidelberg (1990)
27. Pfitzmann, B., Schunter, M., Waidner, M.: How to break another "Provably secure" payment system. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 121–132. Springer, Heidelberg (1995)
28. Sagan, B.E.: The symmetric group, The Wadsworth & Brooks/Cole Mathematics Series, Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, CA, Representations, combinatorial algorithms, and symmetric functions (1991)
29. Serjantov, A., Danezis, G.: Towards an information theoretic metric for anonymity. In: Dingledine and Syverson [10], pp. 41–53
30. Wikström, D.: A sender verifiable mix-net and a new proof of a shuffle. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 273–292. Springer, Heidelberg (2005)
31. Wilson, R.M.: A diagonal form for the incidence matrices of t-subsets vs. k-subsets. European J. Combin. 11(6), 609–615 (1990)

On Formal Verification of Arithmetic-Based Cryptographic Primitives

David Nowak

Research Center for Information Security, AIST, Japan

Abstract. Cryptographic primitives are fundamental for information security: they are used as basic components for cryptographic protocols or public-key cryptosystems. In many cases, their security proofs consist in showing that they are reducible to computationally hard problems. Those reductions can be subtle and tedious, and thus not easily checkable. On top of the proof assistant Coq, we had implemented in previous work a toolbox for writing and checking game-based security proofs of cryptographic primitives. In this paper we describe its extension with number-theoretic capabilities so that it is now possible to write and check arithmetic-based cryptographic primitives in our toolbox. We illustrate our work by machine checking the game-based proofs of unpredictability of the pseudo-random bit generator of Blum, Blum and Shub, and semantic security of the public-key cryptographic scheme of Goldwasser and Micali.

Keywords: machine formalization, cryptographic primitives, CSPRBG, semantic security.

1 Introduction

Cryptographic primitives are fundamental components for information security. In many cases, their security proofs consist in showing that they are reducible to computationally hard problems. Those reductions can be subtle and tedious, and thus not easily checkable. Bellare and Rogaway even claim in [4] that:

> "*many proofs in cryptography have become essentially unverifiable. Our field may be approaching a crisis of rigor.*"

As a remedy, they, and also Shoup [15], advocate game-based security proofs. This is a methodology for writing proofs which makes them easier to read and check. Halevi goes further by advocating the need for a software which can deal with the mundane parts of writing and checking game-based proofs [10].

In the game-based approach, a security property is modeled as a probabilistic program which implements a game to be solved by the attacker. The attacker itself is modeled as an external probabilistic procedure interfaced with the game. The goal is then to prove that any attacker has at most a negligible advantage over a random player. An attacker is assumed to be efficient i.e., it is modeled as a probabilistic polynomial-time (PPT) algorithm.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 368–382, 2009.

Related Work. There are tools such as ProVerif [5], CryptoVerif [6] or the prototype implementation of [12] which can make automatic proofs of cryptographic protocols or generic cryptographic schemes. However those tools assume that some secure cryptographic primitives are given. The security of those primitives cannot be proved automatically. Nevertheless their security proofs can be checked by a computer.

The game-based proof of the PRP/PRF switching lemma has been formalized in the proof assistant Coq [1]. Although it is not by itself a cryptographic primitive, this lemma is fundamental in proving security of some cryptographic schemes. The proof has been made in the random oracle model. The machine formalization in [1] is a so-called *deep embedding*: games are syntactic objects; and game transformations are syntactic manipulations which can be automated in the language of the proof assistant. The main advantages of this approach are that one can prove completeness of decision procedures, if any, and get smaller proof terms. However those advantages are not exploited in [1]. Moreover this is at the cost of developing a huge machinery for syntactic manipulations. Two other deep embeddings are currently being developed [2,3].

Previous Work. In cryptographers' papers, the formal semantics of games is either left implicit or, at best, informally explained in English. It is not enough for machine formalization. In previous work, we have (1) proposed a formal semantics for games, (2) implemented it in the proof assistant Coq, and (3) used it to prove the semantic security of the ElGamal and Hashed ElGamal public-key cryptographic schemes [13]. Our machine formalization is a so-called *shallow embedding*: games are probability distributions (as advocated by Shoup [15]). Game transformations can still be automated by going through the metalanguage of the proof assistant. Compared to [1], We have been very careful in making our design choices such that our implementation remains light. This is an important design issue in formal verification because formal proofs grow quickly in size when one tackles real-world use-cases.

Our toolbox comes in two layers. The first layer extends the standard library of Coq with mathematical notions and their properties that are fundamental in cryptography but not available in the standard library of Coq. This consists of a library for probability distributions, bitstrings, and a small library for elementary group theory. On top of this, the second layer consists of formal versions of security definitions and hard problems, and basic game transformations which can be composed to reduce the security of cryptographic primitives to hard problems.

By using our toolbox, one is forced to exhibit all the steps in his or her game-based proof and thus cannot hide assumptions or make proofs by intimidation such as "*Trivial*" or "*The reader may easily supply the details*". In spite of the required level of detail, proofs remain human readable and human checkable.

Our contributions. We have extended our toolbox in order to be able to deal with cryptographic primitives based on number theory. For that purpose we have added to the first layer a library of definitions and lemmas for integers modulo

n. In particular we have formalized the notions of Legendre and Jacobi symbol, Blum primes, and their properties which are of fundamental use in cryptography. This is already by itself a contribution to the theorem proving community.[1] We have also considerably extended and generalized our library on elementary group theory.[2]

Then we have used our extension to the first layer in adding to the second layer the security notion of unpredictability, the quadratic residuosity assumption, and number-theoretic game transformations.

Finally, we have used our extensions to machine check the proof of unpredictability of the pseudo-random bit generator of Blum, Blum and Shub [7]. We have also machine checked the proof of semantic security of the public-key cryptographic scheme of Goldwasser and Micali [8]. Our security proofs of those two primitives are based on the intractability of the quadratic residuosity problem. To the best of our knowledge, this is the first time that security proofs of cryptographic primitives based on number theory are machine checked. This is also the first time that a proof of unpredictability is machine checked. None of the above mentioned related work could be used in their current state to formalize such proofs because they are missing components for number theory.

Outline. We introduce the proof assistant Coq in Sect. 2. In Sect. 3, we give formal meaning to games in terms of distributions. We give in Sect. 4 the number-theoretic facts that we use Sect. 5. In Sect. 5.1 we apply our work to the proof of unpredictability of the Blum-Blum-Shub generator, and in Sect. 5.2 we apply it to the proof of semantic security of the Goldwasser-Micali scheme. Finally, we briefly describe our implementation in Sect. 6 before concluding in Sect. 7.

2 The Coq Proof Assistant

Coq is a proof assistant developed at INRIA since 1984.[3] It is based on a kernel which takes a mathematical statement S and proof term p as input and check whether p is a correct proof of S. On top of this kernel there are: a tactic language which allows to build proof terms in an incremental way; and decision procedures for decidable fragments such as Presburger arithmetic or propositional logic.

Coq is goal-directed. This means that if we are trying to prove that a formula Q (the goal) is true, and we have an already proved theorem stating that P_1 & P_2 implies Q, then we can apply this theorem. Coq will replace the goal Q by two subgoals P_1 and P_2. Proofs by induction are also possible. We proceed this way until we finally reach goals that are either axioms or are true by definition. On the way, Coq builds incrementally a proof term to be checked later by the kernel.

[1] Tools such as *Mathematica* can deal with formal computations involving Legendre and Jacobi symbols, but cannot be used to make formal proofs. In particular they do not allow reasoning by induction.

[2] There are more advanced library on group theory, such as [9], but none of them are available with the version of Coq that we are using (8.2beta3).

[3] `http://coq.inria.fr/`

$$\textbf{return}\ a \;=\; \{\!|(a,1)|\!\} \qquad\qquad \begin{array}{l} x \Leftarrow \delta; \\ \varphi(x) \end{array} \;=\; \bigcup_{(a,p)\in\delta} p\cdot\varphi(a)$$

$$\begin{array}{l} x \overset{R}{\leftarrow} \{a_1,\dots,a_n\}; \\ \varphi(x) \end{array} \;=\; \begin{array}{l} x \Leftarrow \{\!|(a_1,\frac{1}{n}),\dots,(a_n,\frac{1}{n})|\!\}; \\ \varphi(x) \end{array}$$

$$\begin{array}{l} x \leftarrow a; \\ \varphi(x) \end{array} \;=\; \begin{array}{l} x \overset{R}{\leftarrow} \{a\}; \\ \varphi(x) \end{array}$$

Fig. 1. Notations for distributions

The kernel is the only critical part: if a bug outside the kernel causes a wrong proof term to be built, it will be rejected by the kernel.

In order to be closer to mathematical practice, Coq also provides mechanisms for introducing notations, or for inferring implicit parameters and subset coercions. It also comes with a standard library of definitions and lemmas, for instance on elementary arithmetic, analysis or polymorphic lists.

3 Games

We denote a game by a finite probability distribution (from now on, we will abbreviate this term as *distribution*). A distribution δ over a set S is defined as a finite multiset[4] of ordered pairs from $S\times\mathbb{R}$ such that $\sum_{(a,p)\in\delta} p \;=\; 1$. We use the symbols $\{\!|$ and $|\!\}$ as delimiters of mutisets not to confuse them with sets. We write $p\cdot\{\!|(a_1,p_1),\dots,(a_n,p_n)|\!\}$ for the multiset $\{\!|(a_1,p\cdot p_1),\dots,(a_n,p\cdot p_n)|\!\}$.

For convenience, we introduce some notations (cf. Fig. 1) for writing distributions so that they will look like probabilistic programs:

- We write **return** a for the distribution with only the element a of probability 1.
- We write $x \Leftarrow \delta;\ \varphi(x)$ for the distribution built by picking at random a value x according to the distribution δ and then computing the distribution $\varphi(x)$.
- We write $x \overset{R}{\leftarrow} \{a_1,\dots,a_n\};\ \varphi(x)$ for the distribution built by picking at random a value x in the set $\{a_1,\dots,a_n\}$ and then computing the distribution $\varphi(x)$.
- We abbreviate this last case by $x \leftarrow a;\ \varphi(x)$ when the set is a singleton $\{a\}$.

Distributions have a monadic structure [14] and thus satisfy the monad laws (cf. Fig. 2). Those laws state that our notations for distributions behave well. The first one simply states that the occurences of a deterministically assigned

[4] A multiset (a.k.a. a bag) is a generalization of a set: a member of a multiset may be member more that once. For example, the multisets $\{\!|1,2,2|\!\}$ and $\{\!|1,2|\!\}$ are different; and the union of $\{\!|1,2,2,3|\!\}$ and $\{\!|1,4,4|\!\}$ is equal to $\{\!|1,1,2,2,3,4,4|\!\}$.

$$\begin{array}{l} x \leftarrow a; \\ \varphi(x) \end{array} = \varphi(a) \qquad \begin{array}{l} x \Leftarrow \delta; \\ \mathbf{return}\ x \end{array} = \delta \qquad \begin{array}{l} y \Leftarrow (\\ \quad x \Leftarrow \delta; \\ \quad \varphi(x) \\ ); \\ \psi(y) \end{array} = \begin{array}{l} x \Leftarrow \delta; \\ y \Leftarrow \varphi(x); \\ \psi(y) \end{array}$$

Fig. 2. Monad laws

variable x can be replaced by their definition (constant propagation). The second one is a kind of η-reduction. The third one allows to simplify nested sequences.

We write $\mathsf{Pr}\left(P(\delta)\right)$ for the probability that P holds of an element picked at random in the distribution δ. Its value is

$$\sum_{(a,p)\in\delta \text{ s.t. } P(a)} p$$

We define a notion of indistinguishability for distributions. Security definitions, hard problems and game transformations will all be defined with this relation. Two distributions δ_1 and δ_2 are indistinguishable modulo ϵ w.r.t. a predicate P, written $\delta_1 \equiv_{\epsilon}^{P} \delta_2$, iff:

$$\left|\mathsf{Pr}\left(P(\delta_1)\right) - \mathsf{Pr}\left(P(\delta_2)\right)\right| \leq \epsilon$$

$\equiv_{\epsilon}^{P}$ is reflexive and symmetric. If $\delta_1 \equiv_{\epsilon}^{P} \delta_2$ and $\delta_2 \equiv_{\epsilon'}^{P} \delta_3$, then $\delta_1 \equiv_{\epsilon+\epsilon'}^{P} \delta_3$. If $\delta_1 \equiv_{\epsilon}^{P} \delta_2$ and $\epsilon \leq \epsilon'$, then $\delta_1 \equiv_{\epsilon'}^{P} \delta_2$. We write $\equiv_{\epsilon}$ instead of $\equiv_{\epsilon}^{P}$ when P is the predicate on booleans such that $P(b)$ holds iff b is equal to *true*.

Lemma 3.1. *If $f : S \to T$ is a bijection then, for all φ and P,*

$$\begin{array}{l} x \xleftarrow{R} S; \\ \varphi(f(x)) \end{array} \equiv_0^P \begin{array}{l} x \xleftarrow{R} T; \\ \varphi(x) \end{array}$$

It is also true when f is a surjective N-to-one function.

This kind of transformation of one game into another one is at the crux of the security proofs we are dealing with.

4 Some Elementary Number Theory

Let n be a positive number. We write $\mathbb{Z}_n$ for the set of integers modulo n. The multiplicative group of $\mathbb{Z}_n$ is written $\mathbb{Z}_n^*$ and consists of the subset of integers modulo n which are coprime with n. An integer $x \in \mathbb{Z}_n^*$ is a quadratic residue modulo n iff there exists a $y \in \mathbb{Z}_n^*$ such that $y^2 \equiv x \pmod{n}$. Such a y is called a square root of x modulo n. We write QR_n for the set of quadratic residues modulo n, and QNR_n for its complement i.e., the set of quadratic non-residues modulo n. We write $\mathbb{Z}_n^*(+1)$ (respectively, $\mathit{QNR}_n(+1)$) for the subset of integers in $\mathbb{Z}_n^*$ (respectively, QNR_n) with Jacobi symbol equal to 1.

The quadratic residuosity problem is the following: given an odd composite integer n, decide whether or not an $x \in \mathbb{Z}_n^*$ is a quadratic residue modulo n.

Let n be the product of two distinct odd primes p and q. The quadratic residuosity assumption (QRA) states that the above problem is intractable. In our framework, this can be stated as:

Assumption 4.1 (QRA). *For every attacker A', there exists a negligible ϵ such that*

$$\begin{array}{l} x \overset{R}{\leftarrow} \mathbb{Z}_n^*(+1); \\ \widehat{b} \Leftarrow A'(n, x); \\ b \leftarrow \widehat{b} = \mathsf{qr}(x); \\ \textbf{return } b \end{array} \quad \equiv_\epsilon \quad \begin{array}{l} b \overset{R}{\leftarrow} \{true, false\}; \\ \textbf{return } b \end{array}$$

In the left-side game, an x is picked at random in the set $\mathbb{Z}_n^*(+1)$; this x is passed with n to the attacker A'; the attacker returns its guess $\widehat{b}$ for the quadratic residuosity (modulo n) of x; this guess is compared with the true quadratic residuosity (modulo n) $\mathsf{qr}(x)$ of x; and the result b of this comparison is returned. In the rigth-side game, the result is random. QRA states that the advantage ϵ of any attacker over a random player is negligible.

Note that the fact that A' is a randomized algorithm is modeled by the attacker returning a distribution in which $\widehat{b}$ is picked.

In the security proofs, we will need the following well-known mathematical facts (remember that n is the product of two distinct odd primes p and q):

Fact I. The function which maps an $x \in \mathbb{Z}_n^*$ to $x^2 \in QR_n$ is a surjective four-to-one function.

Fact II. For any $y \in QNR_n(+1)$, the function which maps an $x \in QR_n$ to $y \cdot x \in QNR_n(+1)$ is a bijection.

Fact III. $|QR_n| = |QNR_n(+1)|$.

Fact IV. $\mathbb{Z}_n^*(+1) \ = \ QR_n \cup QNR_n(+1)$

Let n be a Blum integer i.e., the product of two distinct prime numbers p and q, each congruent to 3 modulo 4. In this case, any $x \in QR_n$ has a unique square root in QR_n which we denote by $\sqrt{x}$ and is called the principal square root of x. And we get the following additional facts [7]:

Fact V. The function which maps an $x \in QR_n$ to $x^2 \in QR_n$ is a permutation.

Fact VI. The function which maps an $x \in \mathbb{Z}_n^*(+1)$ to $x^2 \in QR_n$ is a surjective two-to-one function.

Fact VII. For all $x \in QR_n$, $\quad \sqrt{x^2} = x$

Fact VIII. For all $x \in \mathbb{Z}_n^*(+1)$, $\quad x \in QR_n \ \Leftrightarrow \ \mathsf{parity}(x) = \mathsf{parity}(\sqrt{x^2})$

```
bbs($len \in \mathbb{N}, seed \in \mathbb{Z}_n^*$) =
  bbs_rec($len, seed^2$)

bbs_rec($len \in \mathbb{N}, x \in QR_n$) =
  match $len$ with
  |$0 \Rightarrow []$
  |$len' + 1 \Rightarrow$ parity($x$) :: bbs_rec($len', x^2$)
  end
```

Fig. 3. The Blum-Blum-Shub generator

5 Applications

In this section, we apply our work to the proofs of unpredictability of the Blum-Blum-Shub generator, and to the proof of semantic security of the Goldwasser-Micali scheme.

5.1 The Blum-Blum-Shub Pseudorandom Bit Generator

The security of many cryptographic systems depends upon a cryptographic primitive for the generation of unpredictable sequences of bits. They are used to generate keys, nonces or salts. Ideally, those sequences of bits should be random, that is, generated by successive flips of a fair coin. In practice, one uses a pseudorandom bit generator (PRBG) which, given a short seed, generates a long sequence of bits that appears random. For the purpose of simulation, one only requires of a PRBG that it passes certain statistical tests (cf. Chapter 3 of [11]). This is not enough for cryptography. A PRBG is cryptographically secure iff it passes all polynomial-time statistical tests: roughly speaking, no polynomial-time algorithm can distinguish between an output sequence of the generator and a truly random sequence.

In [7], the Blum-Blum-Shub generator (BBS) is proved left-unpredictable (under the quadratic residuosity assumption). It was proved by Yao in [17] that this is equivalent to stating that BBS passes all polynomial-time statistical tests. It is shown in [16] that BBS is still secure under the weaker assumption that n is hard to factorize. The same authors also show that, for sufficiently large n, more than one bit can be extracted at each iteration of the algorithm. However, in this paper, we stick to the original proof of [7].

Let $n = p \cdot q$ be a Blum integer. The BBS generator is defined by the function bbs given in Fig. 3 which takes as input a length and a seed, and returns a pseudorandom sequence of bits of the required length.

In our framework, one can state the left-unpredictability of bbs by the following definition.

Definition 5.1 (Left-unpredictability). bbs *is left-unpredictable iff for all length len, for every attacker A, there exists a negligible ϵ such that*

$$\begin{array}{l} seed \xleftarrow{R} \mathbb{Z}_n^*; \\ [b_0, \ldots, b_{len}] \leftarrow \mathsf{bbs}(len + 1, seed); \\ \widehat{b_0} \Leftarrow A([b_1, \ldots, b_{len}]); \\ b \leftarrow \widehat{b_0} = b_0; \\ \textbf{return } b \end{array} \equiv_\epsilon \begin{array}{l} b \xleftarrow{R} \{true, false\}; \\ \textbf{return } b \end{array}$$

In the left-side game, a *seed* is picked at random in the set $\mathbb{Z}_n^*$; the function bbs is then used to compute a pseudorandom sequence of bits $[b_0, \ldots, b_{len}]$ of length $len+1$; this sequence minus its first bit b_0 is passed to the attacker A; the attacker returns its guess $\widehat{b_0}$ for the value of the bit b_0; this guess is compared with b_0; and the result b of this comparison is returned. bbs is left-unpredictable if the advantage ϵ of any attacker over a random player is negligible.

Before proving that bbs is left-unpredictable, we show that it can be reduced to the problem of finding the parity of a random quadratic residue modulo n.

Lemma 5.2. *If, for every attacker A', there exists a negligible ϵ such that*

$$\begin{array}{l} x \overset{R}{\leftarrow} QR_n; \\ \widehat{b} \Leftarrow A'(n, x); \\ b \leftarrow \widehat{b} = \mathsf{parity}(\sqrt{x}); \\ \textbf{return } b \end{array} \quad \equiv_\epsilon \quad \begin{array}{l} b \overset{R}{\leftarrow} \{true, false\}; \\ \textbf{return } b \end{array}$$

then bbs *is left-unpredictable.*

In the left-side game, x is picked at random in the set QR_n; this x is passed with n to the attacker A'; the attacker returns its guess $\widehat{b}$ for the parity of $\sqrt{x}$; this guess is compared with the true parity of $\sqrt{x}$; and the result b of this comparison is returned. The above lemma states that if the advantage ϵ of any attacker over a random player is negligible, then bbs is left-unpredictable.

Proof (of Lemma 5.2). We proceed by rewriting the left-side game of the left-unpredictability specification (Def. 5.1).

