Lecture Notes in Computer Science 3797

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

Editorial Board

David Hutchison
*Lancaster University, UK*
Takeo Kanade
*Carnegie Mellon University, Pittsburgh, PA, USA*
Josef Kittler
*University of Surrey, Guildford, UK*
Jon M. Kleinberg
*Cornell University, Ithaca, NY, USA*
Friedemann Mattern
*ETH Zurich, Switzerland*
John C. Mitchell
*Stanford University, CA, USA*
Moni Naor
*Weizmann Institute of Science, Rehovot, Israel*
Oscar Nierstrasz
*University of Bern, Switzerland*
C. Pandu Rangan
*Indian Institute of Technology, Madras, India*
Bernhard Steffen
*University of Dortmund, Germany*
Madhu Sudan
*Massachusetts Institute of Technology, MA, USA*
Demetri Terzopoulos
*New York University, NY, USA*
Doug Tygar
*University of California, Berkeley, CA, USA*
Moshe Y. Vardi
*Rice University, Houston, TX, USA*
Gerhard Weikum
*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

Subhamoy Maitra C.E. Veni Madhavan
Ramarathnam Venkatesan (Eds.)

# Progress in Cryptology – INDOCRYPT 2005

6th International Conference on Cryptology in India
Bangalore, India, December 10-12, 2005
Proceedings

Springer

Volume Editors

Subhamoy Maitra
Indian Statistical Institute, Applied Statistics Unit
203 B T Road, Kolkata 700 108, India
E-mail: subho@isical.ac.in

C.E. Veni Madhavan
Indian Institute of Science (IISc)
Department of Computer Science and Automation (CSA)
Bangalore 560 012, India
E-mail: cevm@csa.iisc.ernet.in

Ramarathnam Venkatesan
Microsoft Corporation
One Microsoft Way, Redmond, WA 98052, USA
E-mail: venkie@microsoft.com

Library of Congress Control Number: 2005936804

CR Subject Classification (1998): E.3, G.2.1, D.4.6, K.6.5, K.4, F.2.1-2, C.2

| | |
|---|---|
| ISSN | 0302-9743 |
| ISBN-10 | 3-540-30805-9 Springer Berlin Heidelberg New York |
| ISBN-13 | 978-3-540-30805-8 Springer Berlin Heidelberg New York |

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11596219 06/3142 5 4 3 2 1 0

# Preface

Indocrypt began in the year 2000 under the leadership of Bimal Roy and Indocrypt 2005 was the sixth conference in this series. This series has been well accepted by the international research community as a forum for presenting high-quality cryptography research. This year a total of 148 papers were submitted for consideration to the Program Committee and after a careful review process, 31 were accepted for presentation. We would like to thank the authors of all submitted papers, including those that were accepted and those which, unfortunately, could not be accommodated.

The reviewing process for Indocrypt was very stringent and the schedule was extremely tight. The Program Committee members did an excellent job in reviewing and selecting the papers for presentation. During the review process, the Program Committee members were communicating using a review software developed by Bart Preneel, Wim Moreau and Joris Claessens. We acknowledge them for providing the software. The software was hosted at I2R, Singapore and we are grateful to Feng Bao and Jianying Zhou for allowing that. This year's conference was deeply indebted to Qiu Ying of I2R, Singapore, who took the responsibility of maintaining the review software and the server. Without his great cooperation Indocrypt 2005 could not have been possible. In this regard I would like to acknowledge the support of Tanmoy Kanti Das, Dibyendu Chakrabarti, Mridul Nandi, Deepak Kumar Dalai, Sumanta Sarkar and Sourav Mukhopadhyay for handling important administrative issues in the submission and review processes as well as for putting together these proceedings in their final form. We are also grateful to Palash Sarkar for his cooperation and guidance in Indocrypt 2005.

The proceedings include the revised versions of the 31 selected papers. Revisions were not checked by the Program Committee and the authors bear the full responsibility for the contents of the respective papers. Our thanks go to all the Program members and the external reviewers (a list of them is included in the proceedings) who put in their valuable time and effort in providing important feedback to the authors. We thank V. Kumar Murty of the University of Toronto for kindly agreeing to present the invited talk. The talk has been included in the proceedings.

The organization of the conference involved many individuals. We express our heartfelt gratitude to the general Co-chairs K. Gopinath and Soumyendu Raha for taking care of the actual hosting of the conference. They were ably assisted by Kumar Swamy H.V. and the members of the Organizing Committee. There was also invaluable secretarial support from CAS Laboratory, Informatics Laboratory (CSA department) and IMI at the Indian Institute of Science, Bangalore. Finally, we would like to acknowledge Springer for active cooperation and timely production of the proceedings.

December 2005

Subhamoy Maitra
C. E. Veni Madhavan
Ramarathnam Venkatesan

# Organization

Indocrypt 2005 was organized by the Indian Institute of Science, Bangalore, in collaboration with the Cryptology Research Society of India.

## General Co-chairs

| | |
|---|---|
| K. Gopinath | Indian Institute of Science, Bangalore |
| Soumyendu Raha | Indian Institute of Science, Bangalore |

## Program Co-chairs

| | |
|---|---|
| Subhamoy Maitra | Indian Statistical Institute, Kolkata, India |
| C. E. Veni Madhavan | Indian Institute of Science, Bangalore, India |
| Ramarathnam Venkatesan | Microsoft Corporation, Redmond, USA |

## Program Committee

| | |
|---|---|
| V. Arvind | Institute of Mathematical Sciences Chennai, India |
| R. Balasubramanian | Institute of Mathematical Sciences Chennai, India |
| Feng Bao | Institute for Infocomm Research, Singapore |
| Alex Biryukov | K. U. Leuven (ESAT, COSIC group), Belgium |
| Xavier Boyen | Voltage Security, USA |
| Mathieu Ciet | Gemplus International, SA, France |
| Abhijit Das | Indian Institute of Technology, Kharagpur, India |
| Anand Desai | NTT, MCL, USA |
| Josep Domingo-Ferrer | Rovira i Virgili University of Tarragona, Catalonia |
| Orr Dunkelman | Technion, Israel |
| Caroline Fontaine | Universitè des Sciences et Technologies de Lille - CNRS, France |
| Sugata Gangopadhyay | Indian Institute of Technology, Roorkee, India |
| Guang Gong | University of Waterloo, Canada |

| | |
|---|---|
| Kishan Chand Gupta | University of Waterloo, Canada |
| Tor Helleseth | University of Bergen, Norway |
| David Jao | Microsoft Research, USA |
| Andrew Klapper | University of Kentucky, USA |
| Kaoru Kurosawa | Ibaraki University, Japan |
| Tanja Lange | Technical University of Denmark, Denmark |
| John Malone-Lee | University of Bristol, UK |
| Francoise Levy-dit-Vehel | Ècole Nationale Supérieure des Techniques Avancées, France |
| Yucel D. Melek | METU Cryptology Institute, Turkey |
| Pradeep Kumar Mishra | University of Calgary, Canada |
| François Morain | Laboratoire d'Informatique de l' Ècole Polytechnique, France |
| Harald Niederreiter | National University of Singapore, Singapore |
| Rafail Ostrovsky | UCLA, USA |
| Enes Pasalic | Technical University of Denmark, Denmark |
| Josef Pieprzyk | Macquarie University, Australia |
| Zulfikar Ramzan | NTT DoCoMo USA Labs, USA |
| Pandu Rangan | Indian Institute of Technology Madras, India |
| Bimal Roy | Indian Statistical Institute, Kolkata, India |
| Rei Safavi-Naini | University of Wollongong, Australia |
| Pramod Saxena | SAG, India |
| Abhi Shelat | IBM, Zurich, Switzerland |
| Pante Stanica | Auburn University, Montgomery, USA |
| Berk Sunar | Worcester Polytechnic Institute, USA |
| Yuriy Tarannikov | Moscow State University, Russia |
| Moti Yung | Columbia University, USA |
| Jianying Zhou | Institute for Infocomm Research, Singapore |

## External Referees

Michel Abdalla
Masayuki Abe
Raju Agarwal
Carlos Aguilar
Mehdi-Laurent Akkar
Ersan Akyildiz
Shirish Altekar
Tomoyuki Asano
Daniel Augot
Joonsang Baek
Selcuk Baktir
Sverine Baudry
S.S. Bedi
Kamel Bentahar
Raghav Bhaskar
Lilian Bohy
Alexandra Boldyreva
Colin Boyd
Eric Brier
Philippe Bulens
Christophe De Canniere
Jordi Castellà-Roca
Julien Cathalo
Liqun Chen
Joe Cho
Scott Contini
Thomas W. Cusick
Tanmoy Kanti Das
Guerric Meurice de Dormale
Ali Doianaksoy
R. Dupont
Sylvain Duquesne
Adam Elbirt
Haining Fan
Matthew J. Fanto
Pooya Farshim
Benot Feix
Eiichiro Fujisaki

Navneet Gaba
Pierrick Gaudry
Indeevar Gupta
Martin Hell
Javier Herranz
Katrin Hoeper
Susan Hohenberger
Laurent Imbert
Tetsu Iwata
Dimitar P. Jetchev
Shaoquan Jiang
Marc Joye
Pascal Junod
Sarvjeet Kaur
Seluk Kavut
Alexander Kholosha
Takeshi Koshiba
Hugo Krawczyk
Meena Kumari
Noboru Kunihiro
F. Laguillaumie
Joseph Lano
Cedric Lauradoux
Benoit Libert
Moses Liskov
Phil MacKenzie
Frederic Magniez
Dahlia Malkhi
C.R. Mandal
Antoni M.-Balleste
John Erik Mathiassen
Matthew McKague
Alfred Menezes
Jonathan Millen
Sara Miner More
Sourav Mukhopadhyay
Yassir Nawaz
Kazuto Ogawa
Alina Oprea
Elisabeth Oswald
Ferruh Özbudak
Daniel Page
S.K. Pal
Kapil Paranjape
Matthew Parker
Maura Paterson
Roger Patterson
Chris Peikert
Ludovic Perret
Rajesh Pillai
Constantin Popescu
Bart Preneel
Duong Quang Viet
Hvard Raddum
Sibabrata Ray
Jeremie Roland
Pieter Rozenhart
Amit Sahai
Erkay Savas
Hovav Schacham
Kai Schramm
Francesc Sebé
Igor Semaev
Rozenhart Seong
Han Shin
Francesco Sica
Adam Smith
Agustí Solanas
M.C. Srivastava
Martijn Stam
Franois-Xavier Standaert
Ron Steinfeld
Makoto Sugita
Keisuke Tanaka
Adrian Tang
Nicolas Theriault
E. Thomé
Dongvu Tonien
Patrick Traynor
Eran Tromer
Gary Walsh
Huaxiong Wang
Bogdan Warinschi
Eric Wegrzynowski
Kjell Wooding
David Woodruff
Pratibha Yadav
Tugrul Yanik
Bulent Yener
Nam Yul Yu
Huafei Zhu

# Table of Contents

# Abelian Varieties and Cryptography

V. Kumar Murty

Department of Mathematics, University of Toronto,
40 St. George Street, Toronto, ON M5S 3G3, Canada
murty@math.toronto.edu

**Abstract.** Let $A$ be an Abelian variety over a finite field $\mathbb{F}$. The possibility of using the group $A(\mathbb{F})$ of points on $A$ in $\mathbb{F}$ as the basis of a public-key cryptography scheme is still at an early stage of exploration. In this article, we will discuss some of the issues and their current staus. In particular, we will discuss arithmetic on Abelian varieties, methods for point counting, and attacks on the Discrete Logarithm Problem, especially those that are peculiar to higher-dimensional varieties.

## 1 Introduction

Let $A$ be an Abelian variety over a finite field $\mathbb{F}$. Thus $A$ is a smooth projective algebraic variety defined over $\mathbb{F}$ on which there is an algebraic group operation, also defined over $\mathbb{F}$. In particular, the identity element $\mathcal{O}$ of the group is an $\mathbb{F}$-rational point. Abelian varieties of dimension one are called elliptic curves.

The possibility of using the group $A(\mathbb{F})$ of points on $A$ in $\mathbb{F}$ as the basis of a public-key cryptography scheme is still at an early stage of exploration. In this article, we will discuss some of the issues and their current staus. In particular, we will discuss the problem of explicit and efficient arithmetic, algorithms for efficient point counting, and criteria by which to eliminate cryptographically weak Abelian varieties.

In order to keep our discussion to a moderate length, we shall merely outline or draw attention to the many developments in this subject. We shall try to emphasize those aspects in which we believe more work is needed.

Denote by $\overline{\mathbb{F}}$ an algebraic closure of $\mathbb{F}$ and let

$$G \;=\; \mathrm{Gal}(\overline{\mathbb{F}}/\mathbb{F})$$

be the Galois group. It is a procyclic group, being the inverse limit of cyclic groups:

$$G \;\simeq\; \hat{\mathbb{Z}} \;=\; \lim \mathbb{Z}/N\mathbb{Z}.$$

Let Frob $=$ $\mathrm{Frob}_{\mathbb{F}}$ be the map

$$x \;\mapsto\; x^q$$

where $q$ is the number of elements in $\mathbb{F}$. Sometimes, we may also write $\mathrm{Frob}_q$. It is a topological generator of $G$.

© Springer-Verlag Berlin Heidelberg 2005

There is an action of $G$ on $A(\overline{\mathbb{F}})$. In particular, the function

$$n \ \mapsto \ \deg(\mathrm{Frob} - n)$$

is well defined. There is a polynomial $P_A(T)$ with the property that for every $n$ (sufficiently large),

$$P_A(n) \ = \ \deg(\mathrm{Frob} - n).$$

This is called the characteristic polynomial of the Frobenius automorphism. It has many wonderful properties. In particular,

$$|A(\mathbb{F})| = \ P_A(1).$$

Moreover, if $d$ is the dimension of $A$,

$$P_A(T) \ = \ \prod_{i=1}^{2d} (1 - \omega_i T)$$

where

$$|\omega_i| \ = \ q^{\frac{1}{2}}$$

and for $1 \le i \le d$,

$$\omega_i \omega_{d+i} \ = \ q.$$

We see from this that

$$|A(\mathbb{F})| \ = \ q^d \ + \ \mathbf{O}(q^{d-\frac{1}{2}}).$$

Since Abelian varieties of higher dimension have more points (roughly $q^d$ where $d$ is the dimension), a generic attack should take about

$$q^{d/2}$$

steps. This means that it may be possible to use them as the basis of a secure cryptographic scheme with a smaller value of $q$. Thus, for example, from this point of view, a two-dimensional Abelian variety over a field of approximate size $2^{82}$ would be as secure as an elliptic curve over a field of approximate size $2^{164}$.

To realize this in practice, we have to solve several problems:

– Explicit and efficient arithmetic
– Efficient point counting
– Understanding of other attacks that are peculiar to this setting.

## 2 Explicit and Efficient Arithmetic

For explicit and efficient arithmetic, most effort has been directed at elliptic curves. The state of the art in efficient implementations of arithmetic of elliptic curves over finite fields is given in the book [12]. It should be noted that some of this work is, in fact, about improving the efficiency of arithmetic in finite fields. These results can of course be applied directly in the higher dimensional case as well.

In the higher dimensional case, we have already pointed out that the larger group order ostensibly allows us to work securely in a finite field of smaller size. However, there are two difficulties. Firstly, the theory of Abelian varieties in higher dimensions has not, for the most part, been developed from the point of view of explicit equations or explicit arithmetic. Much work remains to be done in this regard. Secondly, even where one is able to explicitly give equations, the number of variables tends to be large and this adds complexity to the algorithm. In general, this added complexity seems to offset any gain that might be had by working over a field of smaller size.

One class of Abelian varieties for which these problems have been studied extensively is that of Jacobians of hyperelliptic curves. In this case, there has been significant progress in developing efficient arithmetic. The general algorithm of Cantor gives formulae for the addition of points on such Jacobians [4]. A considerable amount of work by many authors (including Chao, Gonda, Guajardo, Guyot, Harley, Kaveh, Kuroki, Lange, Matsuo, Nagao, Paar, Patankar, Pelzl, Tsujii, Wollinger and others) has been done on refining this algorithm to improve the complexity. The standard by which such work is compared is the speed relative to the known implementations for comparable elliptic curve arithmetic.

For Abelian varieties that are Jacobians of hyperelliptic curves of genus 3, the work of Guyot, Kaveh and Patankar [11] shows that in some cases, the arithmetic is faster than comparable elliptic curve arithmetic. Their work builds on the explicit formula method of Tanja Lange and others. It should be noted that in making this comparison, the authors took into account the index calculus attack of Thériault [22] on Jacobians of genus three hyperelliptic curves.

There has also been progress on the arithmetic of Abelian varieties that arise as the Jacobian of more general curves. There is a general treatment due to Arita, Miura and Sekiguchi [2].

## 3 Point Counting

For the problem of point counting, there are fast algorithms in the case of hyperelliptic Jacobians over fields of small characteristic (work of Satoh[20], Fouquet, Gaudry and Harley[8], Kedlaya[14], Denef and Vercauteren[7], and others).

For the case of a general Abelian variety, there is only a baby step-giant step approach to point counting. Gaudry and Harley [10] observed that if one knew the number of points modulo an integer $m$, this can be sped up by a factor of $\sqrt{m}$. An interesting result of Chao, Matsuo and Tsujii [5] was that this could be improved if we knew the entire characteristic polynomial of Frobenius $P_A(T)$ modulo $m$. This work was refined by Izadi and the author [13].

As an illustration of this, consider the case $d = 3$ (where $d$ is the dimension of $A$). The Gaudry-Harley algorithm costs

$$\mathbf{O}(q^{5/4}/m^{1/2})$$

steps. The algorithm of Chao-Matsuo-Tsujii costs

$$\mathbf{O}(q^{3/2}/m)$$

steps. The refined algorithm in [13] gives a cost of

$$\mathbf{O}(q^{5/4}/m)$$

steps.

Much further work is needed to develop practical techniques of point counting for general Abelian varieties.

## 4 Primality of the Group Order

The next question that arises is how likely it is that $\#A(\mathbb{F}_q)$ is prime or nearly prime. It will be interesting to estimate this as we vary over all Abelian varieties of a fixed dimension over a fixed finite field. A related problem is to consider a fixed Abelian variety over a number field and its reductions modulo various primes. Let us briefly discuss the latter problem. It is a difficult one even in the case of elliptic curves.