BBS1. We unfold the definition of bbs:

$$\begin{array}{l} seed \overset{R}{\leftarrow} \mathbb{Z}_n^*; \\ [b_0, \ldots, b_{len}] \leftarrow \mathsf{bbs\_rec}(len+1, seed^2); \\ \widehat{b_0} \Leftarrow A([b_1, \ldots, b_{len}]); \\ b \leftarrow \widehat{b_0} = b_0; \\ \textbf{return } b \end{array}$$

BBS2. Because of Fact I, we can rewrite the game as:

$$\begin{array}{l} x \overset{R}{\leftarrow} QR_n; \\ [b_0, \ldots, b_{len}] \leftarrow \mathsf{bbs\_rec}(len+1, x); \\ \widehat{b_0} \Leftarrow A([b_1, \ldots, b_{len}]); \\ b \leftarrow \widehat{b_0} = b_0; \\ \textbf{return } b \end{array}$$

BBS3. x is a quadratic residue, we can thus replace x with $\sqrt{x^2}$ (according to Fact VII).

$$\begin{array}{l} x \overset{R}{\leftarrow} QR_n; \\ [b_0, \ldots, b_{len}] \leftarrow \mathsf{bbs\_rec}(len+1, \sqrt{x^2}); \\ \widehat{b_0} \Leftarrow A([b_1, \ldots, b_{len}]); \\ b \leftarrow \widehat{b_0} = b_0; \\ \textbf{return } b \end{array}$$

BBS4. Because of Fact V, we can rewrite the game as:

```
$x \overset{R}{\leftarrow} QR_n;$
$([b_0, \ldots, b_{len}]) \leftarrow \mathsf{bbs\_rec}(len + 1, \sqrt{x});$
$\widehat{b_0} \Leftarrow A([b_1, \ldots, b_{len}]);$
$b \leftarrow \widehat{b_0} = b_0;$
$\mathbf{return}\ b$
```

BBS5. By unfolding one step of bbs_rec, we get:

```
$x \overset{R}{\leftarrow} QR_n;$
$[b_1, \ldots, b_{len}] \leftarrow \mathsf{bbs\_rec}(len, x);$
$\widehat{b_0} \Leftarrow A([b_1, \ldots, b_{len}]);$
$b \leftarrow \widehat{b_0} = \mathsf{parity}(\sqrt{x});$
$\mathbf{return}\ b$
```

BBS6. We have reduced the game to the left-side one of the hypothesis where the attacker $A'(n, x)$ is instantiated by:

```
$[b_1, \ldots, b_{len}] \leftarrow \mathsf{bbs\_rec}(len, x);$
$\widehat{b_0} \Leftarrow A([b_1, \ldots, b_{len}]);$
$\mathbf{return}\ \widehat{b_0}$
```

□

Using the above lemma, we can now prove that bbs is left-unpredictable.

Theorem 5.3. bbs *is left-unpredictable (under the quadratic residuosity assumption).*

Proof. By Lemma 5.2, we only need to prove that for every attacker A:

$$\begin{array}{l} x \overset{R}{\leftarrow} QR_n; \\ \widehat{b} \Leftarrow A(n, x); \\ b \leftarrow \widehat{b} = \mathsf{parity}(\sqrt{x}); \\ \mathbf{return}\ b \end{array} \quad \equiv_\epsilon \quad \begin{array}{l} b \overset{R}{\leftarrow} \{true, false\}; \\ \mathbf{return}\ b \end{array}$$

We proceed by rewriting the left-side game.

BBS7. Because of Fact VI, we can rewrite the game as:

```
$x \overset{R}{\leftarrow} \mathbb{Z}_n^*(+1);$
$\widehat{b} \Leftarrow A(n, x^2);$
$b \leftarrow \widehat{b} = \mathsf{parity}(\sqrt{x^2});$
$\mathbf{return}\ b$
```

BBS8. By Fact VIII, we can replace the equality test $\widehat{b} = \mathsf{parity}(\sqrt{x^2})$ by $\widehat{b} \oplus \mathsf{parity}(x) \oplus 1 = \mathsf{qr}(x)$ (where $\oplus$ is the notation for the exclusive-or XOR).

```
$x \overset{R}{\leftarrow} \mathbb{Z}_n^*(+1);$
$\widehat{b} \Leftarrow A(n, x^2);$
$b \leftarrow \widehat{b} \oplus \mathsf{parity}(x) \oplus 1 = \mathsf{qr}(x);$
$\mathbf{return}\ b$
```

```
keygen() =
  $pk \leftarrow (n, y)$;
  $sk \leftarrow (p, q)$;
  return $(pk, sk)$
```

```
encrypt($(n, y) \in \mathbb{Z} \times \mathbb{Z}_n^*, b \in \{0, 1\}$) =
  $x \xleftarrow{R} \mathbb{Z}_n^*$;
  $c \leftarrow$ (if $b = 1$ then $y \cdot x^2$ else $x^2$);
  return $c$
```

```
decrypt($(p, q) \in \mathbb{Z} \times \mathbb{Z}, c \in \mathbb{Z}_n^*$) =
  $e \leftarrow \left(\frac{c}{p}\right)$;
  $m \leftarrow$ (if $e = 1$ then $0$ else $1$);
  return $m$
```

Fig. 4. The Goldwasser-Micali scheme

BBS9. We have reduced the game to the left-sided one of QRA (Assumption 4.1) where the attacker $A'(n, x)$ is instantiated by:

```
$\widehat{b} \Leftarrow A(n, x^2)$;
return $\widehat{b} \oplus \mathsf{parity}(x) \oplus 1$
```

□

5.2 The Goldwasser-Micali Public-Key Cryptographic Scheme

The Goldwasser-Micali public-key cryptographic scheme (GM) was the first probabilistic one which was provably secure. More precisely it is semantically secure under the quadratic residuosity assumption [8]. For defining GM, we need a number n which is the product of two distinct prime numbers p and q, and a $y \in QNR_n(+1)$. It is then defined by the three functions given in Fig. 4 where $\left(\frac{c}{p}\right)$ denotes the Legendre symbol of c.

In our framework, one can state the semantic security of the above scheme by the following definition.

Definition 5.4. *GM is semantically secure iff, for every attacker (A_1, A_2), there exists a negligible ϵ such that*

```
$(pk, sk) \Leftarrow$ keygen();
$(m_1, m_2) \Leftarrow A_1(pk)$;
$i \xleftarrow{R} \{1, 2\}$;                          $\equiv_\epsilon$   $b \xleftarrow{R} \{true, false\}$;
$c \Leftarrow$ encrypt$(pk, m_i)$;                                return $b$
$\widehat{\imath} \Leftarrow A_2(pk, (m1, m2), c)$;
return $\widehat{\imath} = i$
```

In the left-side game, a pair (pk, sk) of public and secret keys is generated; the public key pk is passed to the attacker A_1 which returns two messages m_1 and m_2; one of them is picked at random and encrypted with the secret key sk; the obtained cyphertext c is then passed with the public key pk and the pair of picked messages (m_1, m_2) to the attacker A_2; the attacker returns its guess for the picked message; whether the attacker is right or not is returned as a result. A scheme is semantically secure if the advantage ϵ of any attacker over a random player is negligible.

Theorem 5.5. *The scheme of Goldwasser and Micali is semantically secure (under the quadratic residuosity assumption).*

Proof. We proceed by rewriting the left-side game of the semantic-security specification (Def. 5.4).

GM1. We unfold definitions of keygen and encrypt:

```
$(m_1, m_2) \Leftarrow A_1(n, y);$
$i \xleftarrow{R} \{1, 2\};$
$x \xleftarrow{R} \mathbb{Z}_n^*;$
$\widehat{\imath} \Leftarrow A_2((n, y), (m_1, m_2), \textbf{if } m_i = 1 \textbf{ then } y \cdot x^2 \textbf{ else } x^2);$
$\textbf{return } \widehat{\imath} = i$
```

GM2. Because of Fact I, we can rewrite the game as:

```
$(m_1, m_2) \Leftarrow A_1(n, y);$
$i \xleftarrow{R} \{1, 2\};$
$x \xleftarrow{R} QR_n;$
$\widehat{\imath} \Leftarrow A_2((n, y), (m_1, m_2), \textbf{if } m_i = 1 \textbf{ then } y \cdot x \textbf{ else } x);$
$\textbf{return } \widehat{\imath} = i$
```

GM3. Because of Fact II, we can rewrite the game as:

```
$(m_1, m_2) \Leftarrow A_1(n, y);$
$i \xleftarrow{R} \{1, 2\};$
$x \xleftarrow{R} QR_n;$
$z \xleftarrow{R} QNR_n(+1);$
$\widehat{\imath} \Leftarrow A_2((n, y), (m_1, m_2), \textbf{if } m_i = 1 \textbf{ then } z \textbf{ else } x);$
$\textbf{return } \widehat{\imath} = i$
```

Note that this transformation is only valid because the result of the game does not depend on the relation between $y \cdot x$ and x. Indeed, if $m_i = 1$ then only $y \cdot x$ is used while x can be ignored, and vice versa.

Now we consider the different cases for the messages m_1 and m_2 chosen by the attacker.

(i) $(m_1, m_2) = (0, 0)$:

GM4. We can rewrite the game as:

```
$x \xleftarrow{R} QR_n;$
$z \xleftarrow{R} QNR_n(+1);$
$\widehat{\imath} \Leftarrow A_2((n, y), (0, 0), x);$
$i \xleftarrow{R} \{1, 2\};$
$\textbf{return } \widehat{\imath} = i$
```

i can be picked randomly after the calls to the attacker. Therefore $\widehat{\imath}$ does not depend on i. Our goal is proved.

(ii) $(m_1, m_2) = (1, 1)$: This is similar to the previous case except that A_2 is given z instead of x.

(iii) $(m_1, m_2) = (0, 1)$:

GM5. We can rewrite the game GM3 as:

```
$i \xleftarrow{R} \{1, 2\};$
if $i = 1$ then
    $x \xleftarrow{R} QR_n;$
    $\widehat{\imath} \Leftarrow A_2((n, y), (0, 1), x);$
    return $\widehat{\imath} = i$
else
    $z \xleftarrow{R} QNR_n(+1);$
    $\widehat{\imath} \Leftarrow A_2((n, y), (0, 1), z);$
    return $\widehat{\imath} = i$
```

GM6. By definition of qr, we can rewrite the game as:

```
$i \xleftarrow{R} \{1, 2\};$
if $i = 1$ then
    $x \xleftarrow{R} QR_n;$
    $\widehat{\imath} \Leftarrow A_2((n, y), (0, 1), x);$
    $\widehat{b} \leftarrow \widehat{\imath} = 1;$
    return $\widehat{b} = \mathsf{qr}(x)$
else
    $z \xleftarrow{R} QNR_n(+1);$
    $\widehat{\imath} \Leftarrow A_2((n, y), (0, 1), z);$
    $\widehat{b} \leftarrow \widehat{\imath} = 1;$
    return $\widehat{b} = \mathsf{qr}(z)$
```

GM7. Because of Fact III, we can rewrite the game as:

```
$x \xleftarrow{R} QR_n \cup QNR_n(+1);$
$\widehat{\imath} \Leftarrow A_2((n, y), (0, 1), x);$
$\widehat{b} \leftarrow \widehat{\imath} = 1;$
return $\widehat{b} = \mathsf{qr}(x)$
```

GM8. Because of Fact IV, we can rewrite the game as:

```
$x \xleftarrow{R} \mathbb{Z}_n^*(+1);$
$\widehat{\imath} \Leftarrow A_2((n, y), (0, 1), x);$
$\widehat{b} \leftarrow \widehat{\imath} = 1;$
return $\widehat{b} = \mathsf{qr}(x)$
```

GM9. We have reduced the game to the left-side one of QRA (Assumption 4.1) where the attacker $A'(n, x)$ is instantiated by:

```
$\widehat{\imath} \Leftarrow A_2((n, y), (0, 1), x)$;
$\widehat{b} \leftarrow \widehat{\imath} = 1$;
return $\widehat{b}$
```

(iv) $(m_1, m_2) = (1, 0)$: This case is similar to the previous one. □

6 Implementation

We have extended our toolbox with a module which contains number-theoretic lemmas on Legendre and Jacobi symbols, and on Blum integers. Based on those lemmas, we have proved arithmetic-based game transformations in the module dedicated to transformations. We have added the definition of unpredictability and the quadratic residuosity assumption in the appropriate modules. We have then used those extensions to make the formal security proofs of the Blum-Blum-Shub generator and the Goldwasser-Micali scheme.

Future work. Two standard mathematical results remain to be proved in the proof assistant Coq. The first one is Fermat's little theorem. Although there are proofs of this theorem in the contributions of Coq, they are not compatible with its standard library. The second one is the fact that if p is a prime number then the group $\mathbb{Z}_p^*$ is cyclic. Although those theorems are orthogonal to our work, it would be nice to have them machine checked, if only for the sake of completeness. For the moment, we have added them as axioms.

We neither compute exact nor asymptotic running time. This is orthogonal to the verification of game transformations. In the examples we dealt with, the algorithm A' we built from the attacker A at the end of Lemma 5.2, Theorem 5.3 and Theorem 5.5 are trivially PPT and thus valid attackers. However this is not checked by the current implementation.

7 Conclusions

We have extended our toolbox with number-theoretic capabilities. It is thus now possible to use this toolbox for machine-checking game-based proofs of arithmetic-based cryptographic primitives. We have shown usability of our implementation by applying it to the proof of unpredictability of the Blum-Blum-Shub generator and the proof of semantic security of the Goldwasser-Micali scheme. This is the first time that a proof of unpredictability is machine checked. Machine formalization has forced us to make clear all details in those proofs that are usually either left to the reader or roughly explained in English. In spite of this level of details, we claim that our proofs remain human readable and are mechanically human checkable without appealing too much to intuition.

Acknowledgements

We are grateful to Frédérique Oggier and Nicolas Perrin for fruitful discussions.

References

1. Affeldt, R., Tanaka, M., Marti, N.: Formal proof of provable security by game-playing in a proof assistant. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 151–168. Springer, Heidelberg (2007)
2. Backes, M., Berg, M., Unruh, D.: A formal language for cryptographic pseudocode. In: 4th Workshop on Formal and Computational Cryptography (FCC 2008) (2008)
3. Barthe, G., Grégoire, B., Janvier, R., Olmedo, F., Béguelin, S.Z.: Formal certification of code-based cryptographic proofs. In: 4th Workshop on Formal and Computational Cryptography (FCC 2008) (2008)
4. Bellare, M., Rogaway, P.: Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331 (2004)
5. Blanchet, B.: An efficient cryptographic protocol verifier based on Prolog rules. In: Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW-14), pp. 82–96. IEEE Computer Society, Los Alamitos (2001)
6. Blanchet, B., Pointcheval, D.: Automated security proofs with sequences of games. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 537–554. Springer, Heidelberg (2006)
7. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo random number generator. SIAM Journal on Computing 15(2), 364–383 (1986); an earlier version appeared in Proceedings of Crypto 1982
8. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences (JCSS) 28(2), 270–299 (1984); an earlier version appeared in proceedings of STOC 1982
9. Gonthier, G., Mahboubi, A., Rideau, L., Tassi, E., Théry, L.: A modular formalisation of finite group theory. In: Schneider, K., Brandt, J. (eds.) TPHOLs 2007. LNCS, vol. 4732, pp. 86–101. Springer, Heidelberg (2007)
10. Halevi, S.: A plausible approach to computer-aided cryptographic proofs. Cryptology ePrint Archive, Report 2005/181 (2005)
11. Knuth, D.E.: The Art of Computer Programming – Seminumerical Algorithms, vol. 2. Addison-Wesley, Reading (1969)
12. Lafourcade, P., Lakhnech, Y., Ene, C., Courant, J., Daubignard, M.: Towards automated proofs of asymmetric encryption schemes in the random oracle model. In: Proceedings of the 2008 ACM Conference on Computer and Communications Security, ACM, New York (2008) (to appear)
13. Nowak, D.: A framework for game-based security proofs. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 319–333. Springer, Heidelberg (2007); also available as Cryptology ePrint Archive, Report 2007/199
14. Ramsey, N., Pfeffer, A.: Stochastic lambda calculus and monads of probability distributions. In: Proceedings of the 29th ACM Symposium on the Principles of Programming Languages (POPL 2002), pp. 154–165. ACM, New York (2002)

15. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004)
16. Vazirani, U.V., Vazirani, V.V.: Efficient and secure pseudo-random number generation. In: Proceedings of the IEEE 25th Annual Symposium on Foundations of Computer Science (FOCS 1984), pp. 458–463. IEEE Computer Society, Los Alamitos (1984)
17. Yao, A.C.: Theory and applications of trapdoor functions. In: Proceedings of the IEEE 23rd Annual Symposium on Foundations of Computer Science (FOCS 1982), pp. 80–91. IEEE, Los Alamitos (1982)

A New Technique for Multidimensional Linear Cryptanalysis with Applications on Reduced Round Serpent

Joo Yeon Cho, Miia Hermelin, and Kaisa Nyberg

Helsinki University of Technology,
Department of Information and Computer Science,
P.O. Box 5400, FI-02015 TKK, Finland
{joo.cho,miia.hermelin,kaisa.nyberg}@tkk.fi

Abstract. In this paper, we present a new technique for Matsui's algorithm 2 using multidimensional linear approximation. We show that the data complexity of the attack can be reduced significantly by our method even when the linear hull effect is present. We apply our method to the key recovery attack on 5-round Serpent and demonstrate that our attack is superior to previous attacks. We present evidence that it is theoretically possible to reduce the data complexity of the linear attack against 10 round Serpent by factor of 2^{20} when multiple approximations are used.

Keywords: Block Ciphers, Linear Cryptanalysis, Serpent, Multidimensional Linear Approximation.

1 Introduction

Linear cryptanalysis is one of the most important methods of attack against block ciphers. Since Matsui introduced the linear cryptanalysis on DES in 1993, several attempts to generalize linear attack have been published. One approach is to use multiple linear approximations for the linear attack. In 1994, Kaliski and Robshaw [9] showed that the efficiency of the attack could be improved by using multiple linear approximation depending on the same key parity bit. In 2004, Biryukov, et al., [4] proposed a statistical framework for Matsui's algorithm 1 and 2 using multiple linear approximations and assuming similarly to [9] that the approximations are statistically independent. More rigorous statistical framework was proposed independently by Baignères, et al., in [2]. In 2008, Hermelin, et al., proposed a multidimensional statistical framework for Matsui's algorithm 1, for which the assumption on statistical independence is not needed [8].

In 2008, Collard, et al., [6] presented experimental results on the linear attack of Biryukov, et al., against reduced round Serpent. They showed that a linear attack on Serpent using Matsui's algorithm 1 could be improved significantly by exploiting multiple linear approximations, whereas a similar reduction of data complexity was not achieved using Matsui's algorithm 2. Authors claimed that

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 383–398, 2009.

this inconsistency was caused by the lack of good theoretical estimations of the correlations of the approximations due to the linear hull effect [10].

In this paper, we propose new techniques for Matsui's algorithm 2 using multiple linear approximations. In a similar way as in [8], we focus on the distribution of the multiple approximations rather than individual correlations. We present an efficient algorithm to apply the relative entropy between distributions for finding the right key in Matsui's algorithm 2. We also show that the maximum entropy of the distributions can be used to improve the efficiency of the key recovery attack when the distributions satisfy a certain general condition. We apply our techniques to reduced round Serpent and demonstrate that our method can reduce the data complexity of the attack significantly compared to the results of [6]. Hence, it seems to us that the linear hull effect is not the only reason to account for the experimental results of Matsui's algorithm 2 presented in [6].

This paper is organized as follows. In Section 2, the technical background of our attack method is presented. In Section 3, multiple linear approximations for reduced round Serpent are set up and the dependency of the theoretical advantage of the attack is illustrated for different cases according to the number of linearly independent linear approximations. In Section 4, previously proposed generalizations of linear attacks are described and the experimental results are shown. In Section 5, the new techniques are applied to reduced round Serpent and the experimental results are presented. Section 6 concludes this paper.

2 Technical Background

The first step in a traditional linear attack using Matsui's algorithm 2 is to find a linear approximation for the cipher that has the largest bias. Then, an attacker collects a large amount of plaintext-ciphertext pairs and counts the number of pairs that satisfy the linear approximation for each possible key values. The maximum bias over the counted samples indicates the right key value.

In a multidimensional linear attack, the attacker finds a class of linearly independent approximations whose biases are non-negligible. We call such linear independent approximations *base approximations*. If m linearly independent approximations are established, then additional $2^m - 1 - m$ approximations can be constructed as linear combinations of the m base approximations.

Provided that we have $2^m - 1$ approximations and their probabilities are $p_1, \ldots, p_{2^m-1}$, the *capacity* of the approximations, which is denoted by C, is defined as [4]

$$C = \sum_{i=1}^{2^m-1} (2p_i - 1)^2 = \sum_{i=1}^{2^m-1} c_i^2,$$

where $c_i = 2p_i - 1$ is called the *correlation* of the ith approximation.

In [2], a generalized statistical framework of the multidimensional linear attack was proposed. Let us consider a process that generates independent random variables $Z_{1,K}, Z_{2,K}, \ldots, Z_{2^m,K}$ depending on the key $K \in GF(2^l)$. Let K_0 denote the right key and $K_1, \ldots, K_{2^l-1}$ be the wrong key values. We assume that for $K = K_0$, all variables $Z_{i,K}$'s follow the distribution D_0, whereas for $K \neq K_0$, all $Z_{i,K}$'s follow the distribution D_1.

Suppose that we target to recover l-bit last round key. Once m base approximations have been established over all rounds of the cipher except for the last round, the linear attack using multiple approximations proceeds in four phases.

- **Counting Phase.** Collect the samples of the plaintext-ciphertext pairs on the targeted cipher and counts the number of samples which satisfy m-dimensional linear approximation.
- **Analysis Phase.** For each of the 2^l candidate keys, measure the distance of the empirical distribution from the theoretical distribution.
- **Sorting Phase.** Sort 2^l candidate keys according to their distances.
- **Searching Phase.** Exhaustively try all the candidate keys in the sorted order until the correct key is found.

In the analysis phase, the relative entropy between two distributions is measured as follows:

Definition 1. *The relative entropy or Kullback-Leibler distance between two distribution D_0 and D_1 is defined as*

$$D(D_0||D_1) = \sum_{z \in Z} Pr_{D_0}[z] \log \frac{Pr_{D_0}[z]}{Pr_{D_1}[z]}$$

with the assumptions that $p \log \frac{p}{0} = 0$ and $0 \log \frac{0}{p} = 0$.

Let $\Delta(D)$ denote the Squared Euclidean Imbalance [2] of the distribution D of a random variable taking values in the set $Z \subset GF(2^m)$. It is defined as

$$\Delta(D) = |Z| \sum_{z \in Z} (Pr_D[z] - \frac{1}{|Z|})^2.$$

Note that $C = \Delta(D)$ if D is the probability distribution of m base approximations as shown in [8].

Let N denote the number of samples and $\Phi(t)$ denote the cumulative normal distribution function that is defined as

$$\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{1}{2}u^2} du.$$

We apply the key ranking procedure, originally developed in [2] for the LLR-statistic, to the Kullback-Leibler distance, and assume that $D(D_0||D_1)|_{K=K_0} - D(D_0||D_1)|_{K \neq K_0}$ is approximately normally distributed with mean $\mu = N\Delta(D)$ and standard deviation $\sigma = \sqrt{2N\Delta(D)}$ [2]. Thus, the probability that a wrong key $K \neq K_0$ has a better rank than K_0 is approximately $\Phi(-\mu/\sigma) = \Phi(-\sqrt{N\Delta(D)/2})$ when the number of samples is large. Since the rank of K_0 is

$$1 + \sum_{K} 1_{D(D_0||D_1)|_{K=K_0} < D(D_0||D_1)|_{K \neq K_0}}$$

so the expected rank of K_0 is $1 + (2^l - 1)\Phi(-\sqrt{N\Delta(D)/2})$ [12,2].

In [11], Selçuk provided a statistical analysis of the success probability of linear cryptanalysis. If the correct value of the l-bit key is ranked at the r-th position out of 2^l possible candidates, the attack obtains an $(l - \log r)$-bit *advantage* over exhaustive search [11].[1] Therefore, the advantage a of the attack is expressed as

$$a = l - \log r = l - \log(1 + (2^l - 1)\Phi(-\sqrt{N\Delta(D)/2})) \qquad (1)$$

3 Multiple Linear Approximations of 4 Round Serpent

Suppose that we have m base approximations which are described as follows:

$$u_i \cdot P \oplus v_i \cdot C = \kappa_i \cdot K, \quad i = 1, \ldots, m$$

where u_i, v_i and κ_i stand for the input mask, output mask and the key mask, respectively. Also, P, C and K represent the plaintext, ciphertext and the key, respectively. The ”·” operation means a standard inner product. Given $\gamma = (\gamma_1, \ldots, \gamma_m)$ where $\gamma_i \in \{0, 1\}$ and $\gamma \neq (0, \ldots, 0)$, a combined approximation is constructed by

$$\bigoplus_{i=1}^{m} \gamma_i(u_i \cdot P \oplus v_i \cdot C) = \bigoplus_{i=1}^{m} \gamma_i(\kappa_i \cdot K).$$

Hence, we obtain $2^m - 1$ approximations in total.

We target to attack the 5-round Serpent using Matsui's algorithm 2. For this, we need to establish a chain of linear approximations over 4 rounds that has a significant bias. The best linear approximations for the 4-round Serpent were presented in [3] and [7]. Due to the structure of the round function of Serpent, one can obtain several linear approximations that hold with equal or slightly smaller bias based on the same round approxiamtions. The input and output masks on the base approximations used for our attack are listed in Table 3 in Appendix B. The linear approximations start from round 4 (using S-box 4) and end up in round 7 (using S-box 7). The output mask is chosen in such a way the number of active S-box in round 8 is minimal. Hence, the multiple approximations use only a single output mask and it is denoted as v_1 in Table 3.

Table 1 shows the correlations and the capacity of approximations for different numbers m of base approximations by which $2^m - 1$ approximations are obtained in total. Note that the base approximations are taken from the top of the list from Table 3 in order. Using Equation (1) and Table 1, we derive, for different values of m, the relation between the advantage of the attack and data complexity, which is illustrated in Figure 1.

So far, two types of linear attacks using multiple linear approximations have been investigated in the literature: linear attack using correlation (or type-I attack) and linear attack using distribution (or type-II attack). The attacks presented in [4] and [6] can be classified as type-I attack, whereas the multidimensional attack in [8] is a type-II attack. In the next section, we apply type-I attack to reduced round Serpent and show the experimental results.

[1] A slightly different measure of success was proposed for use in [4] where it was called as *gain*.

Table 1. The correlations and capacities according to 1, 4, 7, 10 and 12 base approximations

# base appr.		1	4	7	10	12
# combined appr.		0	11	120	1013	4083
correlation	2^{-13}	1	8	8	8	8
	2^{-14}	0	0	32	64	80
	2^{-15}	0	0	0	128	256
	2^{-16}	0	0	0	0	256
	0	0	7	87	823	3495
capacity		2^{-26}	2^{-23}	2^{-22}	2^{-21}	$2^{-20.42}$

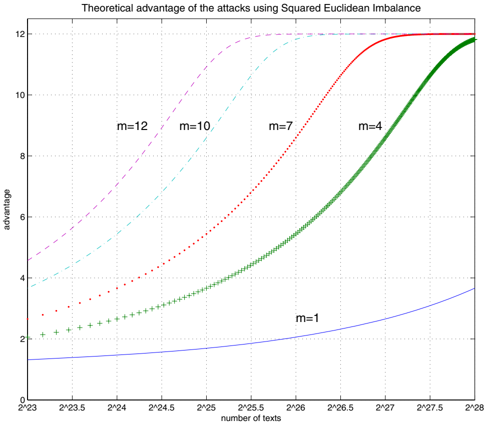


Fig. 1. Evaluation of the theoretical advantage of attacks using 1,4,7,10 and 12 base approximations

4 Linear Attacks Using Correlations of Multiple Approximations

Suppose that we have M linear approximations with correlations $c_1, \ldots, c_M$. The empirical correlations of M approximations by the key K are denoted by $\hat{c}_{1,K}, \ldots, \hat{c}_{M,K}$. Then, we consider the sum of the square of the correlations

$$||\hat{c}_K||^2 = \sum_{i=1}^{M} \hat{c}_{i,K}^2, \text{ where } K = 0, \ldots, 2^l - 1. \tag{2}$$

According to the wrong key hypothesis, it is assumed that $\hat{c}_{i,K \neq K_0}$ does not have any correlation (just like a random variable). Thus, the distance $||\hat{c}_K||^2$ by the correct key $K = K_0$ is expected to be significantly higher than the one induced by incorrectly guessed key $K \neq K_0$. Hence, the correct key can be recovered by taking K whose $||\hat{c}_K||^2$ is maximal.

In this method, it is not important whether the empirical correlations by the right key are matched to the theoretically calculated values or not. On the other hand, a method which Biryukov, et al., suggested in [4] is to extend Matsui's algorithm 1 for Matsui's algorithm 2 using multiple approximations. Hence, the accuracy of theoretically calculated correlations affects the performance of the attack.

Let us denote the parity key bits of the M approximations by $G = (g_1, \ldots, g_M)$, that is, $u_i \cdot P + v_i \cdot C = g_i$ where $1 \leq i \leq M$. For each value of a pair (K, G), a vector of theoretical correlations is constructed as follows:

$$c_{K,G} = (0, \ldots, 0, (-1)^{g_1} c_1, \ldots, (-1)^{g_M} c_M, 0, \ldots, 0),$$

where the location of the subvector $((-1)^{g_1} c_1, \ldots, (-1)^{g_M} c_M)$ depends on the value of K. Hence, the vector $c_{K,G}$ has $M \times 2^l$ entries and the number of possible pairs is $2^m \times 2^l$. Then, the distance between empirical correlation and theoretical correlation is measured using the following equation:

$$||\hat{c}_K - c_{K,G}||^2 = \sum_{j=1}^{M} (\hat{c}_{j,K} - (-1)^{g_j} c_j)^2 + \sum_{\kappa \neq K} \sum_{j=1}^{M} \hat{c}_{j,\kappa}^2. \tag{3}$$

The correct key is recovered by taking the key value whose $||\hat{c}_K - c_{K_i,G}||^2$ is minimal. If the linear hull effect [10] is not present, Equation (3) is slightly better than Equation (2) since two terms in Equation (3) are distinguishable for each value of K and G.

We applied two type-I attacks to the 5-round Serpent with various multiple linear approximations taken from Table 1. The experimental results are displayed in Figure 2. We can see in this figure that the advantage of the attack is far worse than the theoretical expectation shown in Figure 1. Furthermore, when more than 4 base approximations are used, the advantage of the attack becomes worse even though the capacity increases. This exemplifies that the data complexity required for the type-I attacks depends not only on the capacity but also on the distribution of approximations. The (exact) relation between the capacity, the number of approximations and data complexity required for the type-I attack remains an open problem.

5 Linear Attacks Using Distribution of Multiple Approximations

In this section, we propose new techniques on the linear attack using the distribution of multiple approximations. Our attack can be seen as an extension of the multidimensional linear attack [8] that was applied to Matsui's algorithm 1.

Suppose we have m base approximations and the boolean values of m approximations are $G = (g_1, \ldots, g_m)$. Using m base approximations, we build $2^m - 1$ approximations whose correlations are $c_1, \ldots, c_{2^m-1}$. Then, the theoretical prob-

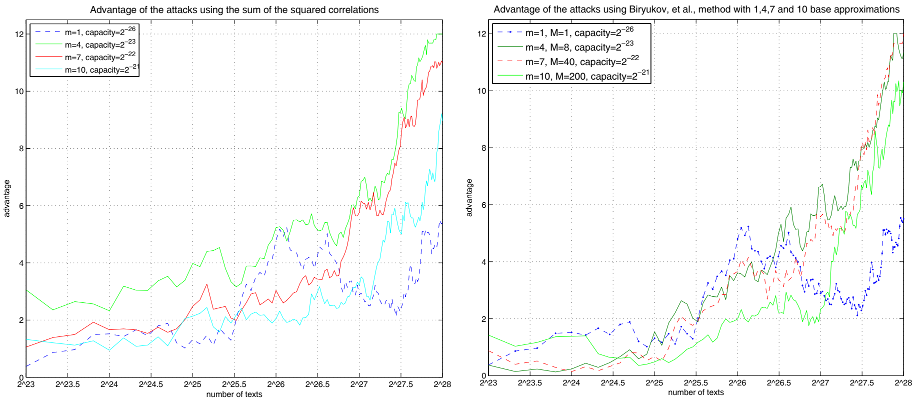


Fig. 2. Type-I linear attacks with 1, 4, 7 and 10 base approximations using Equation (2) (left) and (3) (right)

ability distribution of approximations is constructed in the following way [8]:

$$p_{i,G} = 2^{-m} + 2^{-m} \sum_{j=1}^{2^m-1} (-1)^{j \cdot i \oplus j \cdot G} c_j, \text{ where } i, G \in \{0,1\}^m. \tag{4}$$

Note that the size of theoretical distribution is $2^m \times 2^m$.

Let $P_G = (p_{0,G}, \ldots, p_{2^m-1,G})$ denote the theoretical distributions by the G. Then, it is clear from Equation (4) that the distribution G has the following property:

Property 1. A distribution $P_{G'}$ is a permutation of P_G for all $G' \neq G$. In particular, $p_{i,G} = p_{\bar{i},\bar{G}}$ where $\bar{X}$ is a bitwise negation of X.

Let us remind that only one output mask is used for the base approximations. This is a common situation for Matsui's algorithm 2 using multiple approximations for minimizing the active S-boxes. Since the output mask v_i for all base approximations is the same, only odd number of combinations of the base approximations have nonzero correlations among $2^m - 1$ possible approximations. Thus, Equation (4) is equivalently expressed as

$$p_{i,G} = 2^{-m} + 2^{-m} \sum_{j \in V_m} (-1)^{j \cdot i \oplus j \cdot G} c_j. \tag{5}$$

where $V_m = \{\nu | 0 < \nu < 2^m, \text{Hamming weight of } \nu \text{ is odd}\}$. From Equation (5), we can derive the following property:

Property 2. Since $\nu \cdot G \oplus \nu \cdot \bar{G} = 1$ for $\nu \in V_m$, we have

$$p_{i,G} = 2^{-m} + 2^{-m} \sum_{j \in V_m} (-1)^{j \cdot i \oplus j \cdot G} c_j = 2^{-m} + 2^{-m} \sum_{j \in V_m} (-1)^{j \cdot i \oplus j \cdot \bar{G} \oplus 1} c_j = 2^{-m+1} - p_{i,\bar{G}}.$$

By similar reason, we get $p_{\bar{i},G} = 2^{-m+1} - p_{i,G}$.

Since we target to recover l-bit of the last round key, we obtain 2^l empirical distributions for each of candidate key in the counting phase. Let $\hat{Q}_K = (\hat{q}_{0,K}, \ldots, \hat{q}_{2^m-1,K})$ denote the empirical distribution by the key K. It is known that a relative entropy between two distributions is measured optimally by Kullback-Leibler distance [2,8]. According to Definition 1, the Kullback-Leibler distance between the empirical distribution $\hat{Q}_K = (\hat{q}_{0,K}, \ldots, \hat{q}_{2^m-1,K})$ by K and the theoretical distributions $P_G = (p_{0,G}, \ldots, p_{2^m-1,G})$ by G is calculated as follows:

$$D(\hat{Q}_K||P_G) = \sum_{i=0}^{2^m-1} \hat{q}_{i,K} \log \frac{\hat{q}_{i,K}}{p_{i,G}}. \tag{6}$$

Once the empirical distribution for each candidate key is obtained, the analysis phase of our attack proceeds in two steps:

- **Step 1:** For each K, measure the distances $D(\hat{Q}_K||P_G)$ for all candidates of $G \in \{0,1\}^m$ and sort the candidates of G according to their distances.
- **Step 2:** For sorted values of G, measure $D(\hat{Q}_K||P_G)$ for all candidates of $K \in \{0,1\}^l$.

The step 1 applies Matsui's algorithm 1 to determine the right value of G, whereas in the step 2, Matsui's algorithm 2 is applied to recover the right value of K.

5.1 Using the Maximum Distance

In the original Matsui's algorithm 1, the correct parity key bit has the minimum Euclidean distance, whereas the maximum Euclidean distance indicates the opposite sign of the correct parity key. When multiple approximations are applied to Matsui's algorithm 1, it is natural to think that the correct values of multiple parity key bits hold the minimum squared Euclidean distance by Equation (2), whereas the opposite signed key parity bits have the maximum squared Euclidean distance. In this way, the maximum distance has the same amount of information as the minimum distance. However, it has been often ignored and not used for the linear attacks on the block ciphers.

When the distribution of the approximations is taken into account under the condition that all multiple approximations have the same output masks, a similar intuition can be applied. Due to Property 1, if $p_{i,G} = 2^{-m} + \epsilon_i$, then, $p_{i,\bar{G}} = 2^{-m} - \epsilon_i$. Hence, if the right value of G has the minimum value of $D(\hat{Q}_K||P_G)$, then, equivalently, the right value of $\bar{G}$ is expected to have the maximum value of $D(\hat{Q}_K||P_{\bar{G}})$. This intuition is proved in the following lemma:

Lemma 1. *Suppose that only a single output mask is used for m base approximations. Let G_{min} (resp. G_{max}) denote the G such that $D(\hat{Q}_K||P_G)$ is minimal (resp. maximal). If K is the correct key, then G_{min} and G_{max} are expected to have equivalent information and $G_{max} = \bar{G}_{min}$ where $\bar{X}$ is a bitwise negation of X.*

Proof. (*sketch*) For fixed G_0, we can write

$$D(\hat{Q}_K||P_{G_0}) - D(\hat{Q}_K||P_G) = \sum_{i=0}^{2^m-1} \hat{q}_{i,K} \log \frac{p_{i,G}}{p_{i,G_0}} = \sum_{i=0}^{2^m-1} \hat{q}_{i,K} \log \frac{p_{\bar{i},\bar{G}}}{p_{\bar{i},\bar{G}_0}}. \tag{7}$$

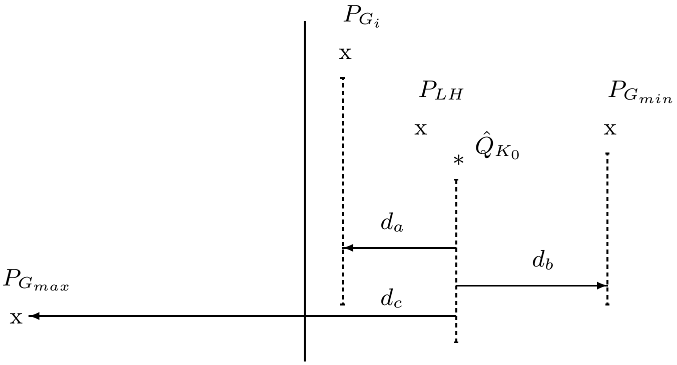


Fig. 3. An example of the usage of G_{max} when linear hull effect is present

It is expected that $\hat{Q}_K \approx P_G$ for some G if K is the right key. Thus, by Property 2, we can put $\hat{q}_{i,K} \approx 2^{-m+1} - \hat{q}_{\bar{i},K}$. Then, Equation (7) is approximated by

$$\sum_{i=0}^{2^m-1} (2^{-m+1} - \hat{q}_{\bar{i},K}) \log \frac{p_{\bar{i},\bar{G}}}{p_{\bar{i},\bar{G}_0}} = - \sum_{i=0}^{2^m-1} \hat{q}_{\bar{i},K} \log \frac{p_{\bar{i},\bar{G}}}{p_{\bar{i},\bar{G}_0}} = - \sum_{i=0}^{2^m-1} \hat{q}_{i,K} \log \frac{p_{i,\bar{G}}}{p_{i,\bar{G}_0}}$$
$$= D(\hat{Q}_K||P_{\bar{G}}) - D(\hat{Q}_K||P_{\bar{G}_0}).$$

since $\sum_{i=0}^{2^m-1} \log \frac{p_{\bar{i},\bar{G}}}{p_{\bar{i},\bar{G}_0}} = 0$ from Property 1. Hence, for any $G \neq G_0$, if $D(\hat{Q}_K||P_{G_0}) > D(\hat{Q}_K||P_G)$, then $D(\hat{Q}_K||P_{\bar{G}}) > D(\hat{Q}_K||P_{\bar{G}_0})$. □

However, our experiments showed that G_{max} was not always equal to $\bar{G}_{min}$. The reason is that, in practice, the theoretical distributions (which are constructed by the theoretical correlations) are not accurate due to the linear hull effect. In particular, our experiments show that the maximum distance is more reliable than the minimum distance.

Figure 3 provides an example of this situation. Let us assume that K_0 is the right key and P_{LH} is a true distribution. (LH denotes the linear hull.) Then, it is expected that an empirical distribution Q_{K_0} is close to P_{LH}. If a distribution $P_{G_{min}}$ is different from P_{LH}, there exist possibilities that $D(\hat{Q}_{K_0}||P_{G_{min}}) > D(\hat{Q}_{K_0}||P_{G_i})$ for some $G_i \neq G_{min}$. However, in the same situation, the relation $D(\hat{Q}_{K_0}||P_{G_{max}}) > D(\hat{Q}_{K_0}||P_{G_i})$ persists as illustrated in Figure 3. If the minimum distance is measured, G_i is (wrongly) guessed as a correct G since $d_b > d_a$. On the other hand, if the maximum distance is measured, G_{max} is guessed as a negation of correct G, since $d_c > d_a$. Our experimental results also show that a key recovery attack using G_{max} is superior to that using G_{min}. Hence, the right key is more reliably recovered by taking the key value from

$$\max_K \max_G D(\hat{Q}_K||P_G).$$

This observation is experimentally verified in Figure 4. More discussions on the experiments will be given in Subsection 5.4.

5.2 Summary of Our Method for Matsui's Algorithm 2

Given N plaintext-ciphertext pairs, our attack is described as follows.

- Initialize 2^l counters where l denotes the targeted key bits of the last round key.
- Compute the theoretical distribution of m approximations for each value of m parity bits and store them in a $2^m \times 2^m$-table.

- For each of the l-bit value of the last round key,
  - Decrypt the ciphertext partially using the guessed l-bit value of the last round key.
  - Compute the XOR of the input parity and output parity for each approximation.
  - Build an m-bit vector whose coordinates correspond the XORed parity bits of approximations.
  - Increment the counter indexed by both the vector and the l-bit guessed key.