More precisely, consider an elliptic curve $E$ over the rational numbers $\mathbb{Q}$. There is the following result of Ali Miri and the author [1]. Let $E$ be an elliptic curve over $\mathbb{Q}$. Assuming the Generalized Riemann Hypothesis(GRH) (for all Dedekind zeta functions), we have that

$$|E(\mathbb{F}_p)|$$

has $\log\log p$ prime divisors for a set of primes of density 1. Since $\log\log p$ grows *very* slowly with $p$, this is bounded in cryptographic ranges.

The Generalized Riemann Hypothesis is the assertion that the non-trivial zeros of the zeta function $\zeta_F(s)$ of a number field $F$ are on the critical line $Re(s) = \frac{1}{2}$. This hypothesis is often introduced because it helps us to control the error terms when counting prime ideals that satisfy certain splitting conditions. In turn, certain natural Galois representations allow us to relate group orders of Abelian varieties to the number of prime ideals with prescribed splitting conditions.

In some cases, it is possible to dispense with the GRH by using sieve methods. For example, in the case that $E$ has complex multiplication, the above result has been proved unconditionally by Cojocaru [6].

Note that there is a conjecture of Koblitz that asserts that $\#E(\mathbb{F}_p)$ should be prime for

$$\sim c_E \frac{x}{(\log x)^2}$$

of the primes $p \leq x$ where $c_E > 0$ is a constant depending on $E$. He made this conjecture in analogy with the conjectures of Hardy and Littlewood about primes of the form $2p + 1$. Koblitz's conjecture is still open. The first progress towards the conjecture of Koblitz was the result of Ali Miri and the author [13]. There it was shown that assuming the GRH, there are

$$\gg \frac{x}{(\log x)^2}$$

primes $p \leq x$ such that $\#E(\mathbb{F}_p)$ has atmost 13 prime divisors.

This used the lower bound Selberg sieve method. The result has been improved by Steuding and Weng [21] who showed (using the weighted sieve) that 13 can be replaced by 8. In the case that $E$ has complex multiplication, these results have been proved unconditionally and refined by Cojocaru [6]. In particular, she shows that 13 can be replaced by 6 in the CM case.

For elliptic curves without complex multiplication, we can assert the existence of large prime power divisors for a positive proportion of the primes. More precisely, we have the following result due to Ram Murty and the author.

**Theorem 1.** *(Murty-Murty).*
*Let $E$ be an elliptic curve defined over the rationals which does not have complex multiplication. Assume the Generalized Riemann Hypothesis (GRH) for Dedekind zeta functions. Then, for a positive proportion of the primes $p$,*

$$|E(\mathbb{F}_p)|$$

*has a prime power divisor $> p^{1/5-\epsilon}$.*

Note that $|E(\mathbb{F}_p)|$ is roughly of size $p$.

*Outline of Proof.* Let us set

$$N_p \;=\; \#E(\mathbb{F}_p).$$

Then by the Weil bound,

$$N_p \;=\; p \;+\; \mathbf{O}(p^{\frac{1}{2}}).$$

Thus, by the prime number theorem,

$$\sum_{p \leq x} \log N_p \;\sim\; x.$$

On the other hand, the sum on the left is also equal to

$$\sum_{d \leq x} \Lambda(d)\pi(x,d)$$

where $\Lambda(d)$ is the usual von Mangoldt function and

$$\pi(x,d) \;=\; \#\{p \leq x : N_p \;\equiv\; 0 \mod d\}.$$

Assuming the GRH and using the Chebotarev density theorem, we have

$$\pi(x,d) \;=\; \frac{1}{d}\pi(x) \;+\; \mathbf{O}(d^{3/2}x^{1/2}\log dNx)$$

where $N$ is the conductor of $E$. This means that

$$\sum_{d \le x^{\frac{1}{5}-\epsilon}} \Lambda(d)\pi(x,d) \;=\; \pi(x)(\frac{1}{5}-\epsilon)\log x \;+\; \mathbf{O}(x^{1-\epsilon}).$$

Hence

$$\sum_{x^{\frac{1}{5}-\epsilon} \le d \le x} \Lambda(d)\pi(x,d) \;\sim\; (\frac{4}{5}+\epsilon)x.$$

Since the left hand side is

$$\sum_{p \le x} \sum_{\substack{x^{\frac{1}{5}-\epsilon} \le d \le x \\ d | N_p}} \Lambda(d) \;\le\; (\log x) \sum_{p \le x} \sum_{\substack{x^{\frac{1}{5}-\epsilon} \le d \le x \\ d | N_p \\ d \text{ prime power}}} 1,$$

we deduce that for a set of primes $p$ of density at least

$$\frac{4}{5}$$

$N_p$ has a prime power divisor $> p^{\frac{1}{5}-\epsilon}$.

## 5 Splitting of Abelian Varieties

A phenomenon that is peculiar to the higher dimensional case is that of "splittting modulo all primes". It is possible to have a simple (or absolutely simple) Abelian variety defined over a number field which has the property that with only finitely many exceptions, when it is reduced modulo a prime (ideal), it factors into Abelian varieties of smaller dimension. In particular, the group order will not be prime. By the usual attacks, this makes such an Abelian variety not optimal for cryptographic purposes.

This phenomenon of course cannot occur for elliptic curves. But it already occurs in the two dimensional case, that is for Abelian surfaces. In particular, let $A$ be an Abelian surface that has endomorphisms by an indefinite quaternion division algebra over $\mathbb{Q}$. At all but finitely many primes $p$, the reduction $A_p$ modulo $p$ is of the form

$$A_p \;\sim\; E_p \;\times\; E_p$$

where $E_p$ is an elliptic curve over the residue field. Thus, even though $A$ is simple globally, it splits everywhere locally.

This is the geometric analogue of a phenomenon that has been known for a long time in the context of polynomials. For example, the polynomial $T^4+1$ is irreducible over $\mathbb{Q}$ but factors modulo $p$ for every prime $p$.

This failure of the "local-global principle" was studied in [17] and in the thesis of Patankar [19]. Much further investigation is needed here to identify which Abelian varieties have this property.

## 6 The Weil and Tate Pairings

Let $\hat{A}$ denote the dual Abelian variety. The first pairing to consider is one that comes from the cup product:

$$< \cdot, \cdot >: A[m] \ \times \ \hat{A}[m] \ \longrightarrow \ \mu_m.$$

This is the Weil pairing and it is a non-degenerate pairing. In particular, if $P$ is a point in $A[m]$ rational over $\mathbb{F}$, then there is a point $R \in \hat{A}[m]$ rational over $\mathbb{F}$ such that

$$< P, R > \neq 1.$$

Now, if $Q$ is a point in $A[m]$ with $Q = rP$, then

$$< Q, R > = < rP, R > = < P, R >^r .$$

Thus, if the pairing $< \cdot, \cdot >$ can be computed efficiently, and if $R$ can be found efficiently, then the Discrete Logarithm problem on $A(\mathbb{F})$ can be transferred to one in $\mu_m$. For the latter, there are subexponential algorithms available.

This is the basis of the Menezes-Okamato-Vanstone [15] attack. They considered the case of elliptic curves. In this case, $\hat{E} \ = \ E$ and we have a self-pairing

$$E[m] \ \times \ E[m] \ \longrightarrow \ \mu_m$$

that is alternating and non-degenerate.

Using the isomorphism

$$E[m] \ \simeq \ (\mathbb{Z}/m)^2,$$

the above pairing is the exterior square map. Indeed, fix a basis $P, Q$ say of $E[m]$. For $T_1, T_2 \in E[m]$, write

$$T_i \ = \ a_i P \ + \ b_i Q.$$

Then

$$< T_1, T_2 > = \ \det \begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix} .$$

In this case there is an efficient algorithm for computing the Weil pairing due to Miller [16]. We shall return to this later.

Frey and Rück [9] have indicated that a different pairing can be used in a similar way. Suppose that $m$ is prime to the characteristic of $\mathbb{F}$ and suppose that the $m$-th roots of unity are in $\mathbb{F}$. They define a pairing

$$A(\mathbb{F})/mA(\mathbb{F}) \ \times \ A(\mathbb{F})[m] \ \longrightarrow \ \mathbb{F}^{\times}/\mathbb{F}^{\times m} \ \simeq \ \mu_m.$$

Frey and Rück call this the Lichtenbaum-Tate pairing (or just the Tate pairing for short). The method of Miller allows for the computation of this pairing as well in the case of elliptic curves.

## 7 Computation of Pairings

Let $E/\mathbb{F}_q$ be an elliptic curve (where $q$ is a power of the prime $p$). Let $gcd(m,p) = 1$. Denote by $\mathrm{Div}^0(E)$ the abelian group of divisors of degree zero on $E$.Two such divisors, $D_1$ and $D_2$ say, are said to be *linearly equivalent* (written $D_1 \sim D_2$) if their difference is the divisor of a rational function on $E$. There is an isomorphism

$$E \simeq \mathrm{Div}^0(E)/\sim$$

given by

$$P \mapsto \text{the class of } (P) - (\mathcal{O}).$$

For $P, Q \in E[m]$, take $D_P, D_Q \in \mathrm{Div}^0(E)$ with $D_P \sim (P) - (\mathcal{O})$ and $D_Q \sim (Q) - (\mathcal{O})$. Let $f_P, f_Q$ be rational functions such that $\mathrm{div}(f_P) = mD_P$, $\mathrm{div}(f_Q) = mD_Q$. Suppose that $D_P$ and $D_Q$ have disjoint supports. Then the *Weil pairing* is given by

$$< P, Q >= \frac{f_P(D_Q)}{f_Q(D_P)},$$

The *Tate pairing* can also be described using $f_P(D_Q)$. We must assume that $\mathbb{F}$ contains the $m$-th roots of unity. The pairing

$$T : E(\mathbb{F})[m] \times E(\mathbb{F})/mE(\mathbb{F}) \longrightarrow \mathbb{F}^\times/\mathbb{F}^{\times m}$$

is given by

$$T(P, Q) = f_P(D_Q) \bmod mE(\mathbb{F}).$$

Miller's algorithm provides an efficient method to compute $f_P(D_Q)$. According to this algorithm, one begins by randomly picking $R$, and forming

$$D_P = (P + R) - (R).$$

If

$$\mathrm{div}(f_k) = k(P + R) - k(R) - (kP) + (\mathcal{O})$$

then $f_m = f_P$.

We can compute $f_m$ inductively as follows. For $R, S \in E$, let us denote by $h_{R,S} = 0$ the straight line through $R, S$. Let us also denote by $h_S = 0$ the vertical line through $S$.
Then

$$\mathrm{div}(h_{k_1P,k_2P}) = (k_1P) + (k_2)P + (-(k_1 + k_2)P) - 3\mathcal{O}$$

and

$$\mathrm{div}(h_{(k_1+k_2)P}) = ((k_1 + k_2)P) + (-(k_1 + k_2)P) - 2\mathcal{O}$$

and so

$$f_{k_1+k_2} = \frac{f_{k_1} f_{k_2} h_{k_1P,k_2P}}{h_{(k_1+k_2)P}}.$$

The initial conditions are $f_0 = 1$ and

$$f_1 = \frac{h_{P+R}}{h_{P,R}}.$$

Thus, the algorithm is as follows:

```
INPUTS: $m = \sum_{i=0}^{t} b_i 2^i, S \in E$
OUTPUT: $f = f_m(S)$.

  $f \leftarrow f_1$; $Z \leftarrow P$;
  For $j \leftarrow t-1, t-2, \ldots, 1, 0$ do
    $f \leftarrow f^2 \frac{h_{Z,Z}(S)}{h_{2Z}(S)}$; $Z \leftarrow 2Z$;
    If $b_j = 1$ then
      $f \leftarrow f_1 f \frac{h_{Z,P}(S)}{h_{Z+P}(S)}$; $Z \leftarrow Z + P$;
    Endif
  Endfor

Return $f$
```

There have been refinements and improvements of this basic algorithm in varous settings due to many authors including Barreto, Eisenträger, Galbraith, Harrison, Kim, Lauter, Lynn, Montgomery, Scott and Soldera. In recent joint work with Ian Blake and Greg Xu[3], we have discovered some refinements of Miller's algorithm that apply in general. Our approach works for arbitrary finite fields and saves $\log_2 m$ field multiplications. A variant for finite fields of characteristic three saves $\log_3 m$ field multiplications. (In this case, $\log_3 m$ of point triplings are performed which can be done very efficiently). We expect that similar calculations should work whenever one has an effective Riemann-Roch theorem.

## 8 Attacks on the Abelian Variety Discrete Logarithm Problem Using Pairings

Let us return to the Tate pairing. Work of Lichtenbaum and Tate shows that this is a non-degenerate pairing. To use it for the Discrete Logarithm problem, one tries to find a point $R \in A(\mathbb{F})$ such that the map

$$A[m] \longrightarrow \mu_m$$

given by

$$P \mapsto < R, P >$$

is an isomorphism. One then uses this map as with the Weil pairing to solve the Discrete Logarithm problem. For the Discrete Logarithm problem, the essential point is that there is an embedding of a large cyclic subgroup of $A(\mathbb{F})$ into $\mu_m$ (or more precisely, into the multipicative group $F^{\times}$) where one can use index calculus methods to mount a subexponential attack.

This approach is very succesful for supersingular Abelian varieties. The reason is that in this case, the eigenvalues of Frobenius are of the form

$$q^{\frac{1}{2}} \zeta$$

where $\zeta$ is a root of unity. Since the eigenvalues lie in an extension field of $\mathbb{Q}$ of degree $\leq 2d$, this bounds the order of $\zeta$. For example, for an elliptic curve, we see that $\zeta^2$ lies in a quadratic field. So if $\zeta$ is an $m$-th root of unity, then

$$\phi(m/(2,m)) \ \leq \ 2.$$

This means that $m \leq 6$. Thus, all the eigenvalues are (after normalization) roots of a cyclotomic polynomial. In particular, $|A(\mathbb{F}_q)|$ (or atleast its exponent) divides $q^k - 1$ for some $k$ that depends only on dim $A$. Thus if $m$ divides this order, then $m$ divides $q^k - 1$ and one applies the Tate pairing over the field $\mathbb{F}_{q^k}$.

If one tries to apply this attack in general, the problem is that there is no good bound for $k$. However, one might consider Abelian varieties that are "almost supersingular" in the following sense. Let $L$ be the splitting field of the characteristic polynomial $P_A(T)$ of Frobenius. Choose a prime $\mathfrak{p}$ of $L$ above $p$. Consider the set of slopes

$$\mathit{Slopes}(A) \ = \ \{\mathrm{ord}_{\mathfrak{p}}\alpha : P_A(\alpha) = 0\}.$$

This set is independent of the choice of prime $\mathfrak{p}$ because $L$ is Galois over $\mathbb{Q}$. Define also the length of each slope: for $c \in \mathit{Slopes}(A)$, set

$$\mathit{length}(c) \ = \ \#\{\alpha : \mathrm{ord}_{\mathfrak{p}}\alpha = c\}$$

where $\alpha$ ranges over zeros of $P_A(T)$. A supersingular Abelian variety $A$ can be characterized by

$$\mathit{Slopes}(A) \ = \ \{\frac{1}{2}\ \}$$

and

$$\mathit{length}(\frac{1}{2}) \ = \ 2d.$$

An *almost supersingular Abelian variety A* (or what Zarhin [23] calls Abelian varieties of K3-type) can be defined as one for which

$$\mathit{Slopes}(A) \ = \ \{0, 1, \frac{1}{2}\}$$

with

$$\mathit{length}(0) \ = \ \mathit{length}(1) \ = \ 1$$

and

$$\mathit{length}(\frac{1}{2}) \ = \ 2d - 2.$$

For example, consider

$$A \ = \ E_1 \ \times \ E_2$$

where $E_1$ is a ordinary elliptic curve and $E_2$ is a supersingular elliptic curve. The Discrete Logarithm Problem here can be solved in

$$\mathbf{O}(q^{\frac{1}{2}+\epsilon})$$

steps. This is not subexponential but is much better than the generic square root attack which in this case would take

$$\mathbf{O}(q)$$

steps. Can one use pairings on almost supersingular Abelian varieties to get an attack on DLP that is better than the square root attack?

## References

1. S. A. Miri and V. Kumar Murty, An application of sieve methods to elliptic curves, in: INDOCRYPT 2001, pp. 91-98, *Lecture Notes in Computer Science* 2247, Springer, Berlin, 2001.
2. S. Arita, S. Miura, T. Sekiguchi, An addition algorithm on the Jacobian varieties of curves, *J. Ramanujan Math. Soc.*, **19**(2004), 235-251.
3. I. Blake, V. Kumar Murty and G. Xu, Refinements of Miller's algorithm for computing the Weil/Tate pairing, *J. Algorithms*, to appear.
4. D. Cantor, Computing in the Jacobian of a hyperelliptic curve, *Math. Comp.*, **48**(1987), 95-101.
5. J. Chao, K. Matsuo, S. Tsujii, Baby step giant step algorithms in point counting of hyperelliptic curves, IEICE Trans. Fundamentals, E86-A, **4**(2003).
6. A. Cojocaru, Bounded number of prime factors for the orders of the reductions of a CM elliptic curve, preprint, 2004.
7. J. Denef and F. Vercauteren, An extension of Kedlaya's algorithm to Artin-Schreier curves in characteristic 2, in: ANTS-V, pp. 308-323 eds. C. Fieker and D. Kohel, *Lecture Notes in Computer Science* 2369, Springer-Verlag, 2002.
8. M. Fouquet, P. Gaudry and R. Harley, An extension of Satoh's algorithm and its implementation, *J. Ramanujan Math. Soc.*, **15**(2000), 281-318.
9. G. Frey and H. Rück, A remark concerning $m$-divisibility and the discrete logarithm in the divisor class group of curves, *Math. Comp.*, **62**(1994), 865-874.
10. P. Gaudry and R. Harley, Counting points on hyperelliptic curves over finite fields, in: ANTS-IV, pp. 297-312, ed. W. Bosma, *Lecture Notes in Computer Science* 1838, Springer-Verlag, 2000.
11. C. Guyot, K. Kaveh and V. Patankar, Explicit algorithm for the arithmetic on the hyperelliptic Jacobians of genus 3, *J. Ramanujan Math. Soc.*, **19**(2004), 75-115.
12. D. Hankerson, A. Menezes and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, New York, 2004.
13. F. Izadi and V. Kumar Murty, Counting points on an Abelian variety over a finite field, in: INDOCRYPT 2003, pp. 323-333, eds. T. Johansson and S. Maitra, *Lecture Notes in Computer Science* 2904, Springer, 2004.
14. K. Kedlaya, Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology, *J. Ramanujan Math. Soc.*, **16**(2001), 323-338. See also Errata, **18**(2003), 417-418.
15. A. Menezes, T. Okamoto and S. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, *IEEE Trans. Inform. Theory*, **39(5)**(1993), 1639-1646.
16. V. Miller, The Weil pairing and its efficient calculation, *J. Cryptology*, **17**(2004), 235-261.
17. V. Kumar Murty, Splitting of Abelian varieties: a new local-global problem, in: *Algebra and Number Theory*, ed. R. Tandon, Hindustan Book Agency, Delhi, 2005.