- For each of the l-bit value of the last round key
  - For each of the m-bit value of the parity key, measure the Kullback-Leibler distance between the empirical distributions indexed by the l-bit value and the theoretical distribution indexed by the m-bit value.
  - Choose the maximum value of $D(\hat{Q}_K||P_G)$ for each K and store it as $D(\hat{Q}_K||P_{G_{max}})$.

- Sort all the candidate last round key using their values of $D(\hat{Q}_K||P_{G_{max}})$.
- Exhaustively try all keys from the sorted list of all candidate until the correct key is found.

5.3 Comparison of Time and Memory Complexity

Suppose that the number of base approximations for multidimensional linear attack is m and the targeted key size is l bits. For type-I attacks, we assume that M linear dependent approximations are used where m parity key bits are involved. Thus $m \leq M < 2^m$.

In the counting phase, for each key candidate and for each plaintext-ciphertext pair, type-I attacks need to update M counters by evaluating M approximations, while multidimensional attacks need to update one of 2^m counters by evaluating m base approximations. In the analysis phase, type-I attacks evaluate M correlations for each candidate of the last round key. In the multidimensional attacks, one distribution consisting of 2^m empirical frequencies is compared with 2^m different theoretical distributions by computing KL distances, where each KL distance has 2^m terms. The time complexity of multidimensional attack and type-I attacks using N plaintext-ciphertext pairs are summarized in Table 2.

Table 2. The time complexity of type-I attacks and multidimensional attack

	Squared Correlations Sum (Eq. (2))	Biryukov, et al., (Eq. (3))	Multidimensional
Counting phase	$N \cdot M \cdot 2^l$	$N \cdot M \cdot 2^l$	$N \cdot m \cdot 2^l$
Analysis phase	$M \cdot 2^l$	$M \cdot 2^{2l+m}$	2^{l+2m}
Recovered Key	l bits	$(l+m)$ bits	$(l+m)$ bits

For memory complexity, type-I attacks require 2^m storage for counters and multidimensional attack requires 2^{2m+1} storage for both the counters and the theoretical distribution.

Note that the multidimensional attack and the method of Biryukov, et al., can retrieve the information on the last round key K and the key parity G together. On the other hand, type-I attack using the sum of squares of correlations, see Equation (2), can recover the last round key K only.

5.4 Experimental Results

We applied our attack algorithm to the 5-round Serpent. We picked up 7, 10 and 12 base approximations from Table 1 and targeted to recovering of 12 bits of the last round key. The experimental results are displayed in Figure 4 and the results of type-I attacks are compared to them. In the experiments, the 128-bit secret keys and plaintexts were randomly generated and ciphertexts were collected by encrypting the plaintexts using 5-round Serpent.

Figure 4 shows that the advantage of the multidimensional linear attack using the maximum Kullback-Leibler Distance is significantly higher than for the other attacks. We also show that multidimensional attack using the minimum Kullback-Leibler Distance is worse than the other attacks. This suggests that the linear attack using the minimum distance may be more vulnerable to the linear hull effect. Finally, we note that our experimental results are still worse than the theoretical curves in Figure 1 that were drawn by Equation (1). Further research is required for the statistical modeling of multidimensional linear approximation and to find the optimal multidimensional extension of Matsui's algorithm 2 and to accurately predict its performance.

5.5 Extension for Further Rounds of Serpent

Our attack can be further applied for a larger number of round of Serpent since we can obtain multiple approximations simply by applying various input masks in the first round. For instance, the linear attacks on 10-round Serpent in [3] use a 9-round linear approximation with probability of $\frac{1}{2}(1-2^{-57})$. Thus, the capacity of the best single approximation is $2^{-2\times 57} = 2^{-114}$. On the other hand, we can construct multiple linear approximations from the same linear trail of 9-round Serpent. The first round of the linear trail includes 10 active S-boxes and each S-box has 10 non-negligible approximations (2 for 2^{-1} and 8 for 2^{-2} correlations for each S-box) for a fixed output. Thus, we can construct in total $10^{10} \approx 2^{33}$ approximations that have non-negligible correlations. The best correlation of the

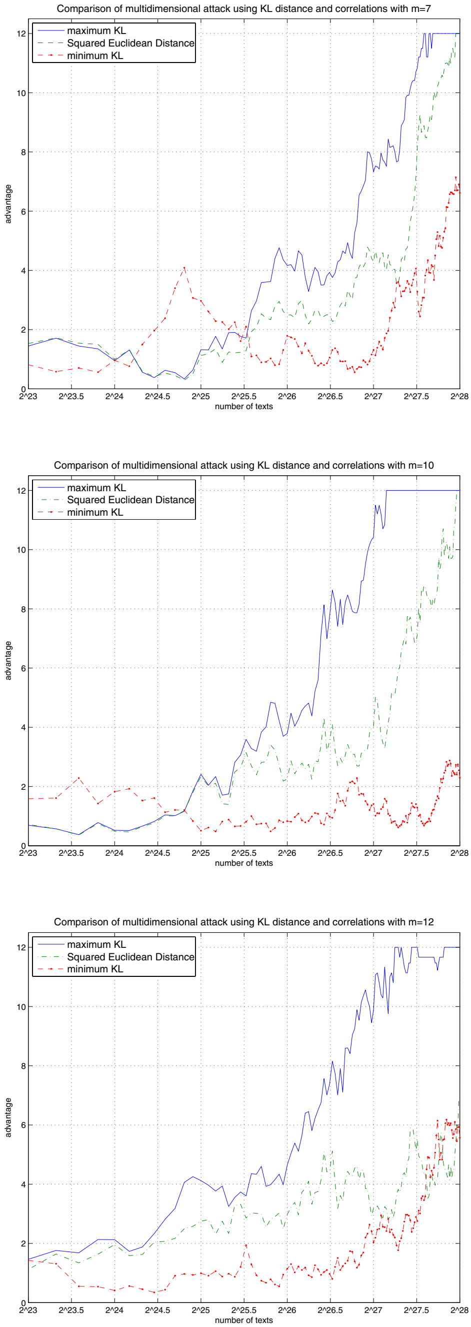


Fig. 4. Comparison of multidimensional attacks and other attacks with various base approximations

first round approximation is 2^{-10} and the number of approximations with the best correlations is 2^{10}. In the same way, the second best correlation of the first round approximation is 2^{-11} and 10×2^{12} approximations hold such correlation, and so on. Hence, the capacity of 2^{33} approximations can be computed as

$$C = 2^{10}\binom{10}{0}2^{-57\times 2} + 2^{12}\binom{10}{1}2^{-58\times 2} + \cdots + 2^{30}\binom{10}{10}2^{-67\times 2} = 2^{-94}. \quad (8)$$

Therefore, the data complexity of linear attack on 10 round Serpent can be reduced theoretically by a factor of 2^{20} at the cost of increased time complexity.

In [5], Collard et al. also presented the multiple linear attacks against 10-round Serpent. According to [5], the best attack on 10-round Serpent needs 2^{99} known plaintexts with 2^{99} time complexity and 2^{55} memory for recovering 44 bits of the last round key. This attack uses $M = 2^{11}$ linear approximations and each approximation has the equal bias of 2^{-55}. Hence, the capacity is $(2 \cdot 2^{-55})^2 \cdot 2^{11} = 2^{-97}$.

On the other hand, the multidimensional linear attack method allows us to use all the linear approximation involved in 9-round linear trails within the span of the base approximations. Since the number of active S-boxes of the first round is 11 and each S-box has 10 linear approximations, the number of possible approximations is actually 10^{11}. Hence, the capacity is computed as $2^{11}\binom{11}{0}2^{-54\times 2}+\cdots+2^{33}\binom{11}{11}2^{-65\times 2} = 2^{-86}$. Therefore, it is theoretically possible to reduce the data complexity of the attack further by a factor of 2^{11}. Instead, the time complexity increases by around $2^{l+2m} = 2^{132}$ with the memory complexity of around $2^{2m+1} = 2^{89}$.

6 Conclusion

In this paper, we proposed a new technique for the multidimensional linear attacks. We showed that the multidimensional linear attack could be very powerful with Matsui's algorithm 2 when multiple linear approximations are available in the block ciphers. The improvements we achieved using the new techniques stem from two reasons. Firstly, we take the distribution of the approximations in a multidimensional way and we measure the distances between two distributions using Kullback-Leibler distance instead of the sum of the squared correlations. Secondly, by taking the maximal value of the distances, our method eliminated errors in the situation where correlations of individual linear approximations could not be calculated accurately due to the linear hull effect. However, it is an open problem whether our heuristic technique is optimal and what is its expected performance.

References

1. Anderson, R., Biham, E., Knudsen, L.: Serpent: A proposal for the advanced encryption standard. In: First Advanced Encryption Standard (AES) conference (1998)
2. Baignères, T., Junod, P., Vaudenay, S.: How Far Can We Go Beyond Linear Cryptanalysis? In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 432–450. Springer, Heidelberg (2004)

3. Biham, E., Dunkelman, O., Keller, N.: Linear cryptanalysis of reduced round Serpent. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 219–238. 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. Collard, B., Standaert, F., Quisquater, J.: Improved and multiple linear cryptanalysis of reduced round Serpent. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 47–61. Springer, Heidelberg (2008)
6. Collard, B., Standaert, F., Quisquater, J.: Experiments on the multiple linear cryptanalysis of reduced round serpent. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 382–397. Springer, Heidelberg (2008)
7. Collard, B., Standaert, F., Quisquater, J. (Accessed on 31.07.2008), `http://www.dice.ucl.ac.be/fstandae/PUBLIS/50b.zip`
8. Hermelin, M., Cho, J., Nyberg, K.: Multidimensional linear cryptanalysis of reduced round Serpent. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 203–215. Springer, Heidelberg (2008)
9. Kaliski, B., Robshaw, M.: Linear cryptanalysis using multiple approximations. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 26–39. Springer, Heidelberg (1994)
10. Nyberg, K.: Linear approximation of block ciphers. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
11. Seluk, A.: On probability of success in linear and differential cryptanalysis. Journal of Cryptology 21(1), 131–147 (2008)
12. Vaudenay, S.: An experiment on DES statistical cryptanalysis. In: CCS 1996: Proceedings of the 3rd ACM conference on Computer and communications security, pp. 139–147. ACM, New York (1996)

A Brief Description of Serpent Algorithm

We use the notation of [1]. Each intermediate value of round i is denoted by $\hat{B}_i$ (a 128-bit value). Each $\hat{B}_i$ is treated as four 32-bit words X_0, X_1, X_2, X_3 where bit j of X_i is bit $4 * i + j$ of the $\hat{B}_i$. Serpent has a set of eight 4-bit to 4-bit S Boxes $S_0, \ldots, S_7$ and a 128-bit to 128-bit linear transformation LT. Each round function R_i uses a single S-box 32 times in parallel.

Serpent ciphering algorithm is formally described as follows.

$$\begin{aligned} \hat{B}_0 &= P \\ \hat{B_{i+1}} &= R_i(\hat{B}_i) \\ C &= B_{32} \end{aligned}$$

where

$$\begin{aligned} R_i(X) &= LT(\hat{S}_i(X \oplus \hat{K}_i)), \quad i = 0, \ldots, 30 \\ R_i(X) &= \hat{S}_i(X \oplus \hat{K}_i) \oplus \hat{K}_{32}, \quad i = 31 \end{aligned}$$

The linear transformation LT is described as follows.

$$X_0, X_1, X_2, X_3 = S_i(B_i \oplus K_i)$$

$$\begin{aligned}
X_0 &= X_0 \lll 12\\
X_2 &= X_2 \lll 3\\
X_1 &= X_1 \oplus X_0 \oplus X_2\\
X_3 &= X_3 \oplus X_2 \oplus (X_0 \lll 3)\\
X_1 &= X_1 \lll 1\\
X_3 &= X_3 \lll 7\\
X_0 &= X_0 \oplus X_1 \oplus X_3\\
X_2 &= X_2 \oplus X_3 \oplus (X_1 \lll 7)\\
X_0 &= X_0 \lll 5\\
X_2 &= X_2 \lll 22\\
B_{i+1} &= X_0, X_1, X_2, X_3
\end{aligned}$$

The detailed description of Serpent can be found in [1].

B Linearly Independent Approximations on 4 Round Serpent

In our experiments, we used 12 base approximations from the linear trail of 4 round Serpent. The linear approximations start from round 4 (using S-box 4) and end up in round 7 (using S-box 7). Table 3 shows the input and output masks of the base approximations that are expressed as

$$u_i \cdot P \oplus v_i \cdot C = \kappa_i \cdot K, \quad i = 1, \ldots, m$$

where the u_i and v_i denote the input and out masks, respectively. Hence, u_i is an input mask of round 4 and v_i is an output mask of round 7. We omit the key mask κ_i since the exact knowledge of κ_i is not required for our attack.

Table 3. Input and output masks for the multidimensional linear attack using Matsui's algorithm 2

type	index	mask = (MSB, ..., LSB)
input mask	u_1	(0x70000000, 0x00000000, 0x00000000, 0x07000900)
	u_2	(0x70000000, 0x00000000, 0x00000000, 0x07000B00)
	u_3	(0x70000000, 0x00000000, 0x00000000, 0x0B000900)
	u_4	(0xB0000000, 0x00000000, 0x00000000, 0x07000900)
	u_5	(0x70000000, 0x00000000, 0x00000000, 0x07000500)
	u_6	(0x70000000, 0x00000000, 0x00000000, 0x07000600)
	u_7	(0x70000000, 0x00000000, 0x00000000, 0x07000C00)
	u_8	(0x70000000, 0x00000000, 0x00000000, 0x01000900)
	u_9	(0x70000000, 0x00000000, 0x00000000, 0x0A000900)
	u_{10}	(0xB0000000, 0x00000000, 0x00000000, 0x03000B00)
	u_{11}	(0x10000000, 0x00000000, 0x00000000, 0x07000900)
	u_{12}	(0x40000000, 0x00000000, 0x00000000, 0x0B000B00)
output mask	v_1	(0x00001000, 0x01000000, 0x00000000, 0x00000000)

The notation of masks are following [3]. For instance, in the input mask

$$u_1 = (0x70000000, 0x00000000, 0x00000000, 0x07000900)$$

the first 4 bits (which is '7') is an input of the leftmost S-Box of the first round. Hence, there are three active S-boxes in the first round. In the same way, there are two active S-boxes in the second last round by the output mask v_1.

Almost Fully Optimized Infinite Classes of Boolean Functions Resistant to (Fast) Algebraic Cryptanalysis

Enes Pasalic

IMFM Ljubljana & University of Primorska, Koper
Slovenia
enespasalic@yahoo.se

Abstract. In this paper the possibilities of an iterative concatenation method towards construction of Boolean functions resistant to algebraic cryptanalysis are investigated. The notion of $\mathcal{AAR}$ (Algebraic Attack Resistant) function is introduced as a unified measure of protection against classical algebraic attacks as well as fast algebraic attacks. Then, it is shown that functions that posses the highest resistance to fast algebraic attacks are necessarily of maximum $\mathcal{AI}$ (Algebraic Immunity), the notion introduced in [20] defined as a minimum degree of functions that annihilate either f or $1+f$. More precisely, if for any non-annihilating function g of degree e an optimum degree relation $e+d \geq n$ is satisfied in the product $fg=h$ (denoting $deg(h)=d$), then the function f in n variables must have maximum $\mathcal{AI}$, i.e. for nonzero function g the relation $fg=0$ or $(1+f)g=0$ implies $deg(g) \geq \frac{n}{2}$. The presented theoretical framework allows us to iteratively construct functions with maximum $\mathcal{AI}$ satisfying $e+d \geq n-1$, thus almost optimized resistance to fast algebraic cryptanalysis. This infinite class for the first time, apart from almost optimal resistance to algebraic cryptanalysis, in addition generates the functions that possess high nonlinearity (superior to previous constructions) and maximum algebraic degree, thus unifying most of the relevant cryptographic criteria.

Keywords: Fast Algebraic attacks, Algebraic Immunity, Annihilators, Algebraic Attack Resistant, High Degree Product, Stream ciphers, Boolean function.

1 Introduction

Algebraic cryptanalysis has received a lot of attention recently. The technique has proved efficient in cryptanalysis of certain LFSR-based stream ciphers such as LILI-128 proposed in [25] and Toyocrypt[1], both successfully cryptanalyzed in [12]. Apart from its application to LFSR-based stream ciphers algebraic cryptanalysis is also used as an efficient representation method for certain block cipher algorithms such as encryption standards DES [11] and AES [13,22].

[1] Submission to the Japanese government Cryptrec call for cryptographic primitives.

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 399–414, 2009.

The design of LFSR-based stream ciphers traditionally resides on the use highly nonlinear Boolean functions as filtering functions; the two major representatives being nonlinear filter generators and nonlinear combiners [21]. For instance, in the case of nonlinear filter generators n stages of a single Linear Feedback Shift Registers (LFSRs) (whose initial state consists of the secret key) are filtered by a nonlinear Boolean function $f : GF(2)^n \rightarrow GF(2)$ to provide the keystream sequence.

Apart from already established cryptographic criteria such as nonlinearity, algebraic degree, and resiliency, it turned out that the Boolean function must also have a certain order of algebraic immunity. This is due to recently introduced algebraic attacks based on the low degree annihilation of Boolean functions [8,12]. These attacks reflect the property of certain cipher schemes for which the selection of function f of high algebraic degree that follows early ideas of Shannon's concept of confusion [24], and linear complexity attacks [21], is not a sufficient criterion any longer. Due to algebraic attacks, instead of setting up a system of equations of degree determined by the degree of function f, the attacker can consider a lower degree system if there either exists a low degree function g (called annihilator) such that $fg = 0$ or alternatively $(1 + f)g = 0$ [12,20]. The minimum degree of nonzero annihilators g of either f or $1 + f$ is called *algebraic immunity* ($\mathcal{AI}$). Algebraic attacks currently present one of the most efficient cryptanalytic tool in stream cipher cryptanalysis; the applications include many prominent algorithms such as Bluetooth encryption algorithm E_0 analyzed in e.g. [10].

A few construction methods that generate functions reaching the upper bound on algebraic immunity $\lceil \frac{n}{2} \rceil$ (maximum $\mathcal{AI}$) functions) has recently been proposed [6,14,4,15,19]. However, all the known methods do not succeed in optimization of other cryptographic criteria at the same time. Furthermore, though a high order of $\mathcal{AI}$ implies resistance to classical algebraic attacks this property is only necessary but not sufficient criterion. The emergence of fast algebraic cryptanalysis [1,9] still successfully invalidates any design for which there exists low degree function g such that $fg = h$ is of relatively low degree as well. For instance, fast algebraic attack was successfully applied to eSTREAM [16] proposal Sfinks [3], though the cipher was designed to withstand classical algebraic attacks. Denoting by e and d the degree of g and h respectively, the resistance to fast algebraic cryptanalysis is optimized if $e + d \geq n$ for any non-annihilating g and $e \in [1, \lceil \frac{n}{2} \rceil - 1]$.

This work is mainly motivated by the fact that at the time being all the construction methods fail to provide functions unifying all the important cryptographic criteria. This is especially true when the fast algebraic attacks are taken into account. In this direction we first derive some theoretical results that relate the notions of algebraic immunity and resistance to fast algebraic attacks. A unified measure against both fast and classical cryptanalysis is introduced, the notion that we name $\mathcal{AAR}$ (algebraic attack resistance). The notion of $\mathcal{AAR}$ includes the maximum $\mathcal{AI}$ property per definition, but it is shown that another related concept *high degree product* actually includes the maximum $\mathcal{AI}$

property. This framework is then used in deriving the set of sufficient conditions for a certain recursive, concatenation-based construction to generate $\mathcal{AAR}$ functions. These conditions being extremely hard to satisfy, the optimum requirement $e+d \geq n$ is slightly relaxed ($e+d \geq n-1$ is used instead) which then enables an iterative method for constructing functions satisfying all the relevant cryptographic criteria in the design of nonlinear filter generators.

The rest of the paper is organized as follows. Basic definitions and notations are introduced in Section 2. Section 3 gives a thorough treatment regarding the algebraic properties of the iterative construction technique based on the concatenation of four functions. Furthermore, the new notion of $\mathcal{AAR}$ functions is introduced, and the relationship between the optimal resistance to fast and to classical algebraic attacks is deduced. These results are then utilized in Section 4 for proposing an iterative method for generation of suboptimized $\mathcal{AAR}$ functions, with overall good cryptographic properties. The cryptographic properties are discussed in details, both from the security and implementation point of view. Some concluding remarks are given in Section 5.

2 Preliminaries

We denote the Galois field of order 2^n by $\mathbb{F}_{2^n}$ and the corresponding vector space by $\mathbb{F}_2^n$. A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is usually represented via so called *algebraic normal form* (ANF),

$$f(x_1, \ldots, x_n) = \sum_{u \in \mathbb{F}_2^n} \lambda_u \left(\prod_{i=1}^{n} x_i^{u_i} \right) \;, \quad \lambda_u \in \mathbb{F}_2 \;, u = (u_1, \ldots, u_n). \tag{1}$$

For the rest of this paper, if otherwise not stated, x will denote a vector containing n input binary variables, that is $x = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$. Then the *algebraic degree* of f, denoted by $deg(f)$ or sometimes simply d, is the maximal value of the Hamming weight of u such that $\lambda_u \neq 0$. The set of all Boolean functions in n variables is denoted by $\mathcal{B}_n$, and functions of degree at most one are called *affine* functions, whose associated set is denoted $\mathcal{A}_n$. The *nonlinearity* of an n-variable function f is defined as

$$\mathcal{N}_f = min_{g \in \mathcal{A}_n}(d_H(f, g)), \tag{2}$$

where d_H is the Hamming distance between two binary vectors, that is the number of positions where f and g differ.