18. D. Mumford, *Abelian Varieties*, Oxford.
19. V. Patankar, Splitting of Abelian varieties, Ph.D Thesis, University of Toronto, 2005.
20. T. Satoh, The canonical lift of an ordinary elliptic curve over a finite field and its point counting, *J. Ramanujan Math. Soc.*, **15**(2000), 247-270.
21. J. Steuding and A. Weng, On the number of prime divisors of the order of elliptic curves modulo $p$, *Acta Arith.*, **117**(2005), 341-352.
22. N. Thériault, Index calculus attack for hyperelliptic curves of small genus, in: ASIACRYPT 2003, pp. 75-92, *Lecture Notes in Computer Science* 2894, Springer-Verlag, New York, 2003.
23. Y. Zarhin, Abelian varieties of K3-type and $\ell$-adic representations, in: Algebraic Geometry and Analytic Geometry, pp. 231-255, Springer-Verlag, Tokyo, 1991.

# Proof of a Conjecture on the Joint Linear Complexity Profile of Multisequences

Harald Niederreiter[1] and Li-Ping Wang[2]

[1] Department of Mathematics, National University of Singapore, 2 Science Drive 2, Singapore 117543, Republic of Singapore
nied@math.nus.edu.sg

[2] Temasek Laboratories, National University of Singapore, 5 Sports Drive 2, Singapore 117508, Republic of Singapore
tslwlp@nus.edu.sg

**Abstract.** We prove a conjecture on the asymptotic behavior of the joint linear complexity profile of random multisequences over a finite field. This conjecture was previously shown only in the special cases of single sequences and pairs of sequences. We also establish an asymptotic formula for the expected value of the $n$th joint linear complexity of random multisequences over a finite field. These results are relevant for the theory of word-based stream ciphers.

**Keywords:** Word-based stream ciphers, multisequences, joint linear complexity, joint linear complexity profile.

## 1 Introduction

The linear complexity is an important tool in the system-theoretic approach to the quality assessment of keystreams in stream ciphers (see [19]). The linear complexity measures to what extent the keystream can be simulated by linear feedback shift register sequences or, equivalently, by linear recurring sequences. Ideally, the keystream should be a "truly random" sequence. Thus, the yardstick in the assessment of keystreams by means of linear complexity is the behavior of random sequences with respect to the linear complexity.

In practice, keystreams will be bit sequences, i.e., sequences of elements of the binary field $\mathbb{F}_2$. Since it does not make any difference in the mathematical treatment, we will consider more generally sequences of elements of a finite field $\mathbb{F}_q$ of order $q$, where $q$ is an arbitrary prime power. It is convenient to speak of a sequence over $\mathbb{F}_q$ when we mean a sequence of elements of $\mathbb{F}_q$.

Recent developments in stream ciphers point towards an interest in word-based or vectorized stream ciphers (see e.g. [2], [4], [10], and the proposals DRAGON and NLS to the ECRYPT stream cipher project [6]). The theory of such stream ciphers requires the study of multisequences, i.e., of parallel streams of finitely many sequences. In the framework of linear complexity theory, the appropriate complexity measure for multisequences is the joint linear complexity defined below. We denote an $m$-fold multisequence consisting of $m$ parallel

© Springer-Verlag Berlin Heidelberg 2005

streams of sequences $S_1, \dots, S_m$ over $\mathbb{F}_q$ by $\mathbf{S} = (S_1, \dots, S_m)$ and we speak of an $m$-fold multisequence over $\mathbb{F}_q$. Here $m$ is an arbitrary positive integer.

**Definition 1.** *Let $n$ be a positive integer and let $\mathbf{S} = (S_1, \dots, S_m)$ be an $m$-fold multisequence over $\mathbb{F}_q$. Then the* $n$th joint linear complexity $L_n^{(m)}(\mathbf{S})$ *of* $\mathbf{S}$ *is the least order of a linear recurrence relation over $\mathbb{F}_q$ that simultaneously generates the first $n$ terms of each sequence $S_j$, $j = 1, 2, \dots, m$. The sequence $L_1^{(m)}(\mathbf{S}), L_2^{(m)}(\mathbf{S}), \dots$ of nonnegative integers is called the* joint linear complexity profile *of* $\mathbf{S}$.

We always have $0 \leq L_n^{(m)}(\mathbf{S}) \leq n$ and $L_n^{(m)}(\mathbf{S}) \leq L_{n+1}^{(m)}(\mathbf{S})$. Note that the definition of $L_n^{(m)}(\mathbf{S})$ makes sense also if each $S_j$, $j = 1, 2, \dots, m$, is a finite sequence containing at least $n$ terms. This remark will be used later on, for instance in equation (1).

The joint linear complexity and the joint linear complexity profile of multisequences have received a lot of attention recently. Feng, Wang, and Dai [8], Xing [22], and Xing, Lam, and Wei [23] constructed multisequences with special joint linear complexity profiles. Wang, Zhu, and Pei [21] discussed algorithmic aspects of the joint linear complexity profile. The papers by Fu, Niederreiter, and Su [9], Meidl [12], and Meidl and Niederreiter [13] are devoted to the study of the joint linear complexity of periodic multisequences. Meidl and Winterhof [14] proved bounds for the joint linear complexity profile of a special family of multisequences. Probabilistic results on the joint linear complexity profile of multisequences were obtained by Feng and Dai [7] and Wang and Niederreiter [20]. Dai, Imamura, and Yang [3] considered aspects of the asymptotic behavior of the joint linear complexity profile of multisequences. A recent survey article on linear complexity, which includes material on the joint linear complexity, is Niederreiter [16]. For earlier work we refer to the books of Cusick, Ding, and Renvall [1] and Ding, Xiao, and Shan [5].

For the assessment of the quality of multisequences for word-based stream ciphers, we need to know the behavior of the joint linear complexity profile of random multisequences over $\mathbb{F}_q$. According to a folklore conjecture (see [17], [20], [22]), a random $m$-fold multisequence $\mathbf{S}$ over $\mathbb{F}_q$ satisfies

$$\lim_{n\to\infty} \frac{L_n^{(m)}(\mathbf{S})}{n} = \frac{m}{m+1}$$

in a sense that will be made precise in Section 2. The only cases in which this conjecture was proved so far are $m = 1$ (see Niederreiter [15]) and $m = 2$ (see Wang and Niederreiter [20]). The main contribution of the present paper is to prove this conjecture for all values of $m$.

## 2 The Results

We first need to set up an appropriate stochastic model for studying the joint linear complexity profile of random $m$-fold multisequences over $\mathbb{F}_q$. The assumptions underlying this stochastic model are the following: (i) strings (i.e., finite sequences) over $\mathbb{F}_q$ of the same length are equiprobable; (ii) corresponding terms in the $m$ streams making up an $m$-fold multisequence over $\mathbb{F}_q$ are statistically independent. The constructions of the following probability measures reflect these assumptions.

Let $\mathbb{F}_q^m$ be the set of $m$-tuples of elements of $\mathbb{F}_q$ and let $(\mathbb{F}_q^m)^\infty$ be the sequence space over $\mathbb{F}_q^m$, i.e., the set of (infinite) sequences with terms from $\mathbb{F}_q^m$. It is obvious that the set $(\mathbb{F}_q^m)^\infty$ can be identified with the set of $m$-fold multisequences over $\mathbb{F}_q$, and henceforth we will use this identification. Let $\mu_{q,m}$ be the uniform probability measure on $\mathbb{F}_q^m$ which assigns the measure $q^{-m}$ to each element of $\mathbb{F}_q^m$. Furthermore, let $\mu_{q,m}^\infty$ be the complete product measure on $(\mathbb{F}_q^m)^\infty$ induced by $\mu_{q,m}$.

For a property $\mathcal{P}$ of $m$-fold multisequences $\mathbf{S} \in (\mathbb{F}_q^m)^\infty$, we write Prob($\mathcal{P}$) for the $\mu_{q,m}^\infty$-measure of the set of all $\mathbf{S} \in (\mathbb{F}_q^m)^\infty$ that have the property $\mathcal{P}$ (provided this set is $\mu_{q,m}^\infty$-measurable). Of particular interest are those properties $\mathcal{P}$ for which Prob($\mathcal{P}$) $= 1$ since these can be viewed as typical properties of a random $m$-fold multisequence over $\mathbb{F}_q$. We say that a property $\mathcal{P}$ holds *with probability 1* if Prob($\mathcal{P}$) $= 1$.

The following theorem establishes the conjecture formulated in Section 1 and gives also a rigorous enunciation in terms of the terminology introduced above.

**Theorem 1.** *For any prime power $q$ and any integer $m \geq 1$ we have*

$$\lim_{n\to\infty} \frac{L_n^{(m)}(\mathbf{S})}{n} = \frac{m}{m+1}$$

*with probability 1.*

As mentioned in Section 1, Theorem 1 was previously known only in the cases $m = 1$ and $m = 2$. Theorem 1 implies an asymptotic expression for the expected value $E_n^{(m)}$ of the $n$th joint linear complexity of random $m$-fold multisequences over $\mathbb{F}_q$. Since the $n$th joint linear complexity depends only on the first $n$ terms of the $m$ streams making up an $m$-fold multisequence over $\mathbb{F}_q$, we can write $E_n^{(m)}$ in the form

$$E_n^{(m)} = q^{-mn} \sum_{\mathbf{T}\in(\mathbb{F}_q^m)^n} L_n^{(m)}(\mathbf{T}), \tag{1}$$

where $(\mathbb{F}_q^m)^n$ is the set of strings of length $n$ with terms from $\mathbb{F}_q^m$. For $m = 1$ it was shown by Rueppel [18–Chapter 4] (see also [19–Section 3.2]) that

$$E_n^{(1)} = \frac{n}{2} + O(1) \qquad \text{as } n \to \infty.$$

For $m = 2$ and $q = 2$ it was proved by Feng and Dai [7] and for $m = 2$ and arbitrary $q$ it was shown by Wang and Niederreiter [20] that

$$E_n^{(2)} = \frac{2n}{3} + O(1) \qquad \text{as } n \to \infty.$$

The following result holds for arbitrary $q$ and $m$.

**Theorem 2.** *For any prime power $q$ and any integer $m \geq 1$ we have*

$$E_n^{(m)} = \frac{mn}{m+1} + o(n) \qquad \text{as } n \to \infty.$$

## 3 The Proofs

We recall some notation from [20]. For any integers $m \geq 1$ and $L \geq 1$, let $P(m;L)$ be the set of $m$-tuples $\mathbf{I} = (i_1, \ldots, i_m) \in \mathbb{Z}^m$ with $i_1 \geq i_2 \geq \cdots \geq i_m \geq 0$ and $i_1 + \cdots + i_m = L$. For any $\mathbf{I} = (i_1, \ldots, i_m) \in P(m;L)$, let $\lambda(\mathbf{I})$ be the number of positive entries in $\mathbf{I}$. Then $\mathbf{I}$ can be written in the form

$$\mathbf{I} = (\underbrace{i_1, \ldots, i_{s_{\mathbf{I},1}}}_{s_{\mathbf{I},1}}, \underbrace{i_{s_{\mathbf{I},1}+1}, \ldots, i_{s_{\mathbf{I},1}+s_{\mathbf{I},2}}}_{s_{\mathbf{I},2}}, \ldots, \underbrace{i_{s_{\mathbf{I},1}+\cdots+s_{\mathbf{I},t-1}+1}, \ldots, i_{s_{\mathbf{I},1}+\cdots+s_{\mathbf{I},t}}}_{s_{\mathbf{I},t}},$$
$$\ldots, \underbrace{i_{s_{\mathbf{I},1}+\cdots+s_{\mathbf{I},\mu(\mathbf{I})-1}+1}, \ldots, i_{s_{\mathbf{I},1}+\cdots+s_{\mathbf{I},\mu(\mathbf{I})}}}_{s_{\mathbf{I},\mu(\mathbf{I})}}, \underbrace{i_{\lambda(\mathbf{I})+1}, \ldots, i_m}_{s_{\mathbf{I},\mu(\mathbf{I})+1}}),$$

where $i_{s_{\mathbf{I},1}+\cdots+s_{\mathbf{I},t-1}+1} = \cdots = i_{s_{\mathbf{I},1}+\cdots+s_{\mathbf{I},t}} > i_{s_{\mathbf{I},1}+\cdots+s_{\mathbf{I},t}+1}$ for $1 \leq t \leq \mu(\mathbf{I})$, $i_{\lambda(\mathbf{I})+1} = \cdots = i_m = 0$, and $\lambda(\mathbf{I}) = s_{\mathbf{I},1} + \cdots + s_{\mathbf{I},\mu(\mathbf{I})}$. If $\lambda(\mathbf{I}) = m$, then $s_{\mathbf{I},\mu(\mathbf{I})+1} = 0$. Furthermore, we define

$$c(\mathbf{I}) = \prod_{i=1}^{\lambda(\mathbf{I})} \frac{(q^{m+1-i}-1)(q^i-1)}{q-1},$$

$$d(\mathbf{I}) = \prod_{j=1}^{\mu(\mathbf{I})} \prod_{i=1}^{s_{\mathbf{I},j}} \frac{q^i-1}{q-1}.$$

Put

$$e_{\lambda(\mathbf{I})} = 2 \times (0,1,2,\ldots,\lambda(\mathbf{I}),0,\ldots,0) \in \mathbb{Z}^{m+1}.$$

For $\mathbf{I} \in P(m;L)$, let $[\mathbf{I}, n-L]$ denote the vector obtained by arranging the $m+1$ numbers between the square brackets in nonincreasing order. Let $\cdot$ denote the standard inner product for vectors with $m+1$ components. As in [20], we define $b(\mathbf{I}, n-L)$ as follows. If $0 \leq n-L < i_{\lambda(\mathbf{I})}$, then we put

$$b(\mathbf{I}, n-L) = e_{\lambda(\mathbf{I})} \cdot [\mathbf{I}, n-L] - \frac{\lambda(\mathbf{I})(\lambda(\mathbf{I})-1)}{2}.$$

If $i_{s_{\mathbf{I},1}+\cdots+s_{\mathbf{I},w+1}} \leq n-L < i_{s_{\mathbf{I},1}+\cdots+s_{\mathbf{I},w}}$ for some integer $w$ with $1 \leq w \leq \mu(\mathbf{I})-1$, then

$$b(\mathbf{I}, n-L) = e_{\lambda(\mathbf{I})} \cdot [\mathbf{I}, n-L] - \left(\frac{\lambda(\mathbf{I})(\lambda(\mathbf{I})+1)}{2} - (s_{\mathbf{I},1} + \cdots + s_{\mathbf{I},w})\right).$$

Finally, if $n-L \geq i_1$, then

$$b(\mathbf{I}, n-L) = e_{\lambda(\mathbf{I})} \cdot [\mathbf{I}, n-L] - \frac{\lambda(\mathbf{I})(\lambda(\mathbf{I})+1)}{2}.$$

For integers $m \geq 1$, $n \geq 1$, and $0 \leq L \leq n$, let $N_n^{(m)}(L)$ be the number of $\mathbf{T} \in (\mathbb{F}_q^m)^n$ with $L_n^{(m)}(\mathbf{T}) = L$. Here $(\mathbb{F}_q^m)^n$ has the same meaning as in (1). It is trivial that $N_n^{(m)}(0) = 1$. For $1 \leq L \leq n$ we have the formula

$$N_n^{(m)}(L) = \sum_{\mathbf{I} \in P(m;L)} \frac{c(\mathbf{I})}{d(\mathbf{I})} q^{b(\mathbf{I},n-L)} \tag{2}$$

which is obtained from [20–eq. (11) and Theorem 2].

We start the proof of Theorem 1 with the following elementary inequality.

**Lemma 1.** *Let $x_1 \geq x_2 \geq \cdots \geq x_m$ be real numbers. Then*

$$\sum_{k=1}^{m} (k-1)x_k \leq \frac{m-1}{2} \sum_{k=1}^{m} x_k.$$

*Proof.* We proceed by induction on $m$. The inequality is trivial for $m = 1$. Suppose that the inequality is shown for $m$ numbers and consider now $m+1$ numbers. By the induction hypothesis and simple steps, we obtain

$$\begin{aligned}\sum_{k=1}^{m+1} (k-1)x_k &= \sum_{k=1}^{m} (k-1)x_k + mx_{m+1} \\ &\leq \frac{m-1}{2} \sum_{k=1}^{m} x_k + mx_{m+1} = \frac{m-1}{2} \sum_{k=1}^{m+1} x_k + \frac{m+1}{2} x_{m+1} \\ &\leq \frac{m-1}{2} \sum_{k=1}^{m+1} x_k + \frac{1}{2} \sum_{k=1}^{m+1} x_k = \frac{m}{2} \sum_{k=1}^{m+1} x_k,\end{aligned}$$

and the induction is complete. □

In the next two lemmas we present upper bounds on $N_n^{(m)}(L)$. In the cases $m = 1$ and $m = 2$ we have better results in the sense of convenient closed-form expressions for $N_n^{(m)}(L)$ (see [20]).

**Lemma 2.** *For any prime power $q$ and any integers $m \geq 1$ and $n \geq 1$ we have*

$$N_n^{(m)}(L) \leq q^{(m+1)L} \qquad \text{for } 0 \leq L \leq n.$$

*Proof.* Since the result is trivial for $L = 0$, we can assume that $1 \leq L \leq n$. If $L$ is the $n$th joint linear complexity of the $m$-fold multisequence $\mathbf{S} = (S_1, \ldots, S_m)$ over $\mathbb{F}_q$, then the first $n$ terms of $S_1, \ldots, S_m$ are determined by a simultaneous linear recurrence relation of order $L$ and by the $mL$ initial values for this linear recurrence relation (i.e., $L$ initial values for each of the $m$ sequences $S_1, \ldots, S_m$). Since there are $q^L$ possibilities for a linear recurrence relation of order $L$ over $\mathbb{F}_q$ and $q^{mL}$ possibilities for the $mL$ initial values from $\mathbb{F}_q$, the desired result follows. □

**Lemma 3.** *For any prime power $q$ and any integers $m \geq 1$ and $n \geq 1$ we have*

$$N_n^{(m)}(L) \leq C(q,m)L^m q^{2mn-(m+1)L} \qquad \text{for } 1 \leq L \leq n,$$

*with a constant $C(q,m)$ depending only on $q$ and $m$.*

*Proof.* Note that for any $\mathbf{I} \in P(m;L)$ we have $d(\mathbf{I}) \geq 1$ and $c(\mathbf{I}) \leq C_1(q,m)$ with a constant $C_1(q,m)$ depending only on $q$ and $m$. Therefore in view of (2),

$$N_n^{(m)}(L) \leq C_1(q,m) \sum_{\mathbf{I} \in P(m;L)} q^{b(\mathbf{I},n-L)}. \tag{3}$$

Now fix $\mathbf{I} = (i_1, \ldots, i_m) \in P(m;L)$. Observe that

$$b(\mathbf{I}, n-L) \leq e_{\lambda(\mathbf{I})} \cdot [\mathbf{I}, n-L], \tag{4}$$

where $e_{\lambda(\mathbf{I})} = (0, 2, 4, \ldots, 2\lambda(\mathbf{I}), 0, \ldots, 0) \in \mathbb{Z}^{m+1}$ and $[\mathbf{I}, n-L]$ is the vector with $m+1$ components obtained by rearranging $i_1, i_2, \ldots, i_m, n-L$ in nonincreasing order. Since the last $m - \lambda(\mathbf{I})$ components of $[\mathbf{I}, n-L]$ are 0, we have

$$e_{\lambda(\mathbf{I})} \cdot [\mathbf{I}, n-L] = e_m \cdot [\mathbf{I}, n-L],$$

and so by (4) we obtain

$$b(\mathbf{I}, n-L) \leq e_m \cdot [\mathbf{I}, n-L]. \tag{5}$$

Suppose that $n-L$ is in position $r$ in the vector $[\mathbf{I}, n-L]$, i.e.,

$$i_1 \geq i_2 \geq \cdots \geq i_{r-1} \geq n-L \geq i_r \geq \cdots \geq i_m.$$

Then

$$\begin{aligned} e_m \cdot [\mathbf{I}, n-L] &= 2\sum_{k=1}^{r-1}(k-1)i_k + 2(r-1)(n-L) + 2\sum_{k=r}^{m} k i_k \\ &= 2\sum_{k=1}^{m}(k-1)i_k + 2(r-1)(n-L) + 2\sum_{k=r}^{m} i_k \\ &\leq 2\sum_{k=1}^{m}(k-1)i_k + 2(r-1)(n-L) + 2(m-r+1)(n-L) \\ &= 2\sum_{k=1}^{m}(k-1)i_k + 2m(n-L). \end{aligned}$$

By applying Lemma 1, we get

$$e_m \cdot [\mathbf{I}, n-L] \leq (m-1)\sum_{k=1}^{m} i_k + 2m(n-L) = 2mn - (m+1)L.$$

In view of (3) and (5) this yields

$$N_n^{(m)}(L) \leq C_1(q,m) q^{2mn-(m+1)L} |P(m;L)|,$$

and by applying the trivial bound $|P(m;L)| \leq (L+1)^m$ we obtain the result of the lemma. □

Since Theorem 1 is known for $m = 1$, we can assume in the proof of Theorem 1 that $m \geq 2$. We note first that with the notation in Section 2 we have

$$\mathrm{Prob}(L_n^{(m)}(\mathbf{S}) = L) = q^{-mn} N_n^{(m)}(L). \tag{6}$$

Now we fix a positive integer $n$ and let

$$A_n = \{\mathbf{S} \in (\mathbb{F}_q^m)^\infty : L_n^{(m)}(\mathbf{S}) \leq \frac{1}{m+1}(mn - 2\log_q n)\}.$$

Put $a(n) = \lfloor \frac{1}{m+1}(mn - 2\log_q n) \rfloor$. Then it follows from (6) and Lemma 2 that

$$\begin{aligned} \mu_{q,m}^\infty(A_n) &= q^{-mn} \sum_{L=0}^{a(n)} N_n^{(m)}(L) \leq q^{-mn} \sum_{L=0}^{a(n)} q^{(m+1)L} \\ &< \frac{q^{m+1}}{q^{m+1}-1} q^{(m+1)a(n)-mn} \leq \frac{q^{m+1}}{(q^{m+1}-1)n^2}. \end{aligned}$$

It follows that $\sum_{n=1}^\infty \mu_{q,m}^\infty(A_n) < \infty$. The Borel-Cantelli lemma [11–p. 228] now shows that the set of all $\mathbf{S} \in (\mathbb{F}_q^m)^\infty$ for which $\mathbf{S} \in A_n$ for infinitely many $n$ has $\mu_{q,m}^\infty$-measure 0. In other words, with probability 1 we have $\mathbf{S} \in A_n$ for at most finitely many $n$. From the definition of $A_n$ it follows then that with probability 1 we have

$$L_n^{(m)}(\mathbf{S}) > \frac{1}{m+1}(mn - 2\log_q n) \quad \text{for all sufficiently large } n. \tag{7}$$

For a fixed positive integer $n$, let now

$$B_n = \{\mathbf{S} \in (\mathbb{F}_q^m)^\infty : L_n^{(m)}(\mathbf{S}) \geq \frac{1}{m+1}(mn + (m+2)\log_q n)\}.$$

Put $b(n) = \lceil \frac{1}{m+1}(mn + (m+2)\log_q n) \rceil$ and assume first that $b(n) \leq n$. Then it follows from (6) and Lemma 3 that

$$\begin{aligned} \mu_{q,m}^\infty(B_n) &= q^{-mn} \sum_{L=b(n)}^{n} N_n^{(m)}(L) \leq C(q,m) q^{-mn} \sum_{L=b(n)}^{n} L^m q^{2mn-(m+1)L} \\ &\leq C(q,m) q^{-mn} n^m \sum_{L=b(n)}^{n} q^{(m+1)(n-L)+(m-1)n} \\ &= C(q,m) q^{-n} n^m \sum_{L=0}^{n-b(n)} q^{(m+1)L} \\ &\leq C_2(q,m) q^{-n} n^m q^{(m+1)(n-b(n))} \leq \frac{C_2(q,m)}{n^2} \end{aligned}$$

with a constant $C_2(q,m)$ depending only on $q$ and $m$. If $b(n) > n$, then $\mu_{q,m}^\infty(B_n) = 0$, and so the above bound holds in all cases.

It follows that $\sum_{n=1}^{\infty} \mu_{q,m}^{\infty}(B_n) < \infty$, and by applying the Borel-Cantelli lemma as before we deduce that with probability 1 we have

$$L_n^{(m)}(\mathbf{S}) < \frac{1}{m+1}(mn + (m+2)\log_q n) \quad \text{for all sufficiently large } n. \tag{8}$$

By combining (7) and (8), we complete the proof of Theorem 1.

For the proof of Theorem 2, we recall from (1) that

$$E_n^{(m)} = q^{-mn} \sum_{\mathbf{T} \in (\mathbb{F}_q^m)^n} L_n^{(m)}(\mathbf{T}).$$

This can be written as the integral

$$E_n^{(m)} = \int_{(\mathbb{F}_q^m)^\infty} L_n^{(m)}(\mathbf{S})\, d\mu_{q,m}^{\infty}(\mathbf{S}).$$

By the dominated convergence theorem [11–p. 125] and Theorem 1, we obtain

$$\begin{aligned}
\lim_{n\to\infty} \frac{E_n^{(m)}}{n} &= \lim_{n\to\infty} \int_{(\mathbb{F}_q^m)^\infty} \frac{L_n^{(m)}(\mathbf{S})}{n}\, d\mu_{q,m}^{\infty}(\mathbf{S}) \\
&= \int_{(\mathbb{F}_q^m)^\infty} \lim_{n\to\infty} \frac{L_n^{(m)}(\mathbf{S})}{n}\, d\mu_{q,m}^{\infty}(\mathbf{S}) = \int_{(\mathbb{F}_q^m)^\infty} \frac{m}{m+1}\, d\mu_{q,m}^{\infty}(\mathbf{S}) \\
&= \frac{m}{m+1},
\end{aligned}$$

which is the result of Theorem 2.

## 4 Conclusions

Theorem 1 shows that the $n$th joint linear complexity of a random $m$-fold multisequence over $\mathbb{F}_q$ is roughly equal to $mn/(m+1)$ for all $n \geq 1$. This establishes a benchmark for testing $m$-fold multisequences over $\mathbb{F}_q$ in terms of the joint linear complexity profile. In other words, the joint linear complexity of an $m$-fold multisequence $\mathbf{S}$ over $\mathbb{F}_q$ that is suitable for word-based stream ciphers should follow the same asymptotic behavior, i.e., we should have $L_n^{(m)}(\mathbf{S})$ close to $mn/(m+1)$ for all $n \geq 1$. As a simple consequence of Theorem 1, we have shown in Theorem 2 an asymptotic formula for the expected value $E_n^{(m)}$ of the $n$th joint linear complexity of random $m$-fold multisequences over $\mathbb{F}_q$.

## References

1. T.W. Cusick, C. Ding, and A. Renvall, *Stream Ciphers and Number Theory*, Elsevier, Amsterdam, 1998.
2. J. Daemen and C. Clapp, Fast hashing and stream encryption with PANAMA, *Fast Software Encryption* (S. Vaudenay, ed.), Lecture Notes in Computer Science, Vol. **1372**, pp. 60–74, Springer, Berlin, 1998.
3. Z.D. Dai, K. Imamura, and J.H. Yang, Asymptotic behavior of normalized linear complexity of multi-sequences, *Sequences and Their Applications – SETA 2004* (T. Helleseth, D. Sarwate, and H.-Y. Song, eds.), Lecture Notes in Computer Science, Vol. **3486**, pp. 129–142, Springer, Berlin, 2005.
4. E. Dawson and L. Simpson, Analysis and design issues for synchronous stream ciphers, *Coding Theory and Cryptology* (H. Niederreiter, ed.), pp. 49–90, World Scientific, Singapore, 2002.
5. C. Ding, G. Xiao, and W. Shan, *The Stability Theory of Stream Ciphers*, Lecture Notes in Computer Science, Vol. **561**, Springer, Berlin, 1991.
6. ECRYPT stream cipher project; available at `http://www.ecrypt.eu.org/stream`.
7. X.T. Feng and Z.D. Dai, Expected value of the linear complexity of two-dimensional binary sequences, *Sequences and Their Applications – SETA 2004* (T. Helleseth, D. Sarwate, and H.-Y. Song, eds.), Lecture Notes in Computer Science, Vol. **3486**, pp. 113–128, Springer, Berlin, 2005.
8. X.T. Feng, Q.L. Wang, and Z.D. Dai, Multi-sequences with $d$-perfect property, *J. Complexity* **21**, 230–242 (2005).
9. F.-W. Fu, H. Niederreiter, and M. Su, The expectation and variance of the joint linear complexity of random periodic multisequences, *J. Complexity*, to appear.
10. P. Hawkes and G.G. Rose, Exploiting multiples of the connection polynomial in word-oriented stream ciphers, *Advances in Cryptology – ASIACRYPT 2000* (T. Okamoto, ed.), Lecture Notes in Computer Science, Vol. **1976**, pp. 303–316, Springer, Berlin, 2000.
11. M. Loève, *Probability Theory*, 3rd ed., Van Nostrand, New York, 1963.
12. W. Meidl, Discrete Fourier transform, joint linear complexity and generalized joint linear complexity of multisequences, *Sequences and Their Applications – SETA 2004* (T. Helleseth, D. Sarwate, and H.-Y. Song, eds.), Lecture Notes in Computer Science, Vol. **3486**, pp. 101–112, Springer, Berlin, 2005.
13. W. Meidl and H. Niederreiter, The expected value of the joint linear complexity of periodic multisequences, *J. Complexity* **19**, 61–72 (2003).
14. W. Meidl and A. Winterhof, On the joint linear complexity profile of explicit inversive multisequences, *J. Complexity* **21**, 324–336 (2005).
15. H. Niederreiter, The probabilistic theory of linear complexity, *Advances in Cryptology – EUROCRYPT '88* (C.G. Günther, ed.), Lecture Notes in Computer Science, Vol. **330**, pp. 191–209, Springer, Berlin, 1988.
16. H. Niederreiter, Linear complexity and related complexity measures for sequences, *Progress in Cryptology – INDOCRYPT 2003* (T. Johansson and S. Maitra, eds.), Lecture Notes in Computer Science, Vol. **2904**, pp. 1–17, Springer, Berlin, 2003.
17. H. Niederreiter, H.X. Wang, and C.P. Xing, Function fields over finite fields and their applications to cryptography, *Topics in Geometry, Coding Theory and Cryptography* (A. Garcia and H. Stichtenoth, eds.), Springer, Berlin, to appear.
18. R.A. Rueppel, *Analysis and Design of Stream Ciphers*, Springer, Berlin, 1986.
19. R.A. Rueppel, Stream ciphers, *Contemporary Cryptology: The Science of Information Integrity* (G.J. Simmons, ed.), pp. 65–134, IEEE Press, New York, 1992.

20. L.-P. Wang and H. Niederreiter, Enumeration results on the joint linear complexity of multisequences, *Finite Fields Appl.*, to appear; available online as document doi:10.1016/j.ffa.2005.03.005.
21. L.-P. Wang, Y.-F. Zhu, and D.-Y. Pei, On the lattice basis reduction multisequence synthesis algorithm, *IEEE Trans. Inform. Theory* **50**, 2905–2910 (2004).
22. C.P. Xing, Multi-sequences with almost perfect linear complexity profile and function fields over finite fields, *J. Complexity* **16**, 661–675 (2000).
23. C.P. Xing, K.Y. Lam, and Z.H. Wei, A class of explicit perfect multi-sequences, *Advances in Cryptology – ASIACRYPT '99* (K.Y. Lam, E. Okamoto, and C.P. Xing, eds.), Lecture Notes in Computer Science, Vol. **1716**, pp. 299–305, Springer, Berlin, 1999.

# Period of Streamcipher *Edon80*

Jin Hong

National Security Research Institute,
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, Korea
`jinhong@etri.re.kr`

**Abstract.** The period of a recent streamcipher proposal *Edon80* is analyzed. Even though the average period may be quite large, we show that for a randomly chosen key and IV pair, there exists a non-dismissible probability that the produced keystream will be of very short period.

**Keywords:** *Edon80*, streamcipher, period.

## 1 Introduction

*Edon80* [3] is one of the streamciphers submitted to eSTREAM, the ECRYPT streamcipher project [1]. It was one of the ciphers chosen for presentation at the Symmetric Key Encryption Workshop (SKEW, Århus, Denmark, May, 2005) and rests on ideas previously presented at FSE 2005 [2]. The core of the cipher consists of what is called a *quasigroup string e-transformation,* and using related theory previously developed, the designers present some provable results supporting its security.

In this short paper, we study *Edon80*, focusing on its period. The designers had projected a period of $2^{103}$, and even though this may be true on average, we show that there exists a non-negligible probability that the keystream will fall into a much shorter period. For example, with random use of key and IV, keystreams of period as short as $2^{55}$ may occur with probability $2^{-71}$ and the existence of at least one key-IV pair producing a period-$2^{11}$ keystream can be expected.

## 2 *Edon80*

The streamcipher *Edon80* [3] will be described briefly in this section. It is a hardware oriented streamcipher with intended security level corresponding to 80 bits. Keys of 80-bit size and IVs of 64-bit size are used.

*Quasigroup.* The first ingredient of *Edon80* is four quasigroups of order 4. The quasigroup operators are given explicitly in Table 1. If you are not familiar with quasigroups, you can simply think of these as four different collections of (possibly non-commutative and non-associative) multiplication rules $\bullet_i$, defined on sets of four elements. Notice that each of the four operators are indexed by a 2-bit number.

© Springer-Verlag Berlin Heidelberg 2005

**Table 1.** Quasigroups

| $\bullet_0$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 2 | 1 | 3 |
| 1 | 2 | 1 | 3 | 0 |
| 2 | 1 | 3 | 0 | 2 |
| 3 | 3 | 0 | 2 | 1 |

| $\bullet_1$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | 3 | 0 | 2 |
| 1 | 0 | 1 | 2 | 3 |
| 2 | 2 | 0 | 3 | 1 |
| 3 | 3 | 2 | 1 | 0 |

| $\bullet_2$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 2 | 1 | 0 | 3 |
| 1 | 1 | 2 | 3 | 0 |
| 2 | 3 | 0 | 2 | 1 |
| 3 | 0 | 3 | 1 | 2 |

| $\bullet_3$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 3 | 2 | 1 | 0 |
| 1 | 1 | 0 | 3 | 2 |
| 2 | 0 | 3 | 2 | 1 |
| 3 | 2 | 1 | 0 | 3 |

*KeySetup* The 80-bit key $K$ is used to select 80 sequential quasigroups that are to be used for keystream production. The key is first divided into 40-many 2-bit subkeys.

$$K = K_0||K_1||\cdots||K_{39}.$$

Then each of the working quasigroup operators $*_i$ $(i = 0, 1, \ldots, 79)$ are assigned to be one of $\bullet_j$ $(j = 0, \ldots, 3)$. Explicitly, recalling that each quasigroup operator is indexed by a 2-bit number, we set

$$*_i \leftarrow \begin{cases} \bullet_{K_i} & 0 \le i < 40, \\ \bullet_{K_{i-40}} & 40 \le i < 80. \end{cases}$$

Hence the key $K$ determines the working quasigroups completely, and any consecutive 40 operators $*_i$ determine the key completely.

*IVSetup.* For any fixed key, *IVSetup* sends a 64-bit IV to a finite sequence $(a_0, \ldots, a_{79})$, where each $a_i$ is a 2-bit value, in a key-dependent manner. For discussions of this paper, we will not need to know its actual inner workings, but we shall assume that it is well-designed, meaning that some sort of randomness can be expected of the process.