The support set of function $f \in \mathcal{B}_n$, denoted by $supp(f)$, is the set of input values where f has a nonzero evaluation, that is,

$$supp(f) = \{x \in \mathbb{F}_2^n \mid f(x) = 1\}.$$

A function f is said to be balanced if it outputs equal number of zeros and ones, that is

$$\#\{x \in \mathbb{F}_2^n : f(x) = 1\} = \#\{x \in \mathbb{F}_2^n : f(x) = 0\}.$$

3 Theoretical Framework towards Resistance to Algebraic Attacks

A construction method based on the concatenation of functions from smaller variable space has been frequently used as an efficient tool in the design of cryptographically strong Boolean functions. Nevertheless, the known methods have failed so far in providing good functions resistant to both fast and classical algebraic cryptanalysis. These classes of functions are also attractive in terms of an efficient hardware implementation. In this section we study the algebraic properties of an iterative concatenation method involving four subfunctions. In addition, a general relation that interlinks the optimum resistance to fast and classical algebraic cryptanalysis is derived.

3.1 Some Properties of Functions with Maximum $\mathcal{AI}$

The purpose of this section is to identify some basic conditions that any function with maximum $\mathcal{AI}$ must satisfy with respect to its subfunctions. For the rest of this manuscript we focus on the representation of $f \in \mathcal{B}_{n+2}$ as a concatenation of four functions, that is, $f = f_1||f_2||f_3||f_4 \in \mathcal{B}_{n+2}$, where each $f_i \in \mathcal{B}_n$ has maximum $\mathcal{AI}$. Using the shortened notation (f_i denoting $f_i(x)$), the ANF of function f is given by:

$$f = x_{n+1}x_{n+2}(f_1 + f_2 + f_3 + f_4) + x_{n+1}(f_1 + f_2) + x_{n+2}(f_1 + f_3) + f_1. \quad (3)$$

A similar expression is then valid for any annihilator g of f,

$$g = x_{n+1}x_{n+2}(g_1 + g_2 + g_3 + g_4) + x_{n+1}(g_1 + g_2) + x_{n+2}(g_1 + g_3) + g_1, \quad (4)$$

where g_i is arbitrary annihilator of f_i (including the trivial annihilation $g_i = 0$). Let g_i denote any minimum degree nonzero annihilator of $f_i \in \mathcal{B}_n$. If $deg(g_i) = d$ then we also use,

$$g_i(x) = T_d(g_i(x)) + T_{d-1}(g_i(x)) + \cdots + T_0(g_i(x)),$$

where each $T_r(g_i)$, for $0 \leq r \leq d$, contains only degree r monomial terms. Then in connection to the representation of annihilator g of f given in (4), the following simple properties are deduced. [2]

Lemma 1. *Let $f = f_1||f_2||f_3||f_4$, where $f_i \in B_n$ are functions with maximum $\mathcal{AI}$. Then any nonzero annihilator g of f represented as in (4) satisfies the following :*
(i) *If any $g_i = 0$ then $deg(g) \geq \lceil \frac{n}{2} \rceil + 1$.*

(ii) *If any g_i is such that $deg(g_i) > \lceil \frac{n}{2} \rceil$ then $deg(g) \geq \lceil \frac{n}{2} \rceil + 1$.*

[2] This result was independently derived in [2, Ch. 4]. The author of this article made this result available on Cryptology eprint archive in 2005, but the paper was soon withdrawn due to one erroneous result.

(iii) *If there exists g such that $deg(g) < \lceil \frac{n}{2} \rceil + 1$ then $deg(g_i) = \lceil \frac{n}{2} \rceil$ for all $i \in [1, 4]$ and furthermore,*

$$T_d(g_1) = T_d(g_2) = T_d(g_3) = T_d(g_4); \qquad T_{d-1}(\sum_{i=1}^{4} g_i) = 0. \tag{5}$$

Proof. (i) W.l.o.g. assume $g_1 = 0$, then $x_{n+1}(g_1 + g_2)$ is of degree at least $\lceil \frac{n}{2} \rceil + 1$ unless $g_2 = 0$. But $g_1 = g_2 = 0$ implies that $g_3 = 0$ due to the term $x_{n+2}(g_1 + g_3)$, as otherwise $deg(g) \geq \lceil \frac{n}{2} \rceil + 1$.
(ii) The similar idea is used here. Taking any g_i so that $deg(g_i) > \lceil \frac{n}{2} \rceil$, implies that $deg(g) \geq \lceil \frac{n}{2} \rceil + 1$.
(iii) Assuming $deg(g) < \lceil \frac{n}{2} \rceil + 1$ gives that,

$$T_d(\sum_{i=1}^{4} g_i) = 0 \quad T_{d-1}(\sum_{i=1}^{4} g_i) = 0 \quad T_d(g_1 + g_2) = 0 \quad T_d(g_1 + g_3) = 0,$$

and the result easily follows. □

The result of Lemma 1, in particular item (iii), is a useful tool for establishing the algebraic properties of given function. Showing that subfunctions $f_1, \ldots, f_4$ of maximum $\mathcal{AI}$ are chosen so that item (iii) above cannot be satisfied for neither f nor $1 + f$ is equivalent to proving $f = f_1||f_2||f_3||f_4$ has a maximum $\mathcal{AI}$. It has been used in [2], where the iterative method of designing the maximum $\mathcal{AI}$ functions was proposed. The construction requires three suitable input functions $f_1^0, f_2^0, f_3^0 \in \mathbb{F}_2^{n_0}$ to iteratively generate maximum $\mathcal{AI}$ functions,

$$\begin{aligned} f_1^i &= f_1^{i-1}||f_2^{i-1}||1 + f_3^{i-1}||f_1^{i-1} \\ f_2^i &= f_2^{i-1}||1 + f_3^{i-1}||f_1^{i-1}||f_2^{i-1} \quad i \geq 1 \\ f_3^i &= 1 + f_3^{i-1}||f_1^{i-1}||f_2^{i-1}||f_3^{i-1}. \end{aligned} \tag{6}$$

This method generates the maximum $\mathcal{AI}$ functions with relatively good nonlinearity, at least the nonlinearity value is superior compared to the constructions in [7,14,6,4,15,19]. A set of initial functions satisfying the conditions for the above recursion to generate maximum $\mathcal{AI}$ is e.g. $f_1^0 = x_1x_2 + x_3, f_2^0 = x_2x_3 + x_4, f_3^0 = x_2x_4 + x_1$ [2]. The main problem with this construction is its unknown susceptibility to fast algebraic attacks.

3.2 Functions Resistant to (Fast) Algebraic Attacks

We have already mentioned that functions optimizing the algebraic immunity does not protect from fast algebraic attacks in case there exists a degree e function g such that $fg = h$ is of degree d, and $e + d < n$. The efficiency of the fast algebraic attack depends on both parameters and finding a tuple (e, d) so that $e + d$ is substantially smaller than n will result in an overall lower attack complexity. A more elaborate description of how fast algebraic attacks work can be found in e.g. [1,9] but for convenience the main steps of algorithm and corresponding complexities are summarized here.

1. *Search for relations.* Finding the low-valued (e, d) equation[s] for f of type $fg = h$ (sometimes also denoted $zX^e + X^d$ [1]). The complexity is roughly $\binom{n}{d}$ and is negligible in comparison to other steps.
2. *Pre-computation step.* For a given LFSR of length L and known characteristic polynomial, a universal binary string α of length $D = \sum_{i=0}^{d} \binom{L}{i}$ can be computed in $D \log^2 D$ operations [9,1,18].
3. *Substitution step.* The original degree d equations are rewritten via substitution process to yield degree e equations. This step takes about $2ED \log^2 D$ operations [18], where $E = \sum_{i=0}^{e} \binom{L}{i}$.
4. *Solving step.* The degree e system of equations is solved by linearization; this requires E^ω operations, where ω is the complexity of solving linear system (usually $\omega = 3$ as a conservative estimate).

Assuming the existence of small e, the dominating step in terms of complexity is the substitution step. Therefore the fast algebraic attacks imply reduced computational complexity (compared to classical algebraic attacks) whenever there exists (e, d) tuple(s) such that $2ED \log^2 D < D_{\mathcal{AI}}^{\omega}$, where $D_{\mathcal{AI}} = \sum_{i=0}^{\mathcal{AI}(f)} \binom{L}{i}$.

In [9], it was proved that there always exists a tuple (e, d) (denoting the degree of functions g and h respectively) if e, d satisfy the bound $e + d \geq n$. But the statement that there are integers e, d meeting this bound, that is $e + d = n$ was recently disproved [17]. In this paper both the necessary and sufficient conditions (not only sufficient as in [9] on the existence of such functions meeting the bound $e + d = n$ was derived. Furthermore, a few examples of so-called degenerated cases when $e + d > n$ are also provided in [17].

A straightforward relationship between the existence of (e, d)-relations and the degree of function f can be deduced [9,2].

Theorem 1. *[2, Ch.3][9] If f is of degree k then f satisfies a $(k, k+i)$-relation for any $i < k$.*

Proof. For any functions f and g of degree k respectively i, $deg(fg) \leq k + i$. □

For a properly chosen algebraic immunity (to resist classical algebraic attacks), ensuring that (e, d) satisfy $e + d \geq n$ for any choice of e, d will imply protection against fast algebraic attacks partially due to the following result:

Lemma 2. *[2, Ch.3] For any functions $f, g \in \mathcal{B}_n$ such that $g \neq 0$ is not an annihilator of f we have $deg(fg) = d \geq \mathcal{AI}(f)$.*

The sufficiency of the condition $e + d \geq n$ comes from the complexity estimate of the substitution step above.

Proposition 1. *The complexity of the substitution step in the fast algebraic attack is approximately the same (up to a logarithmic constant) for any choice of e, d satisfying $e + d = n$, $e \in [1, \mathcal{AI}(f) - 1]$, $d \in [\mathcal{AI}(f), n - 1]$.*

Proof. For a given state size S, the complexity of substitution step is $2ED \log^2 D$, where $D = \sum_{i=0}^{d} \binom{S}{i}$, and $E = \sum_{i=0}^{e} \binom{L}{i}$. Neglecting the logarithmic term, and approximating $\sum_{i=0}^{u} \binom{S}{i} \approx S^u$ (for $u \ll S$) we have,

$$2ED \log^2 D \approx 2ED = 2S^e S^d = 2S^{e+d} = 2S^n.$$

□

Table 1. Complexity of the substitution step for various (e, d); $L = 160$ and $n = 16$

(e, d)	(1,15)	(2,14)	(3,13)	(4,12)	(5,11)	(6,10)	(7,9)
$2ED \log^2 D$	2^{88}	2^{91}	2^{93}	2^{95}	2^{96}	$2^{96.7}$	2^{97}

Note that in case n is odd, for a function f of maximum $\mathcal{AI}$ there will always exist a tuple $(e, d) = (\lceil \frac{n}{2} \rceil - 1, \lceil \frac{n}{2} \rceil)$ and therefore the upper bound on the security for an LFSR based stream cipher application is estimated through the complexity of fast algebraic attacks with above (e, d). The goal is to ensure that (e, d) satisfies $e + d \geq n$ for any $1 \leq e \leq \mathcal{AI}(f) - 1$, and $d \geq \mathcal{AI}(f)$.

The constant behavior of $2ED \log^2 D$ is illustrated in the following practical context of usage. Assume that a nonlinear filtering generator uses an LFSR of length 160 and a filtering Boolean function $f : \mathbb{F}_2^{16} \rightarrow \mathbb{F}_2$. In case the generator is designed for 80 bits security and $e + d \geq 16$ then the complexity of the substitution step is given in Table 1. Thus, it seems to be well-motivated to introduce a new quantity that would measure the resistance of function to both algebraic and fast algebraic attacks.

Definition 1. *Let f be a Boolean function on $\mathbb{F}_2^n$, with n of arbitrary parity. Then the function f is called algebraic attack resistant ($\mathcal{AAR}$) if f has a maximum $\mathcal{AI}$, that is $\mathcal{AI}(f) = \lceil \frac{n}{2} \rceil$, and furthermore for any non-annihilating function g of degree e, $1 \leq e \leq \lceil \frac{n}{2} \rceil - 1$, we necessarily have that $deg(fg) = d$ satisfies $e + d \geq n$. The latter property is referred to as $\mathcal{HDP}$ (High Degree Product) of order n.*

The property of $\mathcal{HDP}$ is irrelevant to the complement operation.

Proposition 2. *If function $f \in \mathcal{B}_n$ satisfies the $\mathcal{HDP}$ property of order n so does the function $1 + f$.*

Proof. By assumption for any non-annihilating function g of degree e and $h = fg$ of degree d, we have $e + d \geq n$. Then for any $deg(g) = e$ function g,

$$(1 + f)g = fg + g = h + g,$$

and consequently $deg((1 + f)g = deg(g + h)$. Since $deg(h) \geq n - e$, then for any $e \in [1, \lceil \frac{n}{2} \rceil - 1]$ we have $n - e > e$ and therefore $deg(g + h) = deg(h)$. □

The $\mathcal{AAR}$ property appears to be somewhat related to the concept of algebraic immunity through the result given by Lemma 2. Indeed, there is an explicit relationship connecting the notions of $\mathcal{AI}$ and $\mathcal{HDP}$ (the upper bound on the $e + d$). It turns out that the functions satisfying the $\mathcal{HDP}$ property of order n automatically achieve the maximum $\mathcal{AI}$.

Theorem 2. *Let $f \in \mathcal{B}_n$. Assume that $fg = h$ satisfies $e + d \geq n$ for any choice of non-annihilating function g of degree e, and h of degree d, for $e \in [1, \mathcal{AI}(f)]$, and $d \in [\mathcal{AI}(f), n - 1]$. Then f has maximum $\mathcal{AI}$.*

Proof. On contrary assume that $\mathcal{AI}(f) < \lceil \frac{n}{2} \rceil$, i.e. f has not maximum $\mathcal{AI}(f)$. When n is odd $\mathcal{AI}(f) = \frac{n+1}{2}$ if and only if $deg(An(f)) = \frac{n+1}{2}$, that is $deg(An(f)) = deg(An(1+f)) = \frac{n+1}{2}$ [5] (for even n the relationship between $An(f)$ and $An(1+f)$ is an open problem). Let $\tilde{g} \in An(1+f)$ such that $deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$. Then,

$$(1+f)\tilde{g} = 0 \Longrightarrow f\tilde{g} = \tilde{g}.$$

Since $deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$ we have actually found (e,d) not satisfying $e + d \geq n$ (as $e = d = deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$), contradicting the assumption on f.

For n even we consider two cases. If $\tilde{g} \in An(1+f)$ such that $deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$ then the proof is exactly the same as above. For $\tilde{g} \in An(f)$ such that $deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$, by Proposition 2 function $(1+f)$ satisfy the $\mathcal{HDP}$ property. That is, for any g, h of degree e and d respectively, $e + d \geq n$ in the product $(1+f)g = h$. Then considering the product $(1+f)\tilde{g} = \tilde{g}$, as $f\tilde{g} = 0$, would contradict the $\mathcal{HDP}$ property if $deg(\tilde{g}) < \lceil \frac{n}{2} \rceil$. □

This result states that $\mathcal{HDP}^{(n)} \Rightarrow \mathcal{AI}$, therefore the criteria for optimum $\mathcal{AI}$ and resistance to fast algebraic analysis is unified through the $\mathcal{HDP}$ property, that is $\mathcal{AAR} \Leftrightarrow \mathcal{HDP}^{(n)}$. Still, a simple attempt to generate $\mathcal{AAR}$ function in $n+1$ variables by relating the $\mathcal{AAR}$ subfunctions in n variables will fail as demonstrated by the example below.

Example 1. *Let $f \in \mathcal{B}_n$ be an $\mathcal{AAR}$ function, for n odd. Since f is an $\mathcal{AAR}$ function then it is easy to show that $f' = f||1+f$ has maximum $\mathcal{AI}$. Now for any non-annihilating function g of fixed degree e we have to prove that $d \geq n+1-e$, where $deg(f'g) = d$. Note that $e \in [1, \frac{n+1}{2} - 1]$, as trivially there is a tuple $(e,d) = (\frac{n+1}{2}, \frac{n+1}{2})$, and we are interested in cases $e < d$, with $d \geq \mathcal{AI}(f') = \frac{n+1}{2}$. Any function $g \in \mathcal{B}_{n+1}$ can be written as,*

$$g(x, x_{n+1}) = x_{n+1}(g_1(x) + g_2(x)) + g_1(x), \quad g_1, g_2 \in \mathcal{B}_n.$$

Then, $f'g = x_{n+1}[g_2 + f(g_1 + g_2)] + fg_1$ and taking $g_1 = g_2$ gives $f'g = [x_{n+1} + f]g_1$ which only satisfies the relation $e+d \geq n$ but not $e+d \geq n+1$. Hence the function $f' = f||1+f$ is of maximum $\mathcal{AI}$ but not necessarily an $\mathcal{AAR}$ function.

4 An Iterative Design of Almost Optimal $\mathcal{AAR}$ Functions

In general, when f and g are represented as a concatenation of four functions (cf. equations (3) and (4)), the product $fg \in \mathcal{B}_{n+2}$ can be after some simplification written as,

$$x_{n+2}x_{n+1}\big[g_4 \sum_{j=1}^{4} f_j + (f_1 + f_2 + f_3) \sum_{j=1}^{4} g_j + (f_1 + f_2)(g_1 + g_3) + (f_1 + f_3)$$
$$(g_1 + g_2)\big] + x_{n+1}(f_1 g_1 + f_2 g_2) + x_{n+2}(f_1 g_1 + f_3 g_3) + f_1 g_1 = fg \qquad (7)$$

The only way to analyze the behavior of the above product is to put certain restrictions on the form of the subfunctions f_j. In order to simplify the above expression we select three distinct function on $\mathbb{F}_2^n$, denoting them f_1, f_2, f_4 and introduce the dependency on f_3, that is $f_3 = 1 + f_1$. Then, the derived class of functions on $\mathbb{F}_2^{n+2}$ is closely related to the construction given by (6).

Theorem 3. *Let $f = f_1||f_2||f_3||f_4$ be a function on $\mathbb{F}_2^{n+2}$, n even, whose subfunctions $f_i \in B_n$ satisfy the following:*

1. *$f_1, \ldots, f_4$ are $\mathcal{AAR}$ functions with $f_3 = 1 + f_1$*
2. *For any function $g = g_1||g_2||g_3||g_4$ of degree e, the functions f_2, f_4 satisfy $deg(g_3 + f_2g_2 + f_4g_4) \geq n - e$, where not both functions g_2, g_4 are zero and $e \in [1, \lceil \frac{n}{2} \rceil]$.*

Then, $f \in B_{n+2}$ is an $\mathcal{AAR}$ function.

Proof. To prove the $\mathcal{AAR}$ property, by Theorem 2 we only need to show that f satisfies degree relation $e + d \geq n + 2$.

Due to the $\mathcal{AAR}$ assumption on f_i, we have $deg(f_ig_j) \geq n - e$ for any degree e function g_j, $e \in [1, \lceil \frac{n}{2} \rceil]$. Using the relation $f_3 = 1 + f_1$ the product fg in (7) may be written as,

$$\begin{aligned} fg = x_{n+2}x_{n+1}[g_3 + f_4g_4 + f_1(g_1 + g_3) + f_2g_2] + x_{n+1}[f_1g_1 + f_2g_2] \\ +x_{n+2}[g_3 + f_1(g_1 + g_3)] + f_1g_1. \end{aligned}$$

We want to show that any nonzero choice of function g of fixed degree e, $e \in [1, \lceil \frac{n}{2} \rceil]$, implies that $deg(fg) = d \geq n + 2 - e$. Recall that,

$$g = x_{n+1}x_{n+2}(g_1 + g_2 + g_3 + g_4) + x_{n+1}(g_1 + g_2) + x_{n+2}(g_1 + g_3) + g_1.$$

implying $deg(g_i) \leq e$. Consider the coefficient $g_3 + f_1(g_1 + g_3)$ of x_{n+2} in the product fg. Obviously, we must have $deg(g_1 + g_3) \leq e - 1$ as otherwise the degree of g is greater than e, due to the term $x_{n+2}(g_1 + g_3)$. The $\mathcal{AAR}$ condition on f_1 implies that $deg(f_1g_a) \geq n - e$ for any nonzero degree e function g_a. By Lemma 2, $n - e \geq \lceil \frac{n}{2} \rceil$.