*Keystream generation.* Consider the array of quasigroup elements given in Table 2. It consists of 81 rows, each row is a sequence of quasigroup elements extending infinitely to the right, and the top row is a fixed repeating pattern of period-4. The first column contains the 80 quasigroup operations $*_i$ previously determined from key. The next column contains the finite sequence obtained

**Table 2.** Keystream generation

| $*_i$ | | 0 | 1 | 2 | 3 | 0 | 1 | 2 | $\cdots$ |
|---|---|---|---|---|---|---|---|---|---|
| $*_0$ | $a_0$ | $a_{0,0}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ | $a_{0,4}$ | $a_{0,5}$ | $a_{0,6}$ | $\cdots$ |
| $*_1$ | $a_1$ | $a_{1,0}$ | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ | $a_{1,4}$ | $a_{1,5}$ | $a_{1,6}$ | $\cdots$ |
| $*_2$ | $a_2$ | $a_{2,0}$ | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ | $a_{2,4}$ | $a_{2,5}$ | $a_{2,6}$ | $\cdots$ |
| $*_3$ | $a_3$ | $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ | $a_{3,4}$ | $a_{3,5}$ | $a_{3,6}$ | $\cdots$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | | | | | |
| $*_{78}$ | $a_{78}$ | $a_{78,0}$ | $a_{78,1}$ | $a_{78,2}$ | $a_{78,3}$ | $a_{78,4}$ | $a_{78,5}$ | $a_{78,6}$ | $\cdots$ |
| $*_{79}$ | $a_{79}$ | $a_{79,0}$ | $a_{79,1}$ | $a_{79,2}$ | $a_{79,3}$ | $a_{79,4}$ | $a_{79,5}$ | $a_{79,6}$ | $\cdots$ |

from the *IVSetup* process. The rest of the elements $a_{i,j}$ are obtained sequentially starting from the top left corner through quasigroup operations. More explicitly, we set

$$a_{i,j} = *_i(a_{i,j-1}, a_{i-1,j}),$$

where we take $a_{-1,j} = j \pmod 4$ to be the top row and where $a_{i,-1} = a_i$ is the second column. Pictorially, we can view this as

$$\begin{tikzcd}
 & & a_{i-1,j} \arrow[dl] \arrow[dd, dashed, no head] \\
 & *_i \arrow[dr] & \\
a_{i,j-1} \arrow[ur] \arrow[uurr, dashed, no head] \arrow[rr, dashed, no head] & & a_{i,j}
\end{tikzcd} .$$

Finally, the keystream itself is given as every other element of the bottom row.

$$\text{keystream} = (a_{79,1}, a_{79,3}, a_{79,5}, \dots).$$

Our discussion will center mostly on the key determining the quasigroup operators $*_i$ $(i = 0, \dots 79)$ and the initial state $(a_0, \dots, a_{79})$ obtained right after the *IVSetup* operation. These two will be referred to together as *key-state* pair from now on.

*Period.* Calculation of each new row in Table 2 is said to be a quasigroup string e-transformation. The designers of *Edon80* assert that each e-transformation increases the string period by a factor of 2.48 on average. Following along arguments of the designers, the final keystream should have period

$$4 \times (2.48)^{80} \times \frac{1}{2} \approx 2^{105.8}$$

on average. The term 4 comes from the period of the initiating sequence at the top, and the term $\frac{1}{2}$ is multiplied because only every other term of last row is used as keystream. But there seems to have been a slight miscalculation by the designers and they project a period of $2^{103}$ instead of $2^{106}$.

No explicit restriction on the length of keystream usage is given by the designers. Hence readers are led to believe that keystreams of length up to $2^{103}$ bits may be used.

## 3 Undesirable Key-State Pairs

We shall instantiate Table 2 in such a way that the bottom row is a sequence of period 4. The corresponding key-state pair will result in producing a keystream of period 2.

## 3.1 Partial Key-State Pairs

Consider the series of five quasigroup string e-transformations given in Table 3.Notice that the period of each row is 4. Actually, we found $166 \approx 2^{7.38}$ such 5-row key-state pairs of period 4. Through an exhaustive searching program, we counted all $d$-row key-state pairs of period $p$, for small values of $d$ and $p$. The results are gathered in Table 4.The actual numbers written down in the table are logarithms of the counts. So, for example, the first entry states that there are approximately $2^{7.38}$ key-state pairs of period 4, consisting of 5 rows.

Going down any column, one can see that the numbers increase at almost a constant rate. This is easier to see in Figure 1, which is the graph version of Table 4. Extrapolating, we obtain the values given in Table 5 for the number of 40-row key-state pairs. We can expect these values to be at least approximately true and our future discussion will not depend too much on their exact value.

**Table 3.** Partial key-state pair of period 4

| $*_i$ | | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bullet_2$ | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | ... |
| $\bullet_0$ | 2 | 3 | 2 | 0 | 2 | 3 | 2 | 0 | 2 | 3 | ... |
| $\bullet_3$ | 1 | 2 | 2 | 0 | 1 | 2 | 2 | 0 | 1 | 2 | ... |
| $\bullet_0$ | 1 | 3 | 2 | 1 | 1 | 3 | 2 | 1 | 1 | 3 | ... |
| $\bullet_2$ | 0 | 3 | 1 | 2 | 0 | 3 | 1 | 2 | 0 | 3 | ... |

**Table 4.** $d$-row period-$p$ key-state pair count

| $d$ | $p = 4$ | $p = 8$ | $p = 16$ |
|---|---|---|---|
| 5 | 7.38 | 11.49 | 13.30 |
| 6 | 9.36 | 13.58 | 15.68 |
| 7 | 11.04 | 15.63 | 18.01 |
| 8 | 12.97 | 17.71 | 20.30 |
| 9 | 14.75 | 19.76 | 22.55 |
| 10 | 16.63 | 21.81 | 24.77 |
| 11 | 18.44 | 23.85 | 26.96 |
| 12 | 20.30 | 25.88 | 29.13 |
| 13 | 22.13 | 27.91 | 31.29 |
| 14 | 23.97 | 29.94 | 33.44 |
| 15 | 25.81 | 31.96 | 35.57 |
| 16 | 27.65 | 33.98 | |
| 17 | 29.49 | | |
| 18 | 31.33 | | |

**Table 5.** Expected number of 40-row short period key-states

| | $p = 4$ | $p = 8$ | $p = 16$ |
|---|---|---|---|
| $d = 40$ | 71.81 | 82.46 | 88.57 |

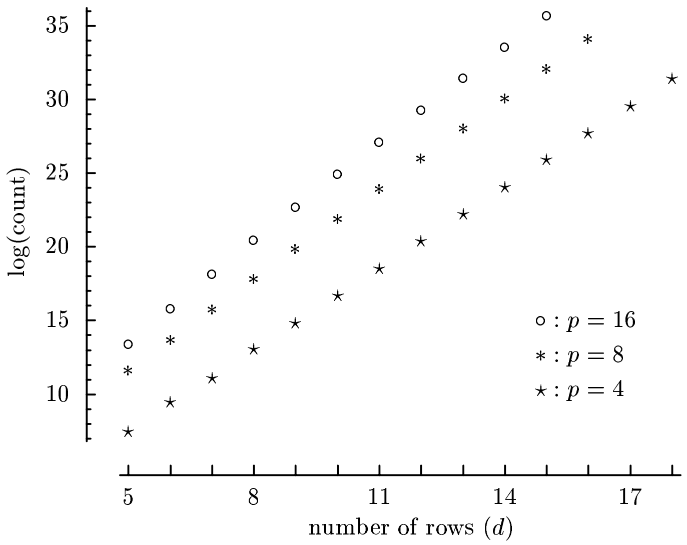


**Fig. 1.** Key-state pair count

## 3.2 Full Key-State Pairs Producing a Period-2 Keystream

Take any one of the $2^{72}$-many 40-row key-state pairs of period 4. Its bottom row, in particular, is a sequence of period 4. So there is a $(1/4)^4$ probability of it being equal to the top initiating sequence $(0, 1, 2, 3, 0, \dots)$. This has been experimentally verified to hold roughly true at smaller row counts.[1] Hence we can expect there to be approximately $2^{64}$-many 40-row key-state pairs having bottom row identical to the initiating sequence. In the appendix, we have written out one such partial key-state pair as an explicit example.

Now, fix any such 40-row key-state and attach another copy of the same partial key-state to its bottom. This gives an explicit instantiation for Table 2. This attaching is made possible by the fact that the 40-th row is identical to the top of the copy. We also remark that, in this argument, we did not overlook the fact that the top 40 rows will determine the key completely and hence also the quasigroup operators for the lower 40 rows. Since $*_i = *_{i+40}$ $(i = 0, \dots, 39)$, our choice of using a copy on the bottom 40 rows does not conflict with this structure of the cipher.

Recall that the actual keystream is every other quasigroup element from the bottom row sequence. Hence, we have shown the existence of at least $2^{64}$-many (full) key-state pairs that all produce the identical keystream $(1, 3, 1, 3, \dots)$ of period 2. We make no claims as to whether these key-state pairs may be reached through normal *IVSetup* process.

[1] We have reasons to believe that the slightly bigger value 1/240 reflects the actual situation better than $(1/4)^4$, but this does not affect the big picture.

## 4 Undesirable Key-IV Pairs

In this section we shall instantiate Table 2 in such a way that the bottom row is a sequence of relatively short period. There will be so many of these that a meaningful number of them will be reachable through normal state initialization process. In a way, we can see this as giving a class of *weak* key-IV pairs.

The instantiation will be done in two stages. First, the top 40 rows are filled so that the 40-th row is of period 4, 8, or 16. Then, the rest of rows are filled randomly subject to the restraints caused by the top 40 rows.

### 4.1 Key-State Pairs Producing Relatively Short Period Keystreams

Fix any 40-row key-state pair of period 4. In particular, the bottom row is a sequence of period 4, which may or may not be equal to the top initiating sequence. These 40-rows determine the key completely, and hence also the quasigroup operators $*_i$ $(i = 40, \ldots, 79)$ for the remaining lower 40 rows. Let us fix these lower row operators accordingly and fill in the remaining 40 initial states $(a_{40}, \ldots, a_{79})$ with arbitrary quasigroup elements.

Following along the arguments of the cipher designers, we can expect a period of $4 \times (2.48)^{40} \approx 2^{54.41}$ at the bottom 80-th row. This leads to a keystream of period $2^{53.41}$ which is much smaller than the value $2^{103}$ projected by the designers of *Edon80* and also smaller than even the intended security level $2^{80}$.

Since the 40-row key-state pairs were approximately $2^{72}$ in number (Table 5), and since we have 80-bit freedom coming from the choice of quasigroup elements filling the bottom 40 rows, we can expect the existence of at least

- $2^{72+80}$ key-state pairs producing keystreams of period $2^{53}$.

A more exact statement would be that there exists a group of $2^{72+80}$ key-state pairs whose average period is $2^{53}$. But we shall be a bit sloppy and express this as in the above.

If we start with 40-row key-state pairs of period 8, or 16, we obtain the existence of at least

- $2^{82+80}$ key-state pairs producing period-$2^{54}$ keystreams and
- $2^{89+80}$ key-state pairs producing period-$2^{55}$ keystreams,

respectively.

### 4.2 Key-IV Pairs Producing Relatively Short Period Keystreams

It remains to see if any of the discussed key-state pairs producing keystreams of short period are reachable by normal *IVSetup* operation.

Recall that the *IVSetup* process is a 64-bit to 160-bit mapping for any fixed key. Hence, under the assumption that the *IVSetup* is well-designed, given any key-state pair, under random use of key and IV, the probability of it being reachable by *IVSetup* is $2^{-96}$.

Thus we have the existence of at least

- $2^{56}$ key-IV pairs producing period-$2^{53}$ keystreams,
- $2^{66}$ key-IV pairs producing period-$2^{54}$ keystreams, and
- $2^{73}$ key-IV pairs producing period-$2^{55}$ keystreams.

Since there are $2^{80+64}$ key-IV pairs, for a randomly chosen key-IV pair, the probability of it producing a keystream

- of period $2^{53}$ is at least $2^{-88}$,
- of period $2^{54}$ is at least $2^{-78}$, and
- of period $2^{55}$ is at least $2^{-71}$.

The latter two probabilities are larger than $2^{-80}$ and hence constitutes a valid, although certificatory, attack on *Edon80*, if keystreams of such length were allowed. Of course, if users were just given the value $2^{103}$ projected as period by the cipher designers, as it is for the moment, keystreams of such length would certainly be allowed.

## 5 Distribution of Keystream Periods

In this section, we show that if we look for keystreams of period slightly longer than was considered in the previous section, then they are easier to encounter during random key and IV use. The existence of key-IV pairs producing extremely short period keystreams is also shown.

### 5.1 Probability / Period Tradeoffs

We do not have to divide the 80 rows appearing in Table 2 into just top 40 and bottom 40 rows. Extrapolating Table 4, one can come up with Table 6 that gives expected number of, say, 34-row key-state pairs of short period.

**Table 6.** Expected number of 34-row short period key-states

| | $p = 4$ | $p = 8$ | $p = 16$ |
|---|---|---|---|
| $d = 34$ | 60.77 | 70.34 | 75.85 |

One could start with any of these 34-row key-state pairs, fill the six rows from the 35-th to 40-th with random key-state values, fill lower 40 row quasigroup operators as defined by the upper 40 rows, and finally fill the rest of the states with random values. This would gives us $4 \cdot 6 + 80 = 104$ bits of freedom.

Since $4 \times (2.48)^{46} \times \frac{1}{2} \approx 2^{61.28}$, and since we loose 96-bit freedom from relating key-state pairs to key-IV pairs, we have the existence of following key-IV pairs.

- $2^{69}$ key-IV pairs producing period-$2^{61}$ keystreams.
- $2^{78}$ key-IV pairs producing period-$2^{62}$ keystreams.
- $2^{84}$ key-IV pairs producing period-$2^{63}$ keystreams.

All of these periods are still small relative to both $2^{103}$ and $2^{80}$. In terms of how often we may encounter these during random use of key and IV, we can expect higher than

- $2^{-75}$ probability of encountering period-$2^{61}$ keystreams,
- $2^{-66}$ probability of encountering period-$2^{62}$ keystreams, and
- $2^{-60}$ probability of encountering period-$2^{63}$ keystreams.

These probabilities are much larger than the intended security level $2^{-80}$.

This shows that there is a tradeoff between how short a period keystream we seek and how often we can encounter it at random. The tradeoff curve is given by Figure 2 for the segment that is most interesting, i.e., when probability is greater than $2^{-80}$ and period is smaller than $2^{80}$. As an example, the left-most "∘" of

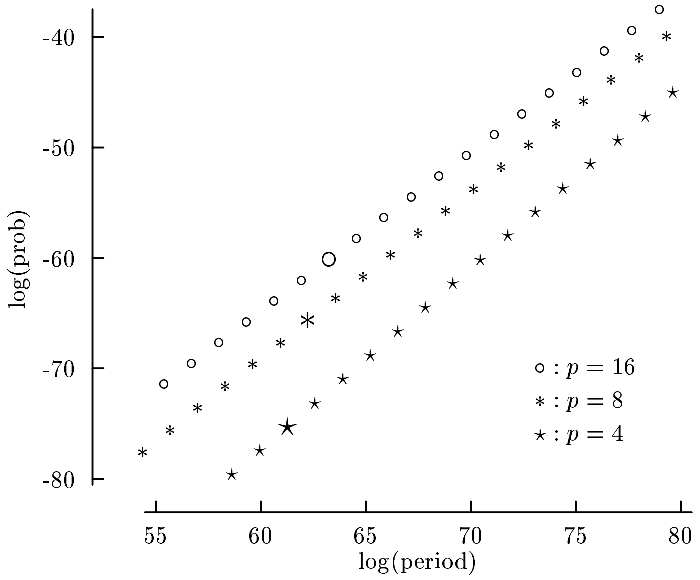


**Fig. 2.** Probability / period tradeoff

the graph tells us that if we use $p = 16$, we can show that with random use of key-IV pairs, one will encounter keystreams of period $2^{55.4}$ with probability no less than $2^{-71.4}$. We have also marked the three points corresponding to $d = 34$ case, used in the above argument, with larger fonts, so that you can verify your understanding of the graph.

Notice that we are only providing a lower bound on the probability for a keystream of certain period to occur. We make no claims as to if these bounds are even close to what actually happens. For example, with $p$ fixed, if the fact that given any divisor $n$ of $m$, a period-$n$ keystream is always a period-$m$ sequence, is considered, one can immediately conclude that the above probability figures are much lower than what actually happens. On the orthogonal side, keystreams of same period may be obtained from multiple $p$ values, so these can also be added and still be used as a lower bound. Furthermore, it is clear that with more computational power, we could work with $p = 32$ or $p = 64$ to obtain even better tradeoff curves.

We chose not to deal with these matters, as our rough lower bounds were already big enough to show that *Edon80* is under trouble with respect to period properties. The methods provided by this paper allows us to see the big picture, but we believe a totally different approach, for example, statistical modeling, is needed to understand the true extent of the period related problem, so as to be used on the constructive side.

## 5.2 Existence of Key-IV Pairs of Very Short Period

We could also divide the 80 rows of Table 2 into two parts below the 40-th row.

Let us go back to Sectio n 3.2 and first fill the top 40 rows with any one of the $2^{64}$-many 40-row period-4 key-state pairs that has the $(0, 1, 2, 3, 0, \dots)$ initiating sequence at the bottom 40-th row. As before, add another copy below, but fill the quasigroup elements $a_{64}, \dots, a_{79}$ for the last 16 rows at random, so that we have an extra 32-bit degree of freedom.

Since $4 \times (2.48)^{16} \times \frac{1}{2} \approx 2^{19.97}$, we have the existence of $2^{64+32}$ key-IV pairs producing period-$2^{20}$ keystreams. Recalling that probability of one of these being reachable by normal *IVSetup* process is $2^{-96}$, one can expect the existence of at least one key-IV pair that leads to a keystream of extremely short period $2^{20}$.

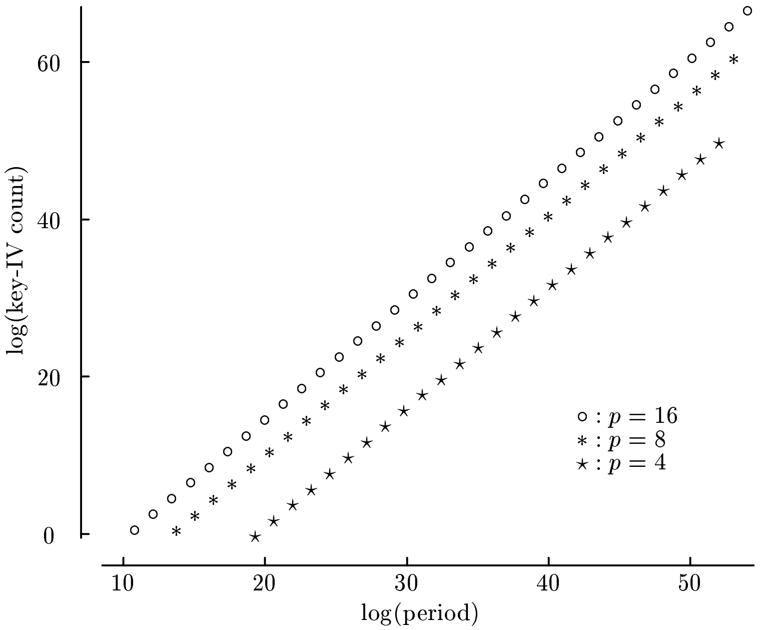


**Fig. 3.** Key-IV count / period tradeoff

If we start with $p = 16$ key-state pairs, we can conclude that there exists at least one key-IV pair producing an even shorter period-$2^{11}$ keystream. This is all shown in Figure 3. Although this single key-IV pair would be hard to reach at random through normal use of this cipher, it still does pose a threat, as the corresponding keystream is of extremely short period.

Once again, the counts we provide are only lower bounds. There may be ways to produce low-period sequences different from the explicit method we have considered.

Before ending this section, we remark that taking note of the top two "$\circ$" from Figure 3 can be interesting. For example, the top point tells us that there are $2^{67}$ key-IV pairs producing period-$2^{54}$ keystreams. The probability of encountering one of these key-IV pairs at random is $2^{-77}$, which is greater than the intended security level $2^{-80}$. It is clear that working with larger $p$ values will give additional meaningful tradeoff points.

## 6 Conclusion

The period property of streamcipher *Edon80* has been studied. We have shown that there are quite a large number of key-state pairs that produce identical sequence of period 2. We have also shown that there is a probability of no less than $2^{-71}$ for a random key-IV pair to produce a keystream of period $2^{55}$. The tradeoff between period and occurrence probability was studied and a period-$2^{63}$ keystream can be expected from a random key-IV pair with probability at least $2^{-60}$. Finally, we can expect the existence of at least one key-IV pair producing the extremely short period-$2^{11}$ keystream.

These short period keystreams occur with probability greater than $2^{-80}$ and the periods are very small relative to $2^{103}$, which is the value designers had projected as cipher period. These numbers are smaller than even $2^{80}$, which many would take for granted from an 80-bit security cipher, unless explicitly stated otherwise. Also, these key-IV pairs of bad characteristics, or *weak* key-IV pairs, are hard to categorize at the moment and hence avoiding them does not seem to be easy.

These results show that while the average period of *Edon80* may still be $2^{103}$ as projected, the range of keystream period is very wide with a non-dismissible portion of key-IV pairs produce keystreams of periods shorter than one would be comfortable with. Furthermore, one should keep in mind that we have only given a (rough) lower bound on the probability of short period keystream occurrences. Recent supplementary results [4] on the period of *Edon80*, written by the designers in response to an earlier version of the current paper, seem to indicate that the actual situation is even worse than what we have pointed out so far.

Even though our results do not give any information on how to recover keys or states, it does show that the period of *Edon80* is far from being well understood. Before *Edon80* can be used in practice, the distribution of keystream periods with respect to randomly chosen key-IV pairs should be fully understood and measures should be taken to prevent use of the shorter ones, if at all possible.

As the designers of *Edon80* put no explicit restriction on the length of keystream usable, and since probabilities for encountering these short keystreams are greater than what is expected from the intended security level, our observation is technically a valid attack on streamcipher *Edon80*.

## References

1. ECRYPT, eSTREAM - ECRYPT Stream Cipher Project. Information available from `http://www.ecrypt.eu.org/stream/`
2. S. Markovski, D. Gligoroski, and L. Kocarev, Unbiased random sequences from quasigroup string transformations. *FSE 2005*, LNCS 3557, pp. 163–180, Springer, 2005.
3. D. Gligoroski, S. Markovski, L. Kocarev, and M. Gušev, *Edon80* - Hardware synchoronous stream cipher. eSTREAM, ECRYPT Stream Cipher Project Report 2005/007, 2005. Presented at Symmetric Key Encryption Workshop, Århus, Denmark, May, 2005. Available from [1].
4. D. Gligoroski, S. Markovski, L. Kocarev, and M. Gušev, Understanding periods in Edon80. eSTREAM, ECRYPT Stream Cipher Project Report 2005/054, 2005. Available from [1].

## A 40-Row Key-State Pair

Here is an explicit key-state pair consisting of 40 rows that contains the initial sequence $(0, 1, 2, 3, 0, \ldots)$ at the bottom row. This is not a concatenation of smaller such partial key-state pairs.

| | $*_i$ | | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|
| 0: | $\bullet_1$ | 2 | 2 | 0 | 0 | 2 |
| 1: | $\bullet_1$ | 0 | 0 | 1 | 0 | 0 |
| 2: | $\bullet_2$ | 2 | 3 | 3 | 0 | 2 |
| 3: | $\bullet_0$ | 0 | 3 | 1 | 2 | 0 |
| 4: | $\bullet_0$ | 3 | 1 | 1 | 3 | 3 |
| 5: | $\bullet_0$ | 0 | 2 | 3 | 1 | 0 |
| 6: | $\bullet_0$ | 3 | 2 | 2 | 3 | 3 |
| 7: | $\bullet_0$ | 0 | 1 | 3 | 1 | 0 |
| 8: | $\bullet_0$ | 1 | 1 | 0 | 2 | 1 |
| 9: | $\bullet_0$ | 0 | 2 | 1 | 3 | 0 |
| 10: | $\bullet_0$ | 0 | 1 | 1 | 0 | 0 |
| 11: | $\bullet_0$ | 1 | 1 | 1 | 2 | 1 |
| 12: | $\bullet_1$ | 3 | 2 | 0 | 0 | 3 |
| 13: | $\bullet_0$ | 0 | 1 | 2 | 1 | 0 |
| 14: | $\bullet_1$ | 0 | 3 | 1 | 1 | 0 |
| 15: | $\bullet_1$ | 1 | 3 | 2 | 0 | 1 |
| 16: | $\bullet_0$ | 2 | 2 | 0 | 0 | 2 |
| 17: | $\bullet_1$ | 0 | 0 | 1 | 0 | 0 |
| 18: | $\bullet_2$ | 2 | 3 | 3 | 0 | 2 |
| 19: | $\bullet_0$ | 0 | 3 | 1 | 2 | 0 |
| 20: | $\bullet_0$ | 3 | 1 | 1 | 3 | 3 |
| 21: | $\bullet_0$ | 0 | 2 | 3 | 1 | 0 |
| 22: | $\bullet_0$ | 3 | 2 | 2 | 3 | 3 |
| 23: | $\bullet_0$ | 0 | 1 | 3 | 1 | 0 |
| 24: | $\bullet_0$ | 2 | 3 | 1 | 1 | 2 |
| 25: | $\bullet_1$ | 2 | 1 | 1 | 1 | 2 |
| 26: | $\bullet_1$ | 0 | 3 | 2 | 0 | 0 |
| 27: | $\bullet_1$ | 2 | 1 | 2 | 2 | 2 |
| 28: | $\bullet_1$ | 1 | 1 | 2 | 3 | 1 |
| 29: | $\bullet_0$ | 0 | 2 | 0 | 3 | 0 |
| 30: | $\bullet_3$ | 1 | 3 | 2 | 1 | 1 |
| 31: | $\bullet_1$ | 1 | 3 | 1 | 1 | 1 |
| 32: | $\bullet_0$ | 2 | 2 | 3 | 0 | 2 |
| 33: | $\bullet_0$ | 2 | 0 | 3 | 3 | 2 |
| 34: | $\bullet_1$ | 3 | 3 | 0 | 2 | 3 |
| 35: | $\bullet_1$ | 2 | 1 | 0 | 0 | 2 |
| 36: | $\bullet_3$ | 0 | 2 | 0 | 3 | 0 |
| 37: | $\bullet_3$ | 1 | 3 | 2 | 1 | 1 |
| 38: | $\bullet_3$ | 1 | 2 | 2 | 3 | 1 |
| 39: | $\bullet_3$ | 3 | 0 | 1 | 2 | 3 |

# On the Algebraic Immunity of Symmetric Boolean Functions

An Braeken and Bart Preneel

Katholieke Universiteit Leuven,
Dept. Elect. Eng.-ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
{an.braeken, bart.preneel}@esat.kuleuven.be

**Abstract.** In this paper, we analyze the algebraic immunity of symmetric Boolean functions. The algebraic immunity is a property which measures the resistance against the algebraic attacks on symmetric ciphers. We identify a set of lowest degree annihilators for symmetric functions and propose an efficient algorithm for computing the algebraic immunity of a symmetric function. The existence of several symmetric functions with maximum algebraic immunity is proven. In this way, we have found a new class of functions which have good implementation properties and maximum algebraic immunity.

## 1 Introduction

Symmetric functions have the property that the function value is determined by the Hamming weight of the input vector. Therefore, a symmetric function in $n$ variables can be defined by a vector of length $n+1$ which represents the function values of the different Hamming weights of the input vectors. For this reason, symmetric functions are very interesting functions in order to obtain low memory in software. In hardware implementation, only a low number of gates is required [15]. Properties such as balancedness and resiliency, propagation characteristics and nonlinearity of symmetric functions are studied by Canteaut and Videau [3]. It is shown that these functions do not behave very well in general with respect to a combination of the properties such as nonlinearity, degree, and resiliency, which are important properties for resisting distinguishing and correlation attacks [2].

In 2002, several successful algebraic attacks on stream ciphers were proposed by Courtois [5]. The success of these attacks do not depend on the classical properties of nonlinearity or resiliency, but mainly on the weak behavior with respect to the property of algebraic immunity. In this paper we study the resistance of the symmetric functions against the algebraic attacks. We identify a set of polynomials whose linear combinations lead to lowest degree annihilators of a symmetric function. Since the size of this set is very small in comparison with the general case, the algorithm for computing the algebraic immunity (AI)

© Springer-Verlag Berlin Heidelberg 2005

of a symmetric function becomes much more efficient. We prove the existence of several symmetric functions with optimal algebraic immunity.

First, Sect. 2 deals with some background on Boolean functions and more in particular on symmetric Boolean functions. Based on the identification of a set of lowest degree annihilators of a symmetric function, we propose an algorithm for computing the algebraic immunity of symmetric functions in Sect. 3. Sect. 4 presents several classes of symmetric functions which possess maximum algebraic immunity. Finally, we conclude in Sect. 5.

## 2 Background

Let us first recall the basic background on Boolean functions together with some properties of symmetric Boolean functions which were proven by Canteaut and Videau [13].

Let $\mathbb{F}_2^n$ be the set of all $n$-tuples of elements in the field $\mathbb{F}_2$ (Galois field with two elements), endowed with the natural vector space structure over $\mathbb{F}_2$. An element $\overline{u} = (u_0, \ldots, u_{n-1})$ in $\mathbb{F}_2^n$ can be represented by an integer $\mathbb{Z}_{2^n}$ belonging to the interval $[0, 2^n - 1]$, *i.e.*, $u = \sum_{i=0}^{n-1} u_i 2^i$. We will use both notations interchangeable in the rest of the paper.

A Boolean function $f$ on $\mathbb{F}_2^n$ is a mapping from $\mathbb{F}_2^n$ onto $\mathbb{F}_2$. It can be uniquely represented by the truth table (TT) which is the vector of length $2^n$ consisting of its function values. The support of the function $f$, $\sup(f)$ contains all vectors $\overline{x}$ for which $f(\overline{x}) = 1$. The (Hamming) weight $\mathrm{wt}(\overline{v})$ of a vector $\overline{v} \in \mathbb{F}_2^n$ is defined as the number of nonzero positions.

Another unique representation, called the ANF, is the polynomial

$$f(\overline{x})= \bigoplus_{(a_0,\ldots,a_{n-1})\in\mathbb{F}_2^n} h(a_0,\ldots,a_{n-1})x_0^{a_0}\ldots x_{n-1}^{a_{n-1}}, h(\overline{a}) = \sum_{\overline{x}\preceq\overline{a}} f(\overline{x}), \text{for any } \overline{a} \in \mathbb{F}_2^n\,,$$

where $\overline{x} \preceq \overline{a}$ means that $x_i \leq a_i$ for all $0 \leq i \leq n-1$. The degree of the polynomial determines the algebraic degree of this function. The ANF of a function consists of the modulo 2 sum of polynomials $(x_0 \oplus a_0 \oplus 1) \cdots (x_{n-1} \oplus a_{n-1} \oplus 1)$ for all $\overline{a} \in \mathbb{F}_2^n$ such that $f(\overline{a}) = 1$. Denote the all-zero function or vector by $\overline{0}$ and the all-one function or vector by $\overline{1}$.

The Walsh transform $W_f$ of a function $f$ on $\mathbb{F}_2^n$ is defined as the real valued transformation

$$W_f(\overline{w}) = \sum_{\overline{x}\in\mathbb{F}_2^n} (-1)^{f(\overline{x})+\overline{w}\cdot\overline{x}}\,.$$

From the Walsh transform, we derive the property of nonlinearity $N_f = 2^{n-1} - \frac{1}{2}\max_{\overline{w}\in\mathbb{F}_2^n} |W_f(\overline{w})|$, which represents the smallest distance between a Boolean function and any affine function [11].

As response to the algebraic attacks, Meier et al. [10] introduced the concept of algebraic immunity (AI) for a Boolean function $f$ on $\mathbb{F}_2^n$. This measure defines the lowest degree of a non-zero function $g$ from $\mathbb{F}_2^n$ into $\mathbb{F}_2$ for which $f \cdot g = \overline{0}$

or $(f \oplus \overline{1}) \cdot g = \overline{0}$. The function $g$ for which $f \cdot g = \overline{0}$ is called an *annihilator function* of $f$. The set of all annihilators of $f$ is denoted by $An(f)$. The AI is upper bounded by $\lceil \frac{n}{2} \rceil$ as proven in [4].

Symmetric functions have the property that the function value of all vectors with the same weight is equal. Consequently, the truth table of the symmetric function on $\mathbb{F}_2^n$ can be replaced by a vector $v_f$ of length $n+1$ where the components $v_f(i)$ for $0 \leq i \leq n$ represent the function value for vectors of weight $i$. The vector $v_f$ is called the value vector (VV) of the symmetric function $f$.

The ANF representation for a symmetric function can also be replaced by a shorter form [3–Prop. 2], called the simplified ANF (SANF). Denote the homogeneous symmetric function, which is the function that contains all terms of degree $i$ for $0 \leq i \leq n$, by $\sigma_i$. Then, the SANF is a polynomial in $\mathbb{F}_2[x_0, \ldots, x_{n-1}]/(x_0^2 - x_0, \ldots, x_{n-1}^2 - x_{n-1})$ with basis elements the homogeneous symmetric functions $\sigma_i$ for $0 \leq i \leq n$:

$$f(\overline{x}) = \bigoplus_{i=0}^{n} \lambda_f(i)\sigma_i, \quad \lambda_f(i) = \sum_{k \preceq i} v_f(k), \text{for } 0 \leq i \leq n\,.$$

The vector $\lambda_f = (\lambda_f(0), \ldots, \lambda_f(n))$ is called the simplified ANF vector (SANF vector).

## 3 Annihilators of Symmetric Functions

We first distinguish a set of polynomials whose linear combinations lead to lowest degree annihilators of a symmetric function. Based on this set, we propose an efficient algorithm for computing the AI of a symmetric Boolean function.

Denote the homogeneous symmetric function of degree $i$ which depends on the $j$ variables $\{x_{n-j}, x_{n-j+1}, \ldots, x_{n-1}\}$ with $j \geq i$ by $\sigma_i^j$. We also use the notation of $P_l^k$ to represent the set of polynomials where each polynomial contains all $k$ variables $\{x_0, \ldots, x_{k-1}\}$ and consists of the product of at most $l$ factors where every factor is either the sum of two variables, one variable, or the complement of one variable. Consequently $\lceil \frac{k}{2} \rceil \leq l$. Note that the variables in the polynomials $P_l^k$ play the same role, which means that changing the indices of the variables does not introduce new polynomials in $P_l^k$. Therefore, we define the role of the variables $\{x_0, \ldots, x_{k-1}\}$ in the polynomials of $P_l^k$ as follows. Depending on $l$, the first factors involving the first variables (starting from $x_0, x_1, \ldots$) may consist of one variable, the complement of one variable or the sum of two variables. The following factors may consist of one variable and the sum of two variables, while the last factors consist of the sum of two variables.

*Example 1.* If $\lceil \frac{k}{2} \rceil = l$, only the polynomial $(x_0 \oplus x_1)(x_2 \oplus x_3) \cdots (x_{k-2} \oplus x_{k-1})$ for $k$ even and the polynomials $x_0(x_1 \oplus x_2)(x_3 \oplus x_4) \cdots (x_{k-2} \oplus x_{k-1})$ and $(x_0 \oplus 1)(x_1 \oplus x_2)(x_3 \oplus x_4) \cdots (x_{k-2} \oplus x_{k-1})$ for $k$ odd belong to $P^k_{\lceil \frac{k}{2} \rceil}$. If $\lceil \frac{k}{2} \rceil = l - 1$, the polynomials $x_0x_1(x_2 \oplus x_3)\cdots(x_{k-2} \oplus x_{k-1})$, $(x_0 \oplus 1)x_1(x_2 \oplus x_3)\cdots(x_{k-2} \oplus x_{k-1})$, $(x_0 \oplus 1)(x_1 \oplus 1)(x_2 \oplus x_3)\cdots(x_{k-2} \oplus x_{k-1})$, $(x_0 \oplus x_1)(x_2 \oplus x_3)\cdots(x_{k-2} \oplus x_{k-1})$, belong to $P^k_{\lceil \frac{k}{2} \rceil + 1}$ for $k$ even.