We now show that $deg(f_1(g_1 + g_3)) > deg(g_3)$ for any choice of g_1, g_3 such that $g_1 + g_3 \neq 0$. The condition $deg(g_1 + g_3) \leq e - 1$ implies $deg(f_1(g_1 + g_3)) \geq n - (e - 1) = n + 1 - e$, and therefore $deg(fg) \geq n - e + 2$. Since f_1 is an $\mathcal{AAR}$ function $n - e \geq \lceil \frac{n}{2} \rceil$. Then $n + 1 - e > \lceil \frac{n}{2} \rceil$ and consequently $deg(f_1(g_1 + g_3)) > deg(g_3)$, as $\deg(g_3) \leq \lceil \frac{n}{2} \rceil$. Hence, the degree of $g_3 + f_1(g_1 + g_3)$ is governed by $f_1(g_1 + g_3)$, and because $deg(f_1(g_1 + g_3)) \geq n + 1 - e$ the function fg is an $\mathcal{AAR}$ function unless $g_1 = g_3$.

The subcase $g_1 = g_3 = 0$ results in an $\mathcal{AAR}$ function f due to the following. The term $x_{n+1}(g_1 + g_2)$ in function g implies that $deg(g_2) \leq e - 1$ assuming $g_1 = 0$. Consequently $deg(f_2g_2) \geq n + 1 - e$ and fg is of degree $\geq n + 2 - e$ due to $x_{n+1}[f_1g_1 + f_2g_2]$. Thus, $g_1 = g_3 = 0$ would imply $g_2 = 0$, implying restriction $deg(g_4) = e - 2$ (because g is of degree e), so that $deg(fg) \geq n + 4 - e$ due to the term $x_{n+2}x_{n+1}[g_3 + f_4g_4 + f_1(g_1 + g_3) + f_2g_2]$.

Hence if f is not an $\mathcal{AAR}$ function we must necessarily have $g_1 = g_3 \neq 0$, and we get somewhat simplified expressions for g and fg,

$$g = x_{n+1}x_{n+2}(g_2 + g_4) + x_{n+1}(g_1 + g_2) + g_1,$$

$$fg = x_{n+2}x_{n+1}[g_3 + f_4g_4 + f_2g_2] + x_{n+1}[f_1g_1 + f_2g_2] + x_{n+2}g_3 + f_1g_1. \quad (8)$$

By assumption $deg(g_3 + f_2g_2 + f_4g_4) \geq n - e$, implying $fg \geq n + 2 - e$. □

Remark 1. *The second condition in Theorem 3 may be slightly relaxed by requiring that $deg(f_2g_2 + f_4g_4) \geq n - e$. In this case the above result holds for any $e \in [1, \lceil \frac{n}{2} \rceil - 1]$ except for the case $e = \lceil \frac{n}{2} \rceil$, as there may exist $g_1, \ldots, g_4$, $deg(g_i) = \lceil \frac{n}{2} \rceil$ such that $deg(g_3 + f_2g_2 + f_4g_4) < n - e = \lceil \frac{n}{2} \rceil$. This would imply the possibility of finding tuple $(e, d) = (\lceil \frac{n}{2} \rceil, \lceil \frac{n}{2} \rceil + 1)$, thus violating $e + d \geq n + 2$.*

4.1 Iterative Construction of Maximum $\mathcal{AI}$ Functions with $e + d \geq n - 1$

To use the result of Theorem 3 recursively the conditions on initial functions turn out to be extremely hard to satisfy. Note first, that a similar set of constraints is obtained after the replacement $f_1 \leftarrow f_2$, $f_2 \leftarrow 1 + f_2$, $f_3 \leftarrow f_1$, $f_4 \leftarrow f_3$, thus referring to the function f_2^i in (6). The product $f_2^i g$ can then be written as,

$$\begin{aligned} f_2^i g = {} & x_{n+2}x_{n+1}[g_2 + f_1g_3 + f_3g_4 + f_2(g_1 + g_2)] + x_{n+1}[f_2(g_1 + g_2) + g_2] \\ & + x_{n+2}[f_2g_1 + f_1g_3] + f_2g_1. \end{aligned}$$

Similarly to the proof of Theorem 3, we get that the condition $g_1 = g_2 \neq 0$ from the term $x_{n+1}[f_2(g_1 + g_2) + g_2]$ then implies that $deg(g_2 + f_1g_3 + f_3g_4) \geq n - e$ for any choice of g_3 and g_4. Thus, a very similar condition as in Theorem 3 applies here, only different subfunctions being involved.

The main question now is what kind of conditions the set of initial functions must satisfy so that the $\mathcal{AAR}$ property is preserved in the recursion given by (6). In other words, assuming that the functions $f_1^{i-1}, f_2^{i-1}, f_3^{i-1}$ satisfy particular set of conditions we would like to show that the $\mathcal{AAR}$ property holds also for f_1^i, f_2^i, f_3^i. One can show that these conditions for f_1^i, f_2^i, f_3^i become rather complicated involving the all three subfunctions and multiplicand g as well, yielding the degree constraint of the form,

$$deg[\sum_{j=1}^{4} a_j g_j^{i-1} + f_1^{i-1}(\sum_{j=1}^{4} b_j g_j^{i-1}) + f_2^{i-1}(\sum_{j=1}^{4} c_j g_j^{i-1}) + f_3^{i-1}(\sum_{j=1}^{4} d_j g_j)] \geq n - e, \quad (9)$$

for some binary coefficients $a_i, \ldots, d_i$.

However, the main obstacle in satisfying such initial conditions turns out to be the term $\sum_{j=1}^{4} a_j g_j^{i-1}$. As already indicated in Remark 1 the conditions are "slightly" relaxed if we allow a small deviation from optimality. That is, allowing the initial functions to satisfy $e + d \geq n - 1$ (instead of $e + d \geq n$) in the product $fg = h$, we may much easier find suitable initial functions to be used in a recursive manner. The condition that $e + d \geq n - 1$ implies that the degree of the expression $g_i + f_j g_k$ for functions $g_i, f_j, g_k \in \mathcal{B}_n$ is always dominated by $f_j g_k$. This is because we now only consider $e \in [1, \lceil \frac{n}{2} \rceil - 1]$ and regardless the parity of n we have,

$$deg(f_j g_k) \geq \mathcal{AI}(f_j) = \lceil \frac{n}{2} \rceil > e.$$

We notice that allowing the suboptimized case of degree relation $e+d \geq n-1$, Theorem 2 is not applicable any longer and we are now forced to induce the optimality of $\mathcal{AI}$ through the construction. A class of function achieving maximum $\mathcal{AI}$ and satisfying the relation $e+d \geq n-1$ is then called *suboptimized* $\mathcal{AAR}$ class. Therefore, we utilize the design ideas given in (6) but in a different manner. We slightly refine the construction to enable the usage of nonbalanced functions as initial functions too, though generating balanced functions in subsequent iterations. This modification will have a great impact on the cryptographic properties of the functions. In the first place the nonlinearity is improved, the functions are of maximum algebraic degree, optimized $\mathcal{AI}$ and they satisfy the $\mathcal{HDP}$ property of order $n - 1$.

The most suitable configuration of subfunctions that allows the use of nonbalanced initial functions seems to be the following one,

$$\begin{aligned} f_1^i &= f_1^{i-1} || f_2^{i-1} || 1 + f_1^{i-1} || f_3^{i-1} \\ f_2^i &= f_2^{i-1} || 1 + f_3^{i-1} || f_1^{i-1} || 1 + f_2^{i-1} \\ f_3^i &= 1 + f_3^{i-1} || f_1^{i-1} || f_2^{i-1} || f_3^{i-1}. \end{aligned} \tag{10}$$

One may readily check that selecting $f_i^0 \in \mathcal{B}_n$ such that their Hamming weight equal to $wt(f_1^0) = wt(f_3^0) = 2^{n-1} - c$, and $wt(f_2^0) = 2^{n-1} + c$ will result in balanced functions f_j^i. It remains to show that a suitable selection of the input functions will initiate the recursion so that any f_j^i, $i \geq 0$ and $j = 1, 2, 3$, is a maximum $\mathcal{AI}$ satisfying $e + d \geq n - 1$.

Theorem 4. *Let $f_1^0, f_2^0, f_3^0 \in \mathcal{B}_n$ be maximum $\mathcal{AI}$ functions satisfying the set of conditions given in Lemma 1 (iii) with respect to the configuration in (10). In addition, let for any $g = g_1^0 || g_2^0 || g_3^0 || g_4^0 \in \mathcal{B}_{n+2}$ of degree $e \in [1, \lceil \frac{n}{2} \rceil - 1]$ the following is satisfied,*

$$deg[f_1^0(\sum_{j=1}^{4} b_j g_j^0) + f_2^0(\sum_{j=1}^{4} c_j g_j^0) + f_3^0(\sum_{j=1}^{4} d_j g_j^0)] \geq n - e - 1; \quad b_j, c_j, d_j \in \mathbb{F}_2. \tag{11}$$

Then the function $f_j^i \in \mathcal{B}_{n+2i}$, $i \geq 0$ and $j = 1, 2, 3$, defined by (10), are maximum $\mathcal{AI}$ functions with almost optimized $\mathcal{HDP}$, that is satisfying $e + d \geq n + 2i - 1$ for $e \in [1, \lceil \frac{n}{2} \rceil + i - 1]$.

Proof. The fact that f_j^i have maximum $\mathcal{AI}$ follows from hypothesis. The result concerning the $e + d$ relation is proved by induction. The case $i = 0$ follows directly from the assumption. Suppose the conditions are satisfied for all $k < i$.

We show that the conditions hold for $k+1$ as well. By assumption, the functions $f_1^k, f_2^k, f_3^k \in \mathcal{B}_{n+2k}$ are such that,

$$deg[f_1^{k-1}(\sum_{j=1}^{4} b_j g_j^{k-1}) + f_2^{k-1}(\sum_{j=1}^{4} c_j g_j^{k-1}) + f_3^{k-1}(\sum_{j=1}^{4} d_j g_j^{k-1})] \geq n+2k-e-1, \tag{12}$$

where $f_j^{k-1}, g_j^{k-1} \in \mathcal{B}_{n+2k-2}$. W.l.o.g. we consider the function $f_1^{k+1} = f_1^k||f_2^k||1 + f_1^k||f_3^k$. Then for a degree e function $g^{k+1} \in \mathcal{B}_{n+2k+2}$ we need to show that, $deg(f_1^{k+1}g^{k+1}) \geq n+2k-e+1$ for any $e \in [1, \lceil \frac{n}{2} \rceil + k - 1]$. Let us focus on the highest degree term in the product $f_1^{k+1}g^{k+1}$ that is,

$$x_{n+2k+1}x_{n+2k+2}[g_3^k + f_4^k g_4^k + f_1^k(g_1^k + g_3^k) + f_2^k g_2^k],$$

where we represent $g^{k+1} = g_1^k||g_2^k||g_3^k||g_4^k$. Now since $deg(g_3^k) \leq e < n+2k-e-1$ for any $e \in [1, \lceil \frac{n}{2} \rceil + k - 1]$, the degree of the terms in the brackets above is dominated by the sum $f_4^k g_4^k + f_1^k(g_1^k + g_3^k) + f_2^k g_2^k$. This sum when written in terms of the subfunctions f_j^{k-1} and g_j^{k-1} gives the condition (12) which is satisfied by induction hypothesis. Thus,

$$deg\{x_{n+2k+1}x_{n+2k+2}[g_3^k + f_4^k g_4^k + f_1^k(g_1^k + g_3^k) + f_2^k g_2^k]\} \geq n+2k-e+1,$$

which proves the statement. □

4.2 Cryptographic Properties of the Construction

Through computer simulations (very non-exhaustive) we have found many sets of initial functions on $\mathbb{F}_2^4$ satisfying the conditions of Theorem 4. The set of initial functions given below gives the best nonlinearity so far.[3]

$$\begin{aligned} f_1 &= x_1 + x_1x_2 + x_3x_4 + x_1x_2x_3 + x_1x_2x_3x_4, \\ f_2 &= x_2 + x_4 + x_1x_2 + x_2x_4 + x_3x_4 + x_1x_2x_3 + x_1x_3x_4 + x_2x_3x_4 + x_1x_2x_3x_4, \\ f_3 &= x_2 + x_3 + x_1x_2 + x_2x_3 + x_3x_4 + x_1x_2x_3 + x_1x_2x_3x_4. \end{aligned}$$

Algebraic degree of f_j^i: The algebraic degree of any f^i is given by the relation $\deg(f^i) = d_0 + 2i$, where d_0 is the degree of initial functions. The only assumption made is that the degree d_0 terms in f_i^0 do not cancel each other in the sum $f_1^0 + f_2^0 + f_3^0$. This is easily verified by considering the ANF of f_1^1 (using $f_3 = 1 + f_1$) for instance, and the degree of f_1 is dominated by $x_{n+1}x_{n+2}(f_2^0 + f_3^0)$ Also, the highest degree terms cannot be canceled out in further iterations which justifies the above statement.

Resistance to probabilistic algebraic attacks: Probabilistic algebraic attacks, formally introduced as scenarios S4 and S6 in [12], are based on a low degree

[3] We have only performed a local search by selecting the three random functions and then manually modifying few bits in the truth tables of functions. An exhaustive search would imply checking around 2^{45} different choices for f_1^0, f_2^0, f_3^0.

Table 2. Comparison of relevant cryptographic criteria

function	degree	nonlinearity	$\mathcal{AI}$	(e,d)
$\phi_8/f^8/f_*^8$	5/6/**7**	88/104/**104**	4 all	?/(2,4)/$e+d \geq 7$
$\phi_{10}/f^{10}/f_*^{10}$	8/8/**9**	372/452/**452**	5 all	?/(3,5)/$e+d \geq 9$
$\phi_{12}/f^{12}/f_*^{12}$	?/10/**11**	?(low)/1884/**1890**	6 all	?/(1,9)/$e+d \geq 11$
$\phi_{14}/f^{14}/f_*^{14}$	?/12/**13**	?(low)/7696/**7780**	7 all	?/(2,10)/$e+d \geq 13$
$\phi_{16}/f^{16}/f_*^{16}$	?/14/**15**	?(low)/31296/**31766**	8 all	?/(1,13)/$e+d \geq 15$

approximation of state equations (or filtering function), so that relatively simple equations that are true with probability close to 1 (preferably) are derived. This approach was successfully applied in cryptanalysis of Toyocrypt [8] due to a serious design flaw of Toyocrypt.[4] A low degree approximation to a filtering function constructed using the iterative method described above seems to be rather unrealistic. This is due to the fact that each iteration step essentially combine/add the monomials of two different functions, the sum being multiplied by new variables, cf. equation (3). Thus, it can be easily verified that the algebraic normal form of the resulting function will contain many high order terms, and therefore approximating these relation would require guessing many variables which in turn would reduce the probability that these relations hold.

Nonlinearity: A rather loose lower bound on nonlinearity was derived in [2], the minimum value is estimated as

$$\mathcal{N}_f \geq n^{(f_1,1+f_1)} \cdot N_{f_1} + n^{(f_2,1+f_2)} \cdot N_{f_2} + n^{(f_3,1+f_3)} \cdot N_{f_3},$$

where $n^{(f_i,1+f_i)}$ denotes the number of times the tuple $(f_i, 1+f_i)$ appears in the overall concatenation. A comparison in terms of relevant cryptographic criteria to the iterative construction methods in [7] and in [2] is given in Table 2. Here the functions ϕ_n and f^n are optimized $\mathcal{AI}$ functions obtained through the methods in [7] and [2] respectively, and f_*^n and corresponding bold face entries denotes our design. Obviously, both f^n and f_*^n are favorable to ϕ_n in terms of the nonlinearity, and degree. Nevertheless, our class is superior to f^n, providing functions *satisfying the $\mathcal{HDP}$ property of order $n-1$ thus providing a better resistance to fast algebraic attacks, having better nonlinearity and optimized algebraic degree.* The nonlinearity values related to our construction are obtained by running a computer program, whereas the algebraic properties (also confirmed by computer simulations) follows from Theorem 4 and above discussion on the algebraic degree.

Remark 2. *The results given above only consider the construction for even n. Though the same technique is applicable when n is odd, good initial functions seem to be harder to find then. The function space $\mathcal{B}_3$ is quite insufficient in this context, whereas selecting $f_j^0 \in \mathcal{B}_5$ gives on the other hand far too many*

[4] Notice that an application of classical algebraic attack on Toyocrypt yields even lower time complexity compared to probabilistic algebraic attacks.

possibilities. A suitable set, that generates highly nonlinear functions, is therefore to be found through sophisticated computer program.

Implementation: In the view of a new version of algebraic attacks introduced in [23], whose running time is significantly lower than for the fast/classical algebraic attacks, an efficient implementation of filtering function seems to be of great importance. Since the running time of this attack is approximately $D = \sum_{i=0}^{\deg(f)} \binom{L}{i}$ (L being as before the length of LFSR), the functions of more than 30 variables are required to guard against different modes of algebraic analysis. Then the implementation issue actually contradicts the fundamental ideas behind the design of nonlinear filters, as these are designed for restricted hardware environments. This implies that the filtering function must have a sufficient algebraic structure for the ease of implementation, especially if the input space of the function is as large as some 30 variables.

To compute the functions in the i-th iteration step of our construction given by (10) (this is of course true for the original construction of [2]) the concatenation of 2^{2^i} initial functions $F = \{f_1^0, f_2^0, f_3^0, 1 + f_1^0, 1 + f_2^0, 1 + f_3^0\}$ is needed. Given the input value $x = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+2i})$ it can be shown that a simple loop of i iterations is required to compute the output value. For instance, evaluating the function $f_1^2 \in \mathcal{B}_8$ for a given input $x = (x_1, \ldots, x_8) \in \mathbb{F}_2^8$ is done by computing the integer value of $(x_5, \ldots, x_8)$ and then evaluating the function enumerated by this number on the input $(x_1, \ldots, x_4)$ in the concatenated sequence,

$$f_1^2 = f_1^0||f_2^0||\bar{f}_1^0||f_3^0||f_2^0||\bar{f}_3^0||f_1^0||\bar{f}_2^0||\bar{f}_1{}^0||\bar{f}_2^0||f_1^0||\bar{f}_3^0||\bar{f}_3^0||f_1^0||f_2^0||f_3^0.$$

4.3 Finding Good Initial Functions

Computer simulations indicate that different choices of input functions result in classes of functions with significantly different nonlinearity values, while in most of the cases the degree and algebraic resistance remain invariant. For instance, complementing 2 bits in f_3^0 given above would yield the nonlinearity values 1864/7748/31500 compared to the above sequence 1890/7780/31766. The nonlinearities given here for even n are obtained by manually flipping a few bits in the truth tables of input functions, and no exhaustive search has been performed. The search for optimum set of initial functions (including larger input spaces) is in progress.

5 Conclusion

This paper proposes an iterative construction method for designing almost fully optimized Boolean functions satisfying most of the cryptographic criteria. The construction is very efficient from the implementation point of view making it attractive even when the input space exceeds some 30 variables. It remains to find optimized set of initial functions (especially for odd n) and possibly another construction configurations (to modify (10)) to further increase the nonlinearity, thus making functions (ciphers) more resistant to fast correlation and distinguishing attacks.

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. Braeken, A.: Cryptographic properties of Boolean functions and S-boxes. Ph. D. thesis, Katholieke Universiteit Leuven, Belgium (2006)
3. Braeken, A., Lano, J., Mentens, N., Preneel, B., Verbauwhede, I.: Sfinks specification and source code. ECRYPT Stream Cipher Project page (2005), `http://www.ecrypt.eu.org/stream/sfinks.html`
4. 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)
5. Canteaut, A.: Invited talk: Open problems related to algebraic attacks stream ciphers. In: Ytrehus, Ø. (ed.) WCC 2005. LNCS, vol. 3969, pp. 120–134. Springer, Heidelberg (2006)
6. Carlet, C.: Improving the algebraic immunity of resilient and nonlinear functions and constructing bent functions. Cryptology ePrint Archive, Report 2004/276 (2004), `http://eprint.iacr.org/`
7. Carlet, C., Dalai, K.D., Gupta, C.K., Maitra, S.: Algebraic immunity for cryptographically significant Boolean functions: Analysis and construction. IEEE Trans. on Inform. Theory IT-52(7), 3105–3121 (2006)
8. Courtois, N.T.: Higher order correlation attacks,XL algorithm and cryptanalysis of toyocrypt. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 182–199. Springer, Heidelberg (2003)
9. Courtois, N.T.: 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. Courtois, N.: Algebraic attacks on combiner with memory and several outputs. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 3–20. Springer, Heidelberg (2005)
11. Courtois, N.T., Bard, G.V.: Algebraic cryptanalysis of the data encryption standard. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 152–169. Springer, Heidelberg (2007)
12. 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)
13. Courtois, N.T., 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)
14. Dalai, D.K., Gupta, K.C., Maitra, S.: Cryptographically significant boolean functions: Construction and analysis in terms of algebraic immunity. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 98–111. Springer, Heidelberg (2005)
15. Dalai, D.K., Maitra, S., Sarkar, S.: Basic theory in construction of Boolean functions with maximum annihilator immunity. Designs, Codes, and Cryptography 40(1), 41–58 (2006)

16. ECRYPT. Call for stream cipher primitives, http://www.ecrypt.eu.org/stream/
17. Gong, G.: Sequences, DFT and resistance against fast algebraic attacks. In: Golomb, S.W., Parker, M.G., Pott, A., Winterhof, A. (eds.) SETA 2008. LNCS, vol. 5203. Springer, Heidelberg (2008)
18. Hawkes, P., Rose, G.G.: Rewriting variables: The complexity of fast algebraic attacks on stream ciphers. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 390–406. Springer, Heidelberg (2004)
19. Li, N., Qi, W.-F.: Construction and analysis of boolean functions of $2t+1$ variables with maximum algebraic immunity. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 84–98. Springer, Heidelberg (2006)
20. 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)
21. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
22. Murphy, S., Robshaw, M.J.B.: Essential algebraic structure within the AES. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 1–16. Springer, Heidelberg (2002)
23. Ronjom, S., Helleseth, T.: A new attack on the filter generator. IEEE Trans. on Inform. Theory IT 53(5), 1752–1758 (2007)
24. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423 (Part I), 623–656 (Part II) (1948)
25. Simpson, L.R., Dawson, E., Golić, J.D., Millan, W.L.: LILI keystream generator. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 248–261. Springer, Heidelberg (2001)

Higher Order Differential Attacks on Reduced-Round MISTY1

Yukiyasu Tsunoo[1], Teruo Saito[2], Maki Shigeri[2], and Takeshi Kawabata[2]

[1] NEC Corporation
1753, Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa 211-8666, Japan
tsunoo@BL.jp.nec.com
[2] NEC Software Hokuriku, Ltd.
1, Anyoji, Hakusan, Ishikawa 920-2141, Japan
{t-saito@qh,m-shigeri@pb,t-kawabata@pb}@jp.nec.com

Abstract. MISTY1 is a 64-bit block cipher that has provable security against differential and linear cryptanalysis. MISTY1 is one of the algorithms selected in the European NESSIE project, and it has been recommended for Japanese e-Government ciphers by the CRYPTREC project. This paper shows that higher order differential attacks can be successful against 6-round and 7-round versions of MISTY1 with FL functions. The attack on 6-round MISTY1 can recover a partial subkey with a data complexity of $2^{53.7}$ and a computational complexity of $2^{53.7}$, which is the smallest computational complexity for an attack on 6-round MISTY1. The attack on 7-round MISTY1 can recover a partial subkey with a data complexity of $2^{54.1}$ and a computational complexity of $2^{120.7}$, which signifies the first successful attack on 7-round MISTY1 without limiting conditions such as a weak key.

Keywords: block cipher, CRYPTREC, higher order differential attack, MISTY1, NESSIE.

1 Introduction

MISTY1 [13] has provable security against differential cryptanalysis [3] and linear cryptanalysis [12]. It is a Feistel-type block cipher that has a block length of 64 bits and a secret key length of 128 bits, and it achieves provable security by recursive repetition of the Feistel structure using two types of S-boxes, 9 bits and 7 bits in size. MISTY1 has good implementability on various platforms. It is one of the algorithms selected in the European NESSIE project [15] and was recommended for Japanese e-Government ciphers by the CRYPTREC project [4]. Moreover, it is one of the world's most widely used block ciphers.

Many methods have been applied to the cryptanalysis of MISTY1 [2, 5, 7, 8, 9, 10, 17, 18]. The strongest of those is the higher order differential attack, which is effective for ciphers with low-degree variables. MISTY1 uses S-boxes with low-degree variables to give priority to optimization in hardware.

MISTY1 with the recommended eight rounds has not yet been broken, but cryptanalysis of modified versions has been successful. The modified versions

P.J. Lee and J.H. Cheon (Eds.): ICISC 2008, LNCS 5461, pp. 415–431, 2009.

can be divided into two broad types in accordance with whether or not the FL functions were included. MISTY1 is supplemented with FL functions to counter attacks other than differential cryptanalysis and linear cryptanalysis. It is therefore important that a model that includes FL functions be evaluated for its resistance against higher order differential attacks.

The best results for cryptanalysis on MISTY1 with FL functions were achieved by a higher order differential attack on 6-round MISTY1 [18]. Using a 46th order differential characteristic found in 4-round MISTY1, the attack presented in [18] successfully recovered a partial subkey using an algebraic method with a data complexity of $2^{53.7}$ and a computational complexity of $2^{64.4}$.

In 2008, Lee et al. proposed a related-key amplified boomerang attack on 7-round MISTY1 with FL functions [10]. This technique uses a 6-round distinguisher assuming weak-key quartets to make a successful attack with a data complexity of 2^{54} and a computational complexity of $2^{55.3}$. The attack presented here, however, works on rounds 2 to 8 of MISTY1 and can be applied only to weak keys that have a 2^{-55} probability of existing. Thus, a number of limiting conditions are attached to this attack and, up to now, no attacks had been reported on 7-round MISTY1 with FL functions without these limiting conditions.

In this paper, we first make the higher order differential attack on 6-round MISTY1 with FL functions presented in [18] more efficient. This attack can recover a partial subkey in 6-round MISTY1 with a data complexity of $2^{53.7}$ and a computational complexity of $2^{53.7}$. The latter value represents the smallest computational complexity for an attack on 6-round MISTY1. Next, we apply a higher order differential attack on 7-round MISTY1 with FL functions. This attack can recover a partial subkey in 7-round MISTY1 with a data complexity of $2^{54.1}$ and a computational complexity of $2^{120.7}$ marking the first report of a successful attack on 7-round MISTY1 without limiting conditions such as a weak key.

Section 2 describes higher order differential attacks. Section 3 explains MISTY1, Section 4 describes characteristics of MISTY1, and Section 5 describes the application of such an attack to MISTY1. Section 6 concludes the paper.

2 Higher Order Differential Attacks

Here, we describe both the higher order differential characteristic and the method for solving the attack equation used in higher order differential attacks.

2.1 Higher Order Differential Characteristic

Let the encryption function be the $E(X; K)$ defined by Eq. (1) with data X and key K as input and data Y as output. Here, $X \in GF(2)^n$, $K \in GF(2)^s$, and $Y \in GF(2)^m$.

$$Y = E(X; K) \tag{1}$$

Let $(A_1, A_2, \cdots, A_d)$ be d linearly independent vectors on $GF(2)^n$, and denote the subspace of $GF(2)^d$ expanded by them as $V^{(d)}$. Then the d^{th} order differential with respect to X of $E(X;K)$ is defined by Eq. (2). Here, the symbol $\oplus$ represents an exclusive OR operation and $\bigoplus_{A\in V^{(d)}}$ the total sum by exclusive OR.

$$\varDelta^{(d)}_{V^{(d)}} E(X;K) = \bigoplus_{A\in V^{(d)}} E(X\oplus A;K) \tag{2}$$

We call the subspace $V^{(d)}$ the variable sub-blocks and the subspace other than $V^{(d)}$ the fixed sub-blocks. In the following, we abbreviate $\varDelta^{(d)}_{V^{(d)}}$ as $\varDelta^{(d)}$. If the Boolean degree of $E(X;K)$ with respect to X is N, Eq. (3) necessarily holds without dependence on X.

$$\begin{cases} \varDelta^{(N)} E(X;K) = \mathit{constant} \\ \varDelta^{(N+1)} E(X;K) = 0 \end{cases} \tag{3}$$

2.2 Attack Equations

This section explains the equations required for an attack using the higher order differential characteristic described in section 2.1. If encryption function E comprises R rounds of functions F^i $(1\le i\le R)$, the $(R-1)^{\text{th}}$ round output for input X is expressed as

$$Y^{R-1}(X) = F^{R-1}(\cdots F^1(X;K_1)\cdots;K_{R-1}), \tag{4}$$

where K_i is the subkey input in the i^{th} round. If the Boolean degree of $Y^{R-1}(X)$ with respect to X is N, Eq. (5) necessarily holds according to Eq. (3).

$$\begin{cases} \varDelta^{(N)} Y^{R-1}(X) = \mathit{constant} \\ \varDelta^{(N+1)} Y^{R-1}(X) = 0 \end{cases} \tag{5}$$

Denoting the ciphertext for input X as $C(X)$ and the function for obtaining Y^{R-1} from $C(X)$ as F^{-1}, we obtain

$$Y^{R-1}(X) = F^{-1}(C(X);K_R). \tag{6}$$

Substituting Eq. (6) into Eq. (5), we obtain

$$\begin{cases} \bigoplus_{A\in V^{(N)}} F^{-1}(C(X\oplus A);K_R) = \mathit{constant} \\ \bigoplus_{A\in V^{(N+1)}} F^{-1}(C(X\oplus A);K_R) = 0. \end{cases} \tag{7}$$

Equation (7) holds when the final round subkey K_R is correct, so the true key, K_R, can be determined by solving Eq. (7). Therefore, Eq. (7) is called the attack equation.

2.3 Algebraic Method

One method of solving the attack equation presented in section 2.2 is an algebraic method. This method regards attack equations as functions on $GF(2)$ and linearizes them. In other words, it transforms the attack equations into linear equations [14, 16]. This is accomplished by replacing the higher degree terms of the key bits with new first-degree unknown terms. This approach has the potential to reduce the computational complexity of solving attack equations.

Let Eq. (7) be an n-bit attack equation derived using a d^{th} order differential. Denoting the key contained in the attack equation as $K_R = (K_{R1}, K_{R2})$, we determine K_{R1} by an exhaustive search and obtain K_{R2} by using an algebraic method. Here, let there be L unknown coefficients in the linear equations relating to K_{R2}, and let $K_{R1} \in GF(2)^{s_1}$.

The K_{R2} for true key K_{R1} can be obtained by solving the $L \times (L+1)$ coefficient matrix obtained from L independent linear equations. Here, the coefficient matrix includes constant terms. Also, if $L+m$ linear equations for obtaining K_{R2} have been prepared, the probability of a linear equation for a false key not being inconsistent is estimated to be 2^{-m}. It is therefore possible to reject false keys by using $L+m$ linear equations that satisfy $2^{-m} \times 2^{s_1} \ll 1$.

Because the attack equation is an n-bit attack equation, n linear equations are obtained with one set of d^{th} order differentials. Therefore, the number of plaintexts needed to obtain $L+m$ linear equations is given by

$$D = 2^d \times \left\lfloor \frac{L+m}{n} \right\rfloor . \tag{8}$$

The computational complexity for obtaining n linear equations is estimated to be the total for $2^d \times (L+1)$ times of the round function. Considering, therefore, that K_{R1} is to be determined by an exhaustive search and that the time for transforming the matrix and solving the simultaneous equations is negligibly small, computational complexity T required to solve the attack equation is given by

$$T = 2^{s_1} \times 2^d \times (L+1) \times \left\lfloor \frac{L+m}{n} \right\rfloor . \tag{9}$$

Here, computational complexity signifies the number of times the round function is calculated.

Computational complexity T can be reduced provided that only w bits of the ciphertext are affected by the linear equations and the relation $w < d$ holds [11]. One attack equation (that is, n linear equations) is calculated by performing an exclusive OR operation on 2^d ciphertexts. Given the property that performing an exclusive OR operation on the same value an even number of times results in a value of 0, calculations can be omitted for values appearing an even number of times. Thus, unknown coefficients of linear equations can be calculated using only the w-bit value appearing an odd number of times resulting in computational complexity T as follows.[1] Here, as well, computational complexity signifies the number of times the round function is calculated.

$$T = 2^{s_1} \times 2^{w-1} \times (L+1) \times \left\lfloor \frac{L+m}{n} \right\rfloor \tag{10}$$

[1] There are about 2^{w-1} instances of ciphertext data on average for which the w-bit value appears an odd number of times.

3 MISTY1

This section briefly describes the structure and previous cryptanalysis of MISTY1.

3.1 Structure

MISTY1, which was proposed by Matsui in 1996, is a block cipher that has provable security against differential and linear cryptanalysis. It is a 128-bit secret key, Feistel-type block cipher with a block length of 64 bits. MISTY1 achieves provable security by recursive repetition of the Feistel structure using two types of S-boxes, 9 bits and 7 bits in size. It also uses supplementary FL functions in a Feistel structure performing logical AND/OR operations with subkeys to improve resistance to other types of attacks without losing this property of provable security. The number of MISTY1 rounds, n, can be varied in multiples of 4, but the designers recommend eight rounds.

The MISTY1 encryption function is shown in Fig. 1. The FO, FI, and FL functions that constitute MISTY1 are respectively shown in Fig. 2, Fig. 3, and Fig. 4. In this paper, bitwise AND, OR, and exclusive OR operations are denoted respectively as $\cap$, $\cup$, and $\oplus$. We denote plaintext as P and ciphertext as C. We denote the subkey input to the FO_i function as KO_i, the subkey input to the FI_{ij} function as KI_{ij}, and the subkey input to the FL_i function as KL_i.

The i^{th} round output data is defined as X^i for the internal variables. In 8-round MISTY1, $0 \le i \le 8$, so $X^0 = P$. Furthermore, we define the data that follows the FL function output to be X'^j. For 8-round MISTY1, $j \in \{0, 2, 4, 6, 8\}$, and $X'^8 = C$.

The 64-bit data Y is separated left and right into two sets of 32-bit data defined as Y_L and Y_R in accordance with the Feistel structure. Other divisions are also defined with regard to the FI function, which is an asymmetric Feistel structure:

$$Y = Y_7 \parallel Y_6 \parallel \cdots \parallel Y_1 \parallel Y_0,$$
$$Y_i \in \begin{cases} GF(2)^7 : i = even \\ GF(2)^9 : i = odd \end{cases}.$$

The $\parallel$ indicates data concatenation. We denote the i^{th} $(0 \le i < n)$ bit of Z as $Z[i]$ and the range from the i^{th} bit of Z to the j^{th} bit of Z as $Z[i-j]$. For example, the leftmost 16 bits of plaintext P is denoted as

$$P[63-48] = P_L[31-16] = P_7 \parallel P_6 = X^0_7 \parallel X^0_6.$$

The key schedule divides the 128-bit secret key into 16-bit data blocks K_i $(1 \le i \le 8)$ and uses the FI function to generate 16-bit data blocks K'_i. Table 1 shows how K_i and K'_i are used in the form of subkeys for each round.

$$SK = K_8 \parallel K_7 \parallel \cdots \parallel K_2 \parallel K_1, \;\; K_i \in GF(2)^{16}$$
$$K'_i = FI(K_i; K_{i+1 \; mod \; 8}) \;\; (i = 1, \ldots, 8)$$

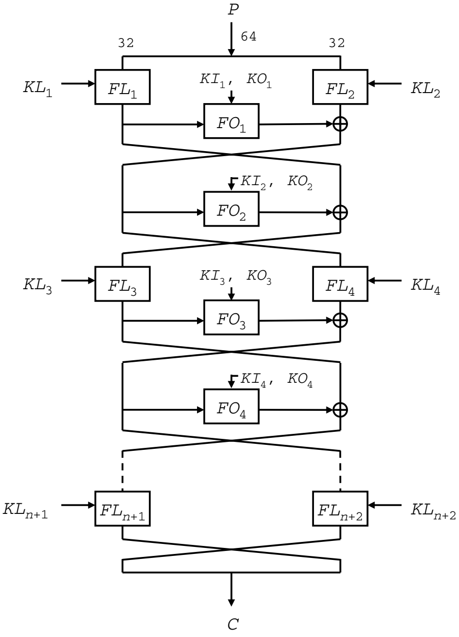


Fig. 1. MISTY1 encryption function

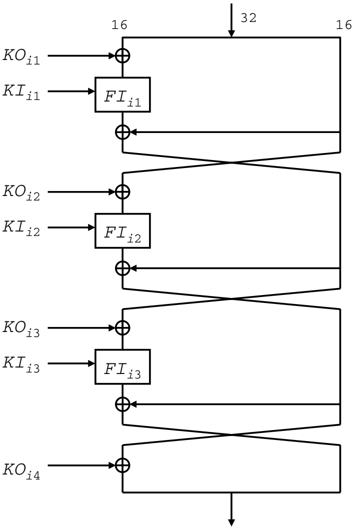


Fig. 2. FO_i function

3.2 Previous Cryptanalysis

As mentioned in the introduction, MISTY1 with the recommended eight rounds has not yet been broken, but cryptanalysis of modified versions has been

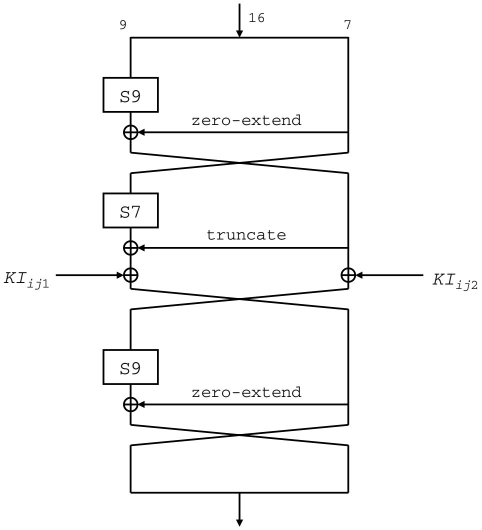


Fig. 3. FI_{ij} function

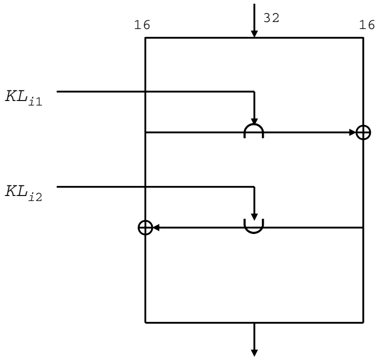


Fig. 4. FL_i function

Table 1. Use of K_i and K'_i as subkeys for each round

KO_{i1}	KO_{i2}	KO_{i3}	KO_{i4}	KI_{i1}	KI_{i2}	KI_{i3}	KL_{i1}	KL_{i2}
K_i	K_{i+2}	K_{i+7}	K_{i+4}	K'_{i+5}	K'_{i+1}	K'_{i+3}	$K_{\frac{i+1}{2}}$ (odd i) $K'_{\frac{i}{2}+2}$ (even i)	$K'_{\frac{i+1}{2}+6}$ (odd i) $K_{\frac{i}{2}+4}$ (even i)

successful. The modified versions can be broadly divided into two types depending on whether or not the FL functions were included. MISTY1 adds FL functions to counter attacks other than differential cryptanalysis and linear cryptanalysis, and a model that includes FL functions should therefore be evaluated for its resistance against higher order differential attacks. In this section, we survey past attacks on MISTY1 with FL functions.

As an attack on MISTY1 with the FL functions, Hatano et al. applied a higher order differential attack on 5-round MISTY1 in 2003 [5]. It utilized a 14^{th} order differential characteristic of 3-round MISTY1 with the FL functions to recover the key using an algebraic method. It broke 5-round MISTY1 with a data complexity of $2^{21.7}$ and a computational complexity of $2^{28.0}$. Following this, a higher order differential attack on 6-round MISTY1 was proposed in 2008 [18]. The attack presented in [18] can break the cipher with a data complexity of $2^{53.7}$ and a computational complexity of $2^{64.4}$ by using a 46^{th} order differential characteristic found in 4-round MISTY1 and recovering a partial subkey using an algebraic method. To the best of our knowledge, the attack presented in [18] constituted the strongest attack against MISTY1 with FL functions up to now.

There have also been several proposals for attacks on 6-round and 7-round MISTY1 with FL functions though under the condition of a weak key. In 2007, Tanaka et al. proposed a higher order differential attack on 6-round MISTY1 [17]. Using weak-key characteristics taking the key schedule into account, Tanaka et al. broke the cipher with a data complexity of $2^{18.9}$ and a computational complexity of $2^{80.6}$. This attack, however, can only be applied to a weak key having a probability of 2^{-32} of existing. Next, in 2008, Lee et al. proposed a related-key amplified boomerang attack on 7-round MISTY1 [10]. Using a 6-round distinguisher assuming weak-key quartets, this attack broke the cipher with a data complexity of 2^{54} and a computational complexity of $2^{55.3}$. The attack proposed by Lee et al., however, works on rounds 2 to 8 of MISTY1 and can be applied only to weak keys that have a 2^{-55} probability of existing. These attacks are successful under the assumption of a weak key, and a successful attack on 7-round MISTY1 with FL functions without limiting conditions like a weak key has yet to be reported.

4 Characteristics of MISTY1

This section describes higher order differential characteristics and the equivalent transformations of MISTY1.

4.1 Higher Order Differential Characteristics of MISTY1

We here describe higher order differential characteristics of MISTY1. First, we present from [2] Theorem 1 for MISTY1 with no FL functions. Here, α and β denote fixed and variable sub-blocks, respectively. As for sub-block size, odd-numbered sub-blocks from the left side are 9 bits while even-number sub-blocks are 7 bits.

Theorem 1. *Given a 7^{th} order differential in the form of chosen plaintext $P = (\alpha, \alpha, \alpha, \alpha, \alpha, \alpha, \alpha, \beta)$ in MISTY1 with no FL functions, intermediate data X_L^3 satisfies Eq. (11).*

$$\Delta^{(7)} X_L^3\,[31-25] = 0\text{x6d} \tag{11}$$

Theorem 1 means that the Boolean degree of $X_L^3[31-25]$ is 7. Next, we obtain Theorem 2 by extending Theorem 1 to MISTY1 with FL functions.[2]

Theorem 2. *Given a* 14^{th} *order differential in the form of* 2^{nd}*-round-input* $X^1 = (\alpha, \alpha, \alpha, \alpha, \alpha, \beta, \alpha, \beta)$ *in MISTY1 with FL functions, intermediate data* X_L^4 *satisfies Eq. (12).*[3]

$$\Delta^{(14)} X_L^4 \, [31-25] = 0 \tag{12}$$

Theorem 3 corresponds to 4-round characteristics obtained by adding one round to the upper side of Theorem 2 characteristics.