The goal of this section is to show that at least one of the lowest degree annihilators with degree strictly less than $\lceil \frac{n}{2} \rceil$ of a symmetric function on $\mathbb{F}_2^n$ is a linear combination of the polynomials of the following form:

$$\begin{aligned}
n \text{ even:}\quad & \sigma_0^2 P^{n-2}_{\frac{n}{2}-1}, \sigma_0^3 P^{n-3}_{\frac{n}{2}-1}, \ldots, \sigma_0^{n-1} P^1_{\frac{n}{2}-1}, \sigma_0,\\
& \sigma_1^4 P^{n-4}_{\frac{n}{2}-2}, \ldots, \sigma_1^{n-1} P^1_{\frac{n}{2}-2}, \sigma_1, \ldots, \sigma^{n-2}_{\frac{n}{2}-2} P^2_1, \sigma^{n-1}_{\frac{n}{2}-2} P^1_1, \sigma_{\frac{n}{2}-2}, \sigma_{\frac{n}{2}-1}\\
n \text{ odd:}\quad & \sigma_0^1 P^{n-1}_{\lceil \frac{n}{2} \rceil -1}, \sigma_0^2 P^{n-2}_{\lceil \frac{n}{2} \rceil -1}, \ldots, \sigma_0^{n-1} P^1_{\lceil \frac{n}{2} \rceil -1}, \sigma_0,\\
& \sigma_1^3 P^{n-3}_{\lceil \frac{n}{2} \rceil -2}, \ldots, \sigma_1^{n-1} P^1_{\lceil \frac{n}{2} \rceil -2}, \sigma_1, \ldots, \sigma^{n-2}_{\lceil \frac{n}{2} \rceil -2} P^2_1, \sigma_{\lceil \frac{n}{2} \rceil -2}, \sigma_{\lceil \frac{n}{2} \rceil -1}.
\end{aligned}$$

As $\lceil \frac{k}{2} \rceil \leq l$, the functions $\sigma_k$ for $k \in \{0, \ldots, \lceil \frac{n}{2} \rceil - 1\}$ depend on $2k+2, 2k+3, \ldots, n$ variables for $n$ even and $2k+1, 2k+2, \ldots, n$ variables for $n$ odd in order to obtain annihilators of degree less than or equal to $\lceil \frac{n}{2} \rceil - 1$. We will call this set of polynomials $\mathrm{AN}_S$. We now give some examples of annihilators which consist of the linear combination of polynomials in $\mathrm{AN}_S$.

*Example 2.* Let $n = 16$, and suppose $f$ is a symmetric Boolean function on $\mathbb{F}_2^n$ with value vector $v_f$ that satisfies $v_f(i) = 0$ for $i \in \{6, 7, 10, 11\}$. Then the function $g(\overline{x}) = \sigma_2^9 x_0(x_1 \oplus x_2)(x_3 \oplus x_4)(x_5 \oplus x_6)$ represents an annihilator of the function $f$. This follows from the fact that $\sigma_2^9$ is equal to 1 only for vectors in $\mathbb{F}_2^9$ with weight equal to 2,3,6,7. The function $x_0(x_1 \oplus x_2)(x_3 \oplus x_4)(x_5 \oplus x_6)$ is equal to 1 only for a subset of vectors in $\mathbb{F}_2^7$ with weight 4. Consequently the function $g$ is equal to 1 only for a subset of vectors of weight 6,7,10,11.

If the value vector in the coordinates 2 and 6 is equal to $c$ where $c \in \{0, 1\}$ for a symmetric function $f$ in 10 variables, then $(x_0 \oplus 1)(\sigma_2^9 \oplus \sigma_3^9)$ represents an annihilator with degree 3 of $f$ if $c = 0$, or $f \oplus \overline{1}$ if $c = 1$.

**Theorem 1.** *One of the lowest degree annihilators of a symmetric function can be constructed by means of a linear combination of the polynomials in* $AN_S$.

*Proof.* Annihilators of symmetric functions are equal to 0 for all vectors of a certain weight which belong to the support of the corresponding symmetric function. But the annihilators can be 0 or 1 for vectors which do not belong to the support of the symmetric function. Therefore, an example of an annihilator is the one which consists of the product of a symmetric function which depends on the last $n-k$ variables in order to guarantee that the function value is 1 for vectors of the same weight, together with a polynomial that depends on the other $k$ variables and which is 1 for a subset of vectors with fixed weight. The polynomials $P^k_l$ in the polynomials of $\mathrm{AN}_S$ are constructed in such way that they are equal to 1 only for a subset of vectors which have exactly one fixed and equal weight. Corollary 1, which is based on Lemma 1, proves that the annihilators constructed by means of a linear combination of the polynomials in $\mathrm{AN}_S$ have lowest possible degree by showing that if one of the factors of the polynomials $P_l^k$ would consist of more than 3 variables (in order to decrease the degree), then there also exists an annihilator constructed by means of linear combinations of the polynomials of $\mathrm{AN}_S$ whose support is contained in the support of this annihilator and which has smaller or equal degree. Therefore, we first prove Lemma 1. $\square$

*Remark 1.* We note that the annihilators constructed by linear combinations of the polynomials in $\mathrm{AN}_S$ do not determine the complete basis of the ideal of annihilators with degree strictly less than $\lceil \frac{n}{2} \rceil$ of a symmetric function. For instance, the function $x_0\sigma_3$ on $\mathbb{F}_2^{10}$ is annihilator of all symmetric functions on $\mathbb{F}_2^{10}$ for which $v_f(4) = v_f(8) = 0$. But the function $x_0\sigma_3^9 \in \mathrm{AN}_S$ also satisfies this property. Both functions are linearly independent. Also note that the variables of the polynomials $P_l^k$ play the same role in the representation, and that they only depend on the first $k$ variables. This is possible due to the symmetry of the symmetric function. Since we are only interested in the existence of at least one annihilator in order to determine the AI of the function, we can restrict us for the search of annihilators into the set functions obtained by linear combinations of the polynomials in $\mathrm{AN}_S$.

**Lemma 1.** *Let $r \geq 3$ and $n \geq r-1$. Define $S_i^n$ as the symmetric function on $n$ variables of degree $i$,*

$$S_i^n = \bigoplus_{0 \leq k \leq i} c_k^S \sigma_k^n \text{ where } c_k^S \in \{0,1\} \text{ for all } 0 \leq k \leq i.$$

*Denote the set of weights in the support of $S_i^n$ by $V_S$. Define also $S_{i-(r-1)}^{n-(r-1)} = \bigoplus_{0 \leq k \leq i} c_k^S \sigma_{k-(r-1)}^{n-(r-1)}$ where $\sigma_i = 0$ for $i < 0$ and denote its support of the value vector by $V_{S'}$. Then*

$$\{a+r-1 : a \in V_{S'}\} \subseteq \{a, a+2, \ldots, a+r-1 : a \in V_S\} \quad (1)$$
$$\{a+r : a \in V_{S'}\} \subseteq \{a+1, a+3, \ldots, a+r : a \in V_S\} \quad (2)$$

We refer to an extended version of the paper for the proof of this lemma.

*Example 3.* Let $n = 10, r = 3$. The support of the value vector of the function $\sigma_0^{10} \oplus \sigma_1^{10} \oplus \sigma_2^{10} \oplus \sigma_5^{10}$ belongs to $V_S = \{0,3,4,5,8\}$. The support of the value vector of $\sigma_0^8 \oplus \sigma_3^8$ belongs to $V_{S'} = \{0,1,2,4,5,6,8\}$. The theorem implies that $\{2,3,4,6,7,8,10\} \subseteq \{0,2,3,4,5,6,7,8,10\}$.

Directly from Lemma 1, we can derive

**Corollary 1.** *Let $r$ be odd and $r \geq 3$, then the support of $S_i^{n-r}(x_0 \oplus \cdots \oplus x_{r-1})$ contains the support of $S_{i-(r-1)}^{n-(2r-1)} x_0(x_1 \oplus x_2)\cdots(x_{2r-3} \oplus x_{2r-2})$. The support of $S_i^{n-r}(x_0 \oplus \cdots \oplus x_{r-1} \oplus 1)$ contains the support of $S_{i-(r-1)}^{n-(2r-1)}(x_0 \oplus 1)(x_1 \oplus x_2)\cdots(x_{2r-3} \oplus x_{2r-2})$. Both pairs of functions have the same degree $i+1$.*

*Let $r$ be even and $r \geq 4$, then the support of $S_i^{n-r}(x_0 \oplus \cdots \oplus x_{r-1})$ contains the support of $S_{i-(r-2)}^{n-(2r-2)}(x_0 \oplus x_1)(x_2 \oplus x_3) \cdots (x_{2r-3} \oplus x_{2r-4})$. Both functions have the same degree $i+1$. The support of $S_i^{n-r}(x_0 \oplus \cdots \oplus x_{r-1} \oplus 1)$ contains the support of $S_{i-r}^{n-2r}(x_0 \oplus x_1)(x_2 \oplus x_3) \cdots (x_{2r-1} \oplus x_{2r-2})$. The last function has degree $i$ in comparison with degree $i+1$ of the first function. This equation also holds for $r = 2$.*

We conclude that if one or more factors of the polynomial $P_l^k$ would consist of the complement of two terms or more than three terms, then there always exists an annihilator constructed by means of a linear combination of polynomials in $\mathrm{AN}_S$ which has degree smaller or equal and whose support is contained in the support of that annihilator.

Let us now compute the number of polynomials in the set $\mathrm{AN}_S$.

**Theorem 2.** *The number $N$ of polynomials in $AN_S$ is equal to*

$$N = 3 \cdot 2^{\lceil \frac{n}{2} \rceil} - 2 \cdot \left\lceil \frac{n}{2} \right\rceil - 3 \,.$$

*Proof.* We will compute the number for $n$ even. In a similar way, the result is obtained for $n$ odd. Denote $R_k^n$ for $n$ even and $0 \leq k \leq \frac{n}{2} - 1$ as the sum of all elements which have $\sigma_k^i$ for $i = 2k+2, \ldots, n$ as factor, *i.e.*, the sum of all elements of the sets $P^i_{\frac{n}{2}-k-1}$ for $i = 0, \ldots, n-(2k+2)$:

$$R_k^n = \sum_{i=0}^{n-(2k+2)} |P^i_{\frac{n}{2}-k-1}| \,.$$

For $i = n-(2k+2)$, there is exactly one element in $P^{n-(2k+2)}_{\frac{n}{2}-k-1}$, namely the polynomial $(x_1 \oplus x_2) \cdots (x_{n-2k-2} \oplus x_{n-2k-3})$. Every decrease of $i$ until $i = \frac{n}{2} - k - 1$ with 1 gives one more degree of freedom, which leads to a factor of two more for the possible polynomials in $P^i_{\frac{n}{2}-k-1}$. For instance, suppose the polynomial $P^i_{\frac{n}{2}-k-1}$ has the form $(x_1 \oplus x_2)(x_3 \oplus x_4) \cdots$ at step $i$. After removing one variable at step $i-1$, we can have two additional elements in $P^{i-1}_{\frac{n}{2}-k-1}$ namely $x_1(x_2 \oplus x_3) \cdots$ and $(x_1 \oplus 1)(x_2 \oplus x_3) \cdots$. Removing another variable leads again to two more polynomials: $(x_1 \oplus x_2) \cdots$, $x_1 x_2 \cdots$, $(x_1 \oplus 1)x_2 \cdots$, $(x_1 \oplus 1)(x_2 \oplus 1) \cdots$. For $i < \frac{n}{2} - k - 1$, due to the smaller number of variables, the total number of polynomials decreases again with a factor of 2. Therefore, we have that for $0 \leq k \leq \frac{n}{2} - 1$:

$$R_k^n = 2 \sum_{i=0}^{\frac{n}{2}-k-2} 2^i + 2^{\frac{n}{2}-k-1} \,.$$

Consequently, the total number of terms belonging to class 2 is equal to

$$N = \sum_{k=0}^{\frac{n}{2}-1} R_k^n = 2 \sum_{i=1}^{\lceil \frac{n}{2} \rceil -1} (2^i - 1) + 2^{\lceil \frac{n}{2} \rceil} - 1 \,. \qquad \square$$

*Example 4.* For $n = 14$, we have that

$$\begin{aligned}
\sigma_0 &\longrightarrow (|P_6^{12}|, \dots, |P_6^0|) = (1,2,4,8,16,32,64,32,16,8,4,2,1)\\
\sigma_1 &\longrightarrow (|P_5^{10}|, \dots, |P_5^0|) = (1,2,4,8,16,32,16,8,4,2,1)\\
\sigma_2 &\longrightarrow (|P_4^{8}|, \dots, |P_4^0|) = (1,2,4,8,16,8,4,2,1)\\
\sigma_3 &\longrightarrow (|P_3^{6}|, \dots, |P_3^0|) = (1,2,4,8,4,2,1)\\
\sigma_4 &\longrightarrow (|P_2^{4}|, \dots, |P_2^0|) = (1,2,4,2,1)\\
\sigma_5 &\longrightarrow (|P_1^{2}|, \dots, |P_1^0|) = (1,2,1)\\
\sigma_6 &\longrightarrow |P_0^0| = 1
\end{aligned}$$

## 3.1 An Algorithm for Computing AI

As shown in the previous section, one of the lowest degree annihilators of degree less than $\lceil \frac{n}{2} \rceil$ consists of a linear combination of the $N$ polynomials in $\mathrm{AN}_S$. As determined in Theorem 2, the size of the set $\mathrm{AN}_S$ is much smaller than the number of all polynomials of degree less than $\lceil \frac{n}{2} \rceil$ which is equal to $\sum_{i=0}^{\lceil \frac{n}{2} \rceil -1} \binom{n}{i}$. Table 1 shows the comparison between both numbers for dimensions $n = 2k$ with $5 \leq k \leq 10$. We can conclude that the difference increases with the dimension.

The main goal of the algorithm that computes the AI of a function consists in finding suitable linear combinations within these terms. Consequently, roughly speaking the complexity for computing the AI of a symmetric function can be upper bounded by $N^{2.81} \approx 58 \cdot 2^{1.4n}$, where 2.81 corresponds to the exponent for Gaussian elimination [1].

Moreover, the additional tricks presented in [10] can be used to accelerate the algorithm even further. Due to the fact that we have much less functions to combine in the algorithm for computing the AI of a symmetric function, the AI of any arbitrary symmetric function can be computed for much larger dimensions.

Instead of checking the whole set of $2^{n+1}$ symmetric functions for functions on $\mathbb{F}_2^n$ with maximum AI, we first present some properties on the value vector of a symmetric function with maximum AI. These properties can be immediately derived from the existence of the annihilators constructed by means of linear combinations of polynomials in $\mathrm{AN}_S$.

**Table 1.** Comparison of the size of annihilator-set

| $n$ | 10 | 12 | 14 | 16 | 18 | 20 |
|---|---|---|---|---|---|---|
| $\sum_{i=0}^{\lceil \frac{n}{2} \rceil -1} \binom{n}{i}$ | 386 | 1 586 | 6 476 | 26 333 | 106 762 | 431 910 |
| $\lvert\mathrm{AN}_S\rvert$ | 83 | 177 | 367 | 749 | 1 524 | 3 049 |

## 3.2 Properties

**Theorem 3.** *Let $f$ be a symmetric Boolean function on $\mathbb{F}_2^n$ with value vector $v_f$. If $v_f(\lceil \frac{n}{2} \rceil - 1) = v_f(\lceil \frac{n}{2} \rceil + 1)$ for all $n$, or in addition for $n$ odd $v_f(\lceil \frac{n}{2} \rceil - 2) = v_f(\lceil \frac{n}{2} \rceil)$, then $f$ can not have maximum AI.*

**Theorem 4.** *Let $2^j \leq n < 2^{j+1} - 1$ where $j \geq 1$ and $f$ be a symmetric Boolean function on $\mathbb{F}_2^n$ with value vector $v_f$. Define for all $0 \leq i < 2^{j-1}$ the set $V_i = \{l : l \equiv i \mod 2^{j-1} \text{ for } 0 \leq l < n\}$. If there exists an $i \in \{0, \ldots, 2^{j-1} - 1\}$ such that $v_f(k) = 0$ (resp. 1) for all $k \in V_i$, then the AI of $f$ is less than or equal to $2^{j-1} - 1$. For $n = 2^{j+1} - 1$ where $j \geq 1$, the value vector of $f$ should be of the form $(\overline{a}|\overline{a}^c)$ where $\overline{a} \in \mathbb{F}_2^j$ in order to reach the maximum AI.*

Finally, we want to mention that also the condition on the weight of a Boolean function, as derived in [6], is very strong for symmetric functions with an odd number of variables. It implies that maximum AI can only be obtained for balanced functions if $n$ is odd. A large set of balanced functions in $n$ odd are the trivially balanced functions, *i.e.*, the functions with value vector $v_f(i) = v_f(n - i) \oplus 1$ for all $0 \leq i \leq \lfloor \frac{n}{2} \rfloor$. In fact, the trivially balanced functions form the whole set of balanced functions for $n$ odd and $n \leq 128$, except in dimensions $n \in \{13, 29, 31, 33, 35, 41, 47, 61, 63, 73, 97, 103\}$ as shown in [14].

## 3.3 Experiments

For the computation of the AI, we can use a more efficient algorithm than the algorithm of [10] as explained above and thus reach higher dimensions.

If $n$ is odd, the condition of trivially balancedness is very powerful. We checked until $n \leq 17$ and can conclude that the only trivially balanced functions with maximum AI have value vector $v_f$ such that

$$v_f(i) = \begin{cases} 0 \text{ for } i < \lceil \frac{n}{2} \rceil \\ 1 \text{ for } i \geq \lceil \frac{n}{2} \rceil . \end{cases} \tag{3}$$

In [12], the complete set of non-trivially balanced functions for $n = 13$ is described. From this description, we derive that the AI of the non-trivial balanced functions in 13 variables is less than or equal to 3 due to Theorem 4. Therefore, we conclude that all symmetric functions in $n$ odd and $n \leq 17$ with maximum AI have value vector defined by (3). We will show in the next section that a symmetric function with such value vector always has maximal AI for every $n$ odd. Moreover, it can be easily proven that for $n = 2^i - 1, 2^i + 1$, with $i \geq 2$, only the trivially balanced functions with value vector determined by (3) have maximum AI. In these dimensions, the property of Theorem 4 is very powerful.

For $n$ even, we found more symmetric functions with maximum AI. In the next section, we will theoretically prove the maximum AI for some of these functions. The theorems will cover all symmetric functions with maximum AI in dimensions less than or equal to 12 and all but one in dimensions 14 and 16. We refer to the extended version of the paper for the complete set of symmetric Boolean functions with maximum AI in dimensions $n = 6, 8, 10, 12, 14, 16$.

## 4 Symmetric Functions with Maximum AI

In this section, we show the existence of several symmetric functions with maximum AI for all dimensions $n$. Let us first recall that the property of AI is invariant under affine transformation in the input variables, *i.e.*, $f(\overline{x})$ and $f(\overline{x}A \oplus \overline{b})$, where $A$ is an $n \times n$ nonsingular matrix and $\overline{b} \in \mathbb{F}_2^n$ will have the same AI. This follows from the fact that if $g$ is annihilator of $f$, then $g(\overline{x}A \oplus \overline{b})$ is annihilator of $f(\overline{x}A \oplus \overline{b})$.