Theorem 3. *Given a* 46^{th} *order differential in the form of chosen plaintext* $P = (\alpha, \beta, \alpha, \beta, \beta, \beta, \beta, \beta)$ *in MISTY1 with FL functions, intermediate data* X_L^4 *satisfies Eq. (13).*[4]

$$\Delta^{(46)} X_L^4 \, [31-25] = 0 \tag{13}$$

4.2 Equivalent Transformations and Equivalent Keys

To simplify the analysis of the attack equations used in section 5, we define a structure for the FO function whereby subkey KO is moved and treated as equivalent subkey EKO and a structure for the FI function whereby subkey KI is moved and treated as equivalent subkey EKI. These newly defined FO and FI functions are shown in Figs. 5 and 6, respectively.

We now describe the relationship between subkeys relevant to the attack equations on the basis of key-schedule characteristics. Subkey KO_{62} and subkey KL_{82} satisfy the following relation according to Table 1.

$$KO_{62} = KL_{82} = K_8 \tag{14}$$

In addition, subkey KO_i and equivalent key EKI_i satisfy the following relation by the equivalent transformation.

$$KO_{ij} = EKI_{ij1} \parallel EKI_{ij2} \;\; (1 \le i \le 8 \;,\; 1 \le j \le 2) \tag{15}$$

Thus, Eq. (16) holds from Eqs. (14) and (15).

$$EKI_{621} \parallel EKI_{622} = KL_{82} = K_8 \tag{16}$$

Furthermore, subkeys $(KO_i,\ KI_i)$ and equivalent key EKO_i satisfy the following relations by the equivalent transformation. Here, $X \ll i$ means a left shift of i bits on data X.

$$\begin{cases} EKO_i[31-16] = A \oplus (A \ll 9) \oplus KO_{i4} \\ A = KO_{i1}[6-0] \oplus KI_{i1}[15-9] \oplus KO_{i2}[6-0] \oplus KI_{i2}[15-9] \end{cases} \tag{17}$$

$$\begin{cases} EKO_i[15-0] = B \oplus (B \ll 9) \\ B = KO_{i2}[6-0] \oplus KI_{i2}[15-9] \oplus KO_{i3}[6-0] \oplus KI_{i3}[15-9] \end{cases} \tag{18}$$

[2] Theorem 2 has been experimentally verified in [18].

[3] See Appendix A for the proof of Theorem 2.

[4] See Appendix B for the proof of Theorem 3.

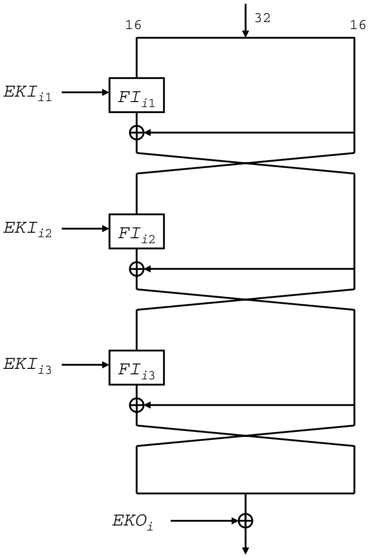


Fig. 5. Equivalent transformation of the FO_i function

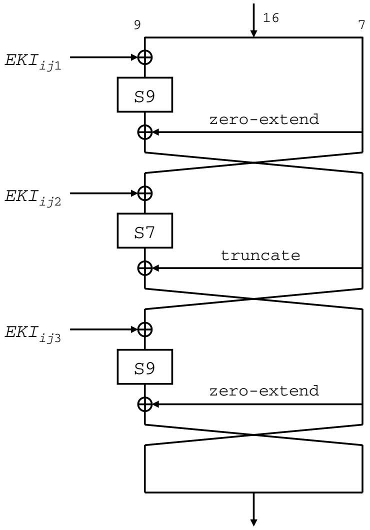


Fig. 6. Equivalent transformation of the FI_{ij} function

Thus, while equivalent key $EKO_i[31-16]$ is an unknown 16-bit variable, equivalent key $EKO_i[15-0]$ can be expressed by only unknown 7-bit variable B. As a result, the number of bits of subkey EKO_i relevant to the attack equations is essentially reduced to 23 bits.

5 Higher Order Differential Attacks on Reduced-Round MISTY1

In this section, we describe higher order differential attacks on 6-round and 7-round versions of MISTY1 with FL functions.

5.1 Attack on 6-Round MISTY1

First, we describe the attack on 6-round MISTY1 with FL functions. For this attack, we use attack equation (19) as given in [18].

$$\bigoplus_{A \in V^{(46)}} X_L'^4[i] = 0 \quad (i = 31, \cdots, 25) \tag{19}$$

Equation (19) holds only under the assumption that subkey $KL_{52}[j] = 1$ $(j = i - 16)$. However, the probability that $KL_{52}[15 - 9] = 0$, that is, that all 7-bit attack equations (19) are inconsistent is $2^{-7} < 0.01$.[5] Accordingly, at least one of those seven attack equations has a very high probability of possessing a solution.

The $X_L'^4[i]$ can be written in terms of a function F_i^{-1}, 34 bits of ciphertext C, and the 65 bits of the subkeys $TK_i = \{KO_{61}, KO_{62}, KL_8, KL_{72}[j]\}$ as follows.

$$X_L'^4[i] = F_i^{-1}(C; TK_i) \tag{20}$$

Here, $F_i^{-1} : GF(2)^{34} \times GF(2)^{65} \rightarrow GF(2)$. Now, substituting Eq. (20) in Eq. (19) gives Eq. (21).

$$\bigoplus_{A \in V^{(46)}} F_i^{-1}(C(X \oplus A); TK_i) = 0 \tag{21}$$

Thus, when $KL_{52}[j] = 1$, the linear equation has a solution for the i^{th} bit of Eq. (21), which means that the 65 bits of TK_i can be recovered while determining that $KL_{52}[j] = 1$. When $KL_{52}[j] = 0$, TK_i cannot be recovered but $KL_{52}[j] = 0$ can be determined.

The number L_i of unknown coefficients appearing in Eq. (21) can be derived by numerical analysis. In addition, if any of the estimated unknown coefficients have a linear sum relation, both data complexity and computational complexity can be reduced. Therefore, the number of unknown coefficients considering linear sum relations is derived as the independent unknown coefficients, l_i.[6] Here, subkeys EKI_{621}, EKI_{622}, and KL_{82} are all included in the attack equation. Thus, by

[5] For the case of $KL_{52}[j] = 0$, the attack equation becomes
$\bigoplus_{A \in V^{(46)}} \{X_L'^4[i] \oplus X_L'^4[j]\} = 0 \quad (i = 31, \cdots, 25\ ,\ j = i - 16)$.
Here, the number of unknown variables in the attack equation increases thereby increasing the computational complexity of the attack.

[6] Independent unknown coefficients l_i can be determined using a method described in [1].

using the relationship of Eq. (16), the number of subkey variables included in the attack equation can be reduced thereby obtaining a smaller L_i compared to that obtained in [18]. The values of L_i and l_i obtained by numerical analysis are listed in Table 2.

Table 2. Number of unknown coefficients in attack equation

	Eq. (21)		Eq. (22)	
Bit position, i	L_i	l_i	L_i	l_i
25th bit	517	173	2778	173
26th bit	494	169	2537	169
27th bit	574	189	3247	189
28th bit	494	169	2526	169
29th bit	517	173	2809	173
30th bit	527	177	2789	177
31st bit	540	181	2888	181
Total	1665	-	10944	-

The number D_i of plaintexts required to collect the linear equations for the i^{th} bit of Eq. (21) can be determined by substituting the value of l_i into Eq. (8). The number D of plaintexts required for the attack is the maximum among D_i's, and is given as follows.

$$D = D_{27} = 2^{46} \times \left\lfloor \frac{l_{27} + m}{n} \right\rfloor$$

Here, $n = 1$ since this attack solves Eq. (21) for each bit. In addition, although $s_1 = 1$ assuming $KL_{52}[j]$, the key value is taken to be 1 to solve Eq. (21), which means that $s_1 = 0$ in effect. Thus, letting $m = 10$ from the condition $2^{-m} \times 2^{s_1} \ll 1$,[7] we get

$$D = 2^{46} \times (189 + 10) \approx 2^{53.64}.$$

Now, to calculate the coefficients of the attack equation, $(FL_8^{-1} + 2 \times S9)$, $(FL_8^{-1} + 2 \times S7)$, and $FL_7^{-1}/2$ must be calculated. These coefficients can be calculated in 18, 14, and 2 bit widths, respectively, thereby satisfying the relation $w < d$. Required computational complexity T_1 can therefore be obtained by substituting the total number of unknown coefficients L_i in Eq. (10). Assuming that the processing times for an FL function and an S-box lookup are equivalent, we get

$$\begin{aligned} T_1 &= \{3 \times (2^{17} + 2^{13}) + 1/2 \times 2^1\} \times 1666 \times (189 + 10) \\ &\approx 2^{37.02}. \end{aligned}$$

[7] $10^{-3} \ll 1$.

This computational complexity signifies the number of times an S-box is looked up. Since S-boxes are looked up nine times per FO function, we get the following for T_1 on converting it to number of 6-round encryptions.

$$T_1 \approx \frac{2^{37.02}}{9 \times 6} \approx 2^{31.27}$$

Computational complexity T_1 as estimated by Eq. (10) was significantly reduced compared to estimating it by Eq. (9) in [18]. Given that computational complexity T required to break the cipher is the sum of the time needed to collect data and the time for calculating coefficients, we get the following expression for T.

$$T = D + T_1 \approx 2^{53.64}$$

5.2 Attack on 7-Round MISTY1

We here describe an attack on 7-round MISTY1 with FL functions. The structure of 7-round MISTY1 targeted by this attack is shown in Fig. 7. The attack equation is given by Eq. (22).

$$\bigoplus_{A \in V^{(46)}} F_i^{-1}(FO_7^{-1}(C(X \oplus A); EK_7); TK_i) = 0 \quad (i = 31, \cdots, 25) \qquad (22)$$

This attack moves equivalent key EKO_7 as shown in Fig. 7. We recover the 75 bits of EKI_7 by an exhaustive search and EKO_7 by an algebraic method. The number of coefficients L_i and l_i that appear in Eq. (22) are given in Table 2. Here, equivalent key EKO_7 can be reduced to 23 bits as shown in section 4.2. The number D of plaintexts required to collect the linear equations of Eq. (22) is given as follows for $m = 85$.

$$D = 2^{46} \times (189 + 85) \approx 2^{54.10}$$

We note here that a 7-round attack additionally requires calculations for one round of decryption. Denoting the computational complexity for one round of decryption as T_2, computational complexity T' required for calculating the coefficients of the linear equations is given as follows.

$$T' = 2^{75} \times (T_1 + T_2)$$

Here, T_1 is as follows.

$$\begin{aligned} T_1 &= \{3.5 \times (2^{17} + 2^{13}) + 1/2 \times 2^1\} \times 10945 \times (189 + 85) \\ &\approx 2^{40.42} \end{aligned}$$

Also, while T_2 is the processing time for performing one round of decryption D times, we can think of it as being divided into calculation time for the FO_7 function and that for the exclusive-OR operation on the output of that function. Since calculations for the FO_7 function can be completed in 2^{32} times at most,

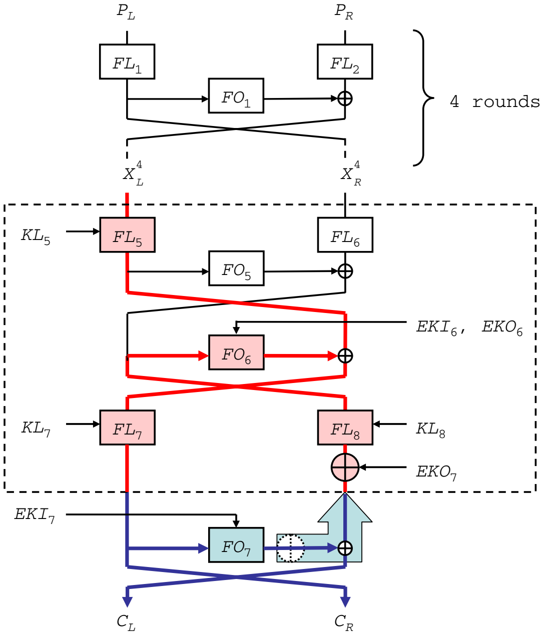


Fig. 7. Seven-round MISTY1

and considering that $D \gg 2^{32}$, T_2 can be said to approximate the time for performing an exclusive-OR operation D times. Thus, from $T_1 \ll T_2$, T' becomes as follows.

$$T' \approx 2^{75} \times T_2 \approx 2^{129.10}$$

This computational complexity signifies the number of times an exclusive-OR operation is performed. Assuming that processing time is equivalent for one S-box lookup and three exclusive-OR operations, the calculation time for one round of encryption is equivalent to the processing time of 50 exclusive-OR operations. Thus, converting T' to number of 7-round encryptions, we get

$$T' \approx \frac{2^{129.10}}{50 \times 7} \approx 2^{120.65}.$$

Since computational complexity T required to break the cipher is the sum of the time needed to collect data and the time for determining coefficients, we get the following for T.

$$T = D + T' \approx 2^{120.65}$$

6 Conclusion

This paper reported on higher order differential attacks on 6-round and 7-round versions of MISTY1 with FL functions. Our attacks can recover a partial subkey in 6-round MISTY1 with a data complexity of $2^{53.7}$ and a computational complexity of $2^{53.7}$ and a partial subkey in 7-round MISTY1 with a data complexity of $2^{54.1}$ and a computational complexity of $2^{120.7}$.

As shown by the results in Table 3, the attack on 6-round MISTY1 presented in this paper achieves the smallest computational complexity for an attack on 6-round MISTY1, and the attack on 7-round MISTY1 represents the first report of a successful attack on 7-round MISTY1 without any limiting conditions such as a weak key.

The recommended number of rounds for MISTY1 is eight, so it must be noted that the method described in this paper is not a direct threat to the security of MISTY1. At the same time, the results presented here suggest that the security margin of MISTY1 has been reduced to only one round.

Table 3. Results of attacks on MISTY1

FO rounds	*FL* layers	Key conditions	Data complexity	Computational complexity	Comments
4	3	-	$2^{22.25}$	2^{45}	Slicing attack [9]
4	3	-	25	2^{27}	Integral attack [7]
5	3	-	2^{34}	2^{48}	Integral attack [7]
5	4	-	$2^{21.7}$	$2^{28.0}$	HOD attack [5]
6	4	Weak key	$2^{18.9}$	$2^{80.6}$	HOD attack [17]
6	4	-	$2^{53.7}$	$2^{64.4}$	HOD attack [18]
6	4	-	$2^{53.7}$	$2^{53.7}$	HOD attack (this paper)
7*	3	Weak key	2^{54}	$2^{55.3}$	RKAB attack [10]
7	4	-	$2^{54.1}$	$2^{120.7}$	HOD attack (this paper)

HOD attack : Higher order differential attack.
RKAB attack : Related-key amplified boomerang attack.
*The attack in [10] works on rounds 2 to 8 of MISTY1.

References

1. Aoki, K.: Practical Evaluation of Security Against Generalized Interpolation Attack. IEICE Transactions 83-A(1), 33–38 (2000)
2. Babbage, S., Frisch, L.: On MISTY1 higher order differential cryptanalysis. In: Won, D. (ed.) ICISC 2000. LNCS, vol. 2015, pp. 22–36. Springer, Heidelberg (2001)
3. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-Like Cryptosystems. J. Cryptology 4(1), 3–72 (1991)
4. Cryptography Research and Evaluation Committees (CRYPTREC), `http://www.cryptrec.jp/english/about.html`
5. Hatano, Y., Tanaka, H., Kaneko, T.: Optimization for the Algebraic Method and Its Application to an Attack of MISTY1. IEICE Transactions 87-A(1), 18–27 (2004)
6. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)

7. Knudsen, L.R., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
8. Kühn, U.: Cryptanalysis of Reduced-Round MISTY. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 325–339. Springer, Heidelberg (2001)
9. Kühn, U.: Improved Cryptanalysis of MISTY1. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 61–75. Springer, Heidelberg (2002)
10. Lee, E., Kim, J., Hong, D., Lee, C., Sung, J., Hong, S., Lim, J.: Weak-Key Classes of 7-Round MISTY 1 and 2 for Related-Key Amplified Boomerang Attacks. IEICE Transactions 91-A(2), 642–649 (2008)
11. Lucks, S.: The Saturation Attack - A Bait for Twofish. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 1–15. Springer, Heidelberg (2002)
12. Matsui, M.: Linear Cryptanalysis of the Data Encryption Standard. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
13. Matsui, M.: New Block Encryption Algorithm MISTY. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 54–68. Springer, Heidelberg (1997)
14. Moriai, S., Shimoyama, T., Kaneko, T.: Higher Order Differential Attak of CAST Cipher. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 17–31. Springer, Heidelberg (1998)
15. New European Schemes for Signature, Integrity, and Encryption (NESSIE), `https://www.cosic.esat.kuleuven.be/nessie/`
16. Shimoyama, T., Moriai, S., Kaneko, T., Tsujii, S.: Improved Higher Order Differential Attack and Its Application to Nyberg-Knudsen's Designed Block Cipher. IEICE Transactions 82-A(9), 1971–1980 (1999)
17. Tanaka, H., Hatano, Y., Sugio, N., Kaneko, T.: Security Analysis of MISTY1. In: Kim, S., Yung, M., Lee, H.-W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 215–226. Springer, Heidelberg (2008)
18. Tsunoo, Y., Saito, T., Nakashima, H., Shigeri, M.: Higher Order Differential Attack on 6-Round MISTY1. IEICE Transactions 92-A(1) (to appear, 2009)

A Proof of Theorem 2

Proof. An FL function is a Feistel structure consisting only of logical operations, and variable sub-blocks at the input and output of the FL_3 function can be given by the following bit-wise relations.

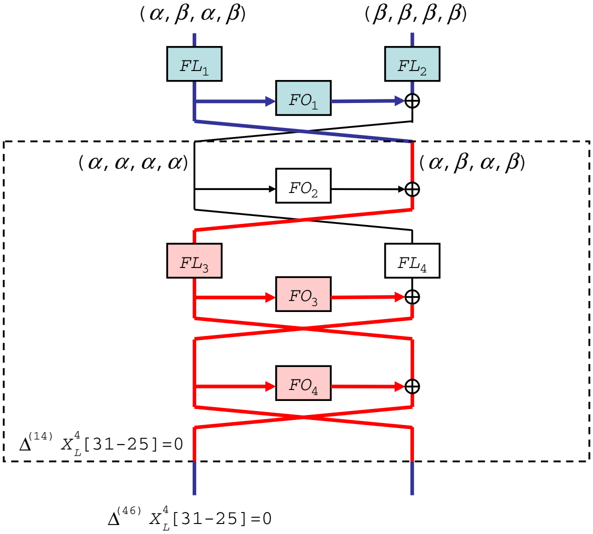
$$\begin{cases} X_4'^2[i] = X_4^2[i] \oplus (KL_{31}[i] \cap X_6^2[i]) \\ X_6'^2[i] = X_6^2[i] \oplus (KL_{32}[i] \cup X_4'^2[i]) \end{cases} \quad (i = 6, \cdots, 0)$$

Fixed sub-blocks can be expressed by bit-wise relations in the same way. Accordingly, since both variable sub-blocks and fixed sub-blocks are bijective, a 14th order differential in the form of $X'^2 = (\alpha, \beta, \alpha, \beta, \alpha, \alpha, \alpha, \alpha)$ can be given via the FL_3 function and FL_4 function by giving a 14th order differential in the form of $X'^1 = (\alpha, \alpha, \alpha, \alpha, \alpha, \beta, \alpha, \beta)$. Thus, since the value of the 7th order differential of Eq. (11) appears 2^7 times in intermediate data X_L^4 according to Theorem 1, its total sum is zero. □

B Proof of Theorem 3

Proof. As in the proof of Theorem 2, both the variable sub-blocks and fixed sub-blocks at the input and output of the FL_1 are bijective thereby preserving the 14th order differential. In addition, though 2^{14} values of random data are output from the FO_1 function, a 46th order differential in the form of $X'^1 = (\beta, \beta, \beta, \beta, \alpha, \beta, \alpha, \beta)$ can be given by giving 1st-round input X_R^0 exhaustively. Thus, since the value of the 14th order differential of Eq. (12) appears 2^{32} times in intermediate data X_L^4 according to Theorem 2, its total sum is zero. □

Figure 8 shows the higher order differential characteristic of Theorem 3. The section within the dotted line shows the higher order differential characteristic of Theorem 2.



$(\alpha, \beta, \alpha, \beta)$
$(\beta, \beta, \beta, \beta)$
FL_1
FL_2
FO_1
$(\alpha, \alpha, \alpha, \alpha)$
$(\alpha, \beta, \alpha, \beta)$
FO_2
FL_3
FL_4
FO_3
FO_4
$\Delta^{(14)} X_L^4[31\text{-}25]=0$
$\Delta^{(46)} X_L^4[31\text{-}25]=0$

Fig. 8. Higher order differential characteristics of MISTY1

Author Index