However, the AI of two functions $f(\overline{x})$ and $f(\overline{x}) \oplus \overline{c} \cdot \overline{x}$ with $\overline{c} \in \mathbb{F}_2^n$ can differ at most with 1. This can be easily seen as follows. Let $g$ be annihilator of $f$ such that $f(\overline{x}) \cdot g(\overline{x}) = 0$, then $g(\overline{x})(\overline{c} \cdot \overline{x} \oplus \overline{1})$ is annihilator of $(f(\overline{x}) \oplus \overline{c} \cdot \overline{x})$ because $(f(\overline{x}) \oplus \overline{c} \cdot \overline{x})g(\overline{x})(\overline{c} \cdot \overline{x} \oplus \overline{1}) = f(\overline{x})g(\overline{x})(\overline{c} \cdot \overline{x} \oplus 1) \oplus (\overline{c} \cdot \overline{x})g(\overline{x})(\overline{c} \cdot \overline{x} \oplus \overline{1}) = 0$. The last equality follows from the fact that $\overline{c} \cdot \overline{x} \oplus \overline{1}$ is annihilator of $\overline{c} \cdot \overline{x}$.

We now investigate the affine transformations on the input variables which will transform a symmetric function into a new symmetric function. Due to the following lemma proven by Dawson and Wu, we only need to check the transformations $\overline{x} \mapsto \overline{x}A \oplus c\overline{1}$, where $A$ is a nonsingular $n \times n$ binary matrix and $c \in \mathbb{F}_2$.

**Lemma 2.** *[8] Let $a \in \mathbb{F}_2^n \setminus \{\overline{0}, \overline{1}\}$. If $f$ is a symmetric Boolean function, then $f(\overline{x} \oplus \overline{a})$ is symmetric if and only if $f$ is affine.*

**Theorem 5.** *In $n$ even, the only binary linear transformation on the input variables of a symmetric function that will compute a new symmetric function on $\mathbb{F}_2^n$ is the transformation $T = \overline{x} \mapsto \overline{x}A$, where $A$ is a nonsingular $n \times n$ matrix over $\mathbb{F}_2$ with the property that the sum of the elements in each row and column of $A$ is equal to $n-1$. For $n$ odd, no such transformations exist.*

*The transformation $(x_0, \ldots, x_{n-1}) \mapsto (x_0 \oplus 1, \ldots, x_{n-1} \oplus 1)$ for all $n$ will map a symmetric function with value vector $v_f$ to a symmetric function with value vector equal to the reverse of this value vector, i.e., $v_f^r$.*

*Proof.* A minimal requirement for a binary linear transformation $x \mapsto \overline{x}A$ which maps a symmetric function onto a symmetric function is that the weight $W$ of the columns and rows of $A$ is equal, since all variables play the same role in a symmetric function. If $W$ is greater than 1 and smaller than $n-1$, the transformation is not bijective or does not lead to a symmetric function.

Consider $n$ even and $W = n-1$. If $\mathrm{wt}(\overline{x})$ is odd and equal to $i$, then we show that $\mathrm{wt}(\overline{x}A)$ is equal to $n-i$. Denote by $V = \{i : x_i \neq 0\}$. The coordinates $j$ with $j \in \{0, \ldots, n-1\}$ in the vector $\overline{x}A$ are 1 if and only if the elements on the corresponding column $j$ of $A$ are 1 exactly on the $i$ positions of the set $V$. (Note that it is not possible that there are $i - 2k$ with $k \geq 1$ elements in the columns of $A$ which are 1 and $2k$ elements which are 0 due to the fact that $W = n-1$.) The number of such columns in $A$ is equal to $\binom{n-i}{n-i-1} = n-i$ for $i$ odd and $1 \leq i \leq n-1$.

Now we show that if $\mathrm{wt}(\overline{x})$ is even and equal to $i$, then $\mathrm{wt}(\overline{x}A) = i$. Denote by $V = \{i : x_i \neq 0\}$. The coordinates $j$ with $j \in \{0, \ldots, n-1\}$ in the vector $\overline{x}A$

are 1 if and only if the elements on the corresponding column $j$ of $A$ are 1 on exactly $i-1$ positions of the set $V$. There are $\binom{i}{i-1} = i$ possibilities for this to occur.

For $n$ odd, the transformation $T$ is not bijective which follows immediately from the fact that vectors of weight 0 and $n$ are both mapped onto vectors of weight 0.

Finally, since the transformation $(x_0, \ldots, x_{n-1}) \mapsto (x_0 \oplus 1, \ldots, x_{n-1} \oplus 1)$ maps a vector of weight $i$ onto a vector of weight $n-i$, this transformation corresponds to the mapping of $v_f(i)$ onto $v_f(n-i)$ for every $i$ with $0 \leq i \leq n$. □

We now present three basic classes of symmetric functions with maximum AI. We refer to the extended version of the paper for the proofs of the theorems in this section.

**Class 1**

**Theorem 6.** *The symmetric function $f$ in $\mathbb{F}_2^n$ with value vector*

$$v_f(i) = \begin{cases} 0 \textit{ for } i < \lceil \frac{n}{2} \rceil \\ 1 \textit{ else} \end{cases} \tag{4}$$

*has maximum AI. Let us denote this function $f$ by $F_k$ where $k$ is equal to the threshold $\lceil \frac{n}{2} \rceil$.*

*Remark 2.* The maximum AI of this class of symmetric functions was independently proven in [7] using a different proof method. This result was also presented at [2].

For $n$ even, we prove that also the function which only differs from the threshold function $F_{\lceil \frac{n}{2} \rceil}$ in the function value of the vector $(1, \ldots, 1)$ has maximum AI. Denote the zero vector on $\mathbb{F}_2^{n+1}$ with 1 on position $i$ by $\overline{e}_i$ for $0 \leq i \leq n$.

**Theorem 7.** *The symmetric function $f$ with value vector $v_{F_{\lceil \frac{n}{2} \rceil}} \oplus \overline{e}_n$ in $\mathbb{F}_2^n$ for $n$ even has maximum AI. The degree of $f$ is equal to $n$ if $n \neq 2^i$ for $i \geq 1$ and equal to $2^{i-1}$ else.*

**Class 2**

For $n \geq 8$ and even, we can distinguish another class of symmetric functions with maximum AI. These symmetric functions differ from $F_{\frac{n}{2}}$ in two symmetric positions such that they possess the same weight as $F_{\frac{n}{2}}$. Denote by $\overline{s}_i$ the all zero vector on $\mathbb{F}_2^{n+1}$ with 1 on positions $i, n-i$ for $0 \leq i < \frac{n}{2}$.

**Theorem 8.** *Let $n = 2k$ and $k \geq 4$. The symmetric function $f$ with value vector $v_{F_{\frac{n}{2}}} \oplus \overline{s}_{k-4}$ on $\mathbb{F}_2^n$ has maximum AI.*

Again, the symmetric functions $f$ which differ from the functions presented in Theorem 8 only in the all-one vector have maximum AI for $n \geq 10$. This can be obtained by using the proof technique of Theorem 7 for showing the non-existence of annihilators with degree less than $\frac{n}{2}$ for $f$ and the proof technique of Theorem 8 for $f \oplus \overline{1}$.

**Theorem 9.** *Let $n = 2k$ and $k \geq 5$. The symmetric function $f$ with value vector $v_{F_{\frac{n}{2}}} \oplus \overline{s}_{k-4} \oplus \overline{e}_n$ on $\mathbb{F}_2^n$ has maximum AI. The degree of $f$ is equal to $n$ if $n \neq 2^i$ with $i \geq 1$ and equal to $2^{i-1}$ else.*

We also present another class of functions which differs from $F_{\frac{n}{2}}$ in two symmetric positions. These functions coincide with the function defined in Theorem 7 for $n = 8$.

**Theorem 10.** *Let $f$ be a symmetric function on $\mathbb{F}_2^n$ with $n$ even. If $\binom{n}{\frac{n}{2}} \equiv 1 \mod 4$, then the function with value vector $v_{F_{\frac{n}{2}}} \oplus \overline{s}_0$ has maximum AI.*

*Example 5.* The numbers $n = 2^i$ for $i \geq 3$ satisfy the property that $\binom{n}{\frac{n}{2}} \equiv 1 \mod 4$.

## Class 3

For $n$ even, the third class of functions with maximum AI differs from $F_{\frac{n}{2}}$ in only one position. Therefore these functions have weight different from the weight of the functions of class 1 or 2.

**Theorem 11.** *Let $f$ be a symmetric function on $\mathbb{F}_2^n$ with $n$ even. For $1 \leq i < \lfloor \frac{n}{4} \rfloor$, if $\binom{\frac{n}{2}+t-i}{t} \equiv 1 \mod 2$ for all $t \in \{1, \ldots, i\}$, then the function $f$ with value vector $v_{F_{\frac{n}{2}}} \oplus \overline{e}_{n-i}$ has maximum AI.*

*Example 6.* For $n = 14$, since $\binom{7}{1} \equiv 1 \mod 2$, the function value vector $v_{F_7} \oplus \overline{e}_{13}$ has maximum AI. Also $\binom{7}{3} \equiv 1 \mod 2$, $\binom{6}{2} \equiv 1 \mod 2$, $\binom{5}{1} \equiv 1 \mod 2$, and thus the function with value vector $v_{F_7} \oplus \overline{e}_{11}$ represents a function with maximum AI.

## Functions Derived From Classes 1, 2, and 3

For $n$ even, the symmetric functions from classes 1, 2, and 3 can be used to derive other symmetric functions by means of the affine transformation $(x_0, \ldots, x_{n-1}) \mapsto (x_0 \oplus x_1 \oplus \cdots \oplus x_{n-2}, x_1 \oplus x_2 \oplus \cdots \oplus x_{n-1}, \ldots, x_{n-1} \oplus x_0 \oplus \cdots \oplus x_{n-3})$. As already explained in the proof of Theorem 4, this transformation maps vectors of odd weight $i$ to vectors with weight $n - i$. If the weight is even, then nothing is changed.

**Corollary 2.** *Let $f$ be a symmetric functions on $\mathbb{F}_2^n$ which belongs to class 1 or 2. If $n = 4k$, then $f \oplus \sigma_1$ has maximum AI. If $n = 4k + 2$, then the symmetric function with value vector $v_{f \oplus \sigma_1} \oplus \overline{e}_{\frac{n}{2}}$ has maximum AI.*

*Let $f$ be a symmetric functions on $\mathbb{F}_2^n$ which belongs to class 3. If $n = 4k$, then the function with value vector $v_{f \oplus \sigma_1} \oplus c\overline{e}_{n-i}$, where $c = 1$ if $i$ is odd and $c = 0$ otherwise, has maximum AI. If $n = 4k + 2$, then the function with value vector $v_{f \oplus \sigma_1} \oplus \overline{e}_{\frac{n}{2}} \oplus c\overline{e}_{n-i}$, where $c = 1$ if $i$ is odd and $c = 0$ otherwise has maximum AI.*

*Remark 3.* We want to note that the symmetric Boolean functions $f$ derived from the function $F_{\lceil \frac{n}{2} \rceil}$ and also $F_{\lceil \frac{n}{2} \rceil} \oplus \sigma_n$ if $n$ is even have very simple annihilators. For instance, it can be easily seen that the functions $x_{i_1} \cdots x_{i_{\lceil \frac{n}{2} \rceil}}$ with $0 \le i_1 < i_2 < \cdots < i_{\lceil \frac{n}{2} \rceil} \le n-1$ are annihilators of $F_{\lceil \frac{n}{2} \rceil} \oplus \overline{1}$. Moreover, they form exactly the basis of the set of annihilators for $F_{\lceil \frac{n}{2} \rceil} \oplus \overline{1}$. The basis of the annihilators of $F_{\lceil \frac{n}{2} \rceil} \oplus \sigma_n \oplus \overline{1}$ consists of the elements $\{x_0 \cdots x_{\lceil \frac{n}{2} \rceil - 1} \oplus x_{i_1} \cdots x_{i_{\lceil \frac{n}{2} \rceil}} : 0 \le i_1 < i_2 < \cdots < i_{\lceil \frac{n}{2} \rceil} \le n-1, (i_1, \ldots, i_{\lceil \frac{n}{2} \rceil}) \ne (0, \ldots, \lceil \frac{n}{2} \rceil - 1)\}$.

A high number of terms in the equations is another important criteria for the algebraic attacks. Therefore, one should be very careful in choosing the taps of the filter function and the taps of the LFSR when using these symmetric functions in a filter generator. The annihilators of the affine equivalent functions are more complicated. However, this does not change the situation, since one can always replace the filter generator by an equivalent generator with different initial state and connection polynomial of the LFSR and with filter function equal to the affine equivalent one (see [9]).

Annihilators of degree $\frac{n}{2}$ of symmetric functions which belong to classes 2 or 3 are more complicated and consist of more terms.

## Properties

Properties such as degree, weight and maximum value in the Walsh spectrum of the functions from classes 1, 2, and 3 for $n$ even are summarized in Table 2. The property of degree can be easily derived by using Proposition 2 and Proposition 4 of [3]. The nonlinearity of the functions is immediately derived from the weight since one can show that $\max_{\overline{w} \in \mathbb{F}_2^n} |W_f(\overline{w})| = |W_f(\overline{0})|$. This is proven in detail by Dalai *et. al* in [7].

The functions from class 1 for $n$ odd are trivially balanced. The nonlinearity of these functions is equal to $2^{n-1} - \binom{n-1}{\frac{n-1}{2}}$. This follows from the fact that the restriction to the subspace $x_n = 0$ (resp. $x_n = 1$) is equal to the symmetric function (resp. complement of symmetric function) of class 1 in $\mathbb{F}_2^{n-1}$. As mentioned

**Table 2.** Properties of Symmetric function on $\mathbb{F}_2^n$ with Maximum AI for $n$ even

| Function | Degree | weight | $\max \lvert W_f \rvert$ |
|---|---|---|---|
| $F_{\frac{n}{2}}$ | $2^{\lfloor \log_2 n \rfloor}$ | $2^{n-1} + \frac{1}{2}\binom{n}{\frac{n}{2}}$ | $\binom{n}{\frac{n}{2}}$ |
| $F_{\frac{n}{2}} \oplus \overline{s}_{\frac{n}{2}-4}$ | $2^{\lfloor \log_2 n \rfloor}$ | $2^{n-1} + \frac{1}{2}\binom{n}{\frac{n}{2}}$ | $\binom{n}{\frac{n}{2}}$ |
| $F_{\frac{n}{2}} \oplus \overline{e}_{n-i}$ | $\ge n-i$ | $2^{n-1} + \frac{1}{2}\binom{n}{\frac{n}{2}} - \binom{n}{n-i}$ | $\binom{n}{\frac{n}{2}} - 2\binom{n}{n-i}$ |

in [3], trivially balanced functions satisfy the property that the derivative with respect to the all one vector is constant, *i.e.*, $D_{\overline{1}}f = \overline{1}$. Also $W_f(\overline{v}) = 0$ for all vectors $\overline{v}$ of even weight.

## 5 Conclusions

We have presented in this paper an efficient algorithm for computing the AI of a symmetric Boolean function. We have identified several classes of symmetric functions with maximum AI.

Since the nonlinearity of functions with maximum AI is not sufficiently high for resisting distinguishing attacks and correlation attacks as explained in [2], we also investigated the existence of symmetric functions with suboptimal AI and better nonlinearity. As shown in the extended version of the paper, it seems that it is not possible to obtain a sufficient order of AI (in the order of 7) together with a reasonable nonlinearity (in the order of $\epsilon = 2^{-9}$) for symmetric functions which depend on less than 32 variables. Therefore in order to use symmetric functions in practise, one should use them as a building block for instance by means of the direct sum with a highly nonlinear Boolean function. Examples of functions with high nonlinearity and which have still reasonable hardware complexity are the Boolean functions which are affine equivalent with the trace function of the power functions.

On the other hand, it is clear that a symmetric function has lots of structure. Therefore, it is an interesting research question whether this structure can be exploited in an attack. Also, the use of the direct sum of two functions has been pointed out as a possible weakness in the design. But again, no attack is known for this. There are two straightforward ways to destroy the symmetry and to still maintain a large set of the properties such as nonlinearity, AI and degree. The first way is by affine transformation on the input variables which keeps the AI, nonlinearity and degree invariant. However, this method is not a good solution, since one can construct an equivalent cipher, with different initial state and different connection polynomial for the LFSR(s) where the function is again symmetric (see [9]). The second way is to add an affine function, which keeps the nonlinearity and degree invariant, but will decrease the AI with 1 in general. For this transformation, it is not immediately clear how to rewrite it to an equivalent scheme where the symmetric function is again obtained.

## Acknowledgement

We thank Anne Canteaut and Marion Videau for their useful comments and in particular for providing an algorithm that computes the AI of symmetric functions. We would also like to thank the anonymous referees for their helpful comments. This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the European Commission through the IST Programme under Contract IST2002507932

ECRYPT. An Braeken is an F.W.O. Research Assistant, sponsored by the Fund for Scientific Research - Flanders (Belgium).

## References

1. D.H. Bailey, K. Lee, and H.D. Simon. Using Strassens algorithm to accelerate the solution of linear systems. Journal of Supercomputing, 4:357371, 1990.
2. A. Braeken and J. Lano. Design principles for LFSR-based stream ciphers. In B. Preneel and S. Tavares, editors, Selected Areas in Cryptography SAC 2005, Lecture Notes in Computer Science. Springer-Verlag, 2005.
3. A. Canteaut and M. Videau. Symmetric Boolean functions. IEEE Transactions on Information Theory, IT-51(8):2791-2811, 2005.
4. N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In E. Biham, editor, Eurocrypt 2003, volume 2656 of Lecture Notes in Computer Science, pages 345-359. Springer-Verlag, 2003.
5. N.T. Courtois. Higher order correlation attacks, XL algorithm, and cryptanalysis of Toyocrypt. In P.J. Lee and C.H. Lim, editors, Information Security and Cryptology ICISC, 2002, volume 2587 of Lecture Notes in Computer Science, pages 182-199. Springer-Verlag, 2002.
6. D.K. Dalai, K.C. Gupta, and S. Maitra. Results on algebraic immunity for cryptographically significant Boolean functions. In A. Canteaut and K. Viswanathan, editors, Indocrypt 2004, volume 3348 of Lecture Notes in Computer Science, pages 92-106. Springer-Verlag, 2004.
7. D.K. Dalai, S. Maitra, and S. Sarkar. Basic theory in construction of Boolean functions with maximum possible algebraic immunity. Cryptology ePrint Archive, Report 2005/229.
8. E. Dawson and C.-H.Wu. On the linear structures of symmetric Boolean functions. 16:87-102, 1996.
9. C. Ding, G. Xiao, and W. Shan. Stability Theory of Stream Ciphers. Springer-Verlag, 1991. ISBN 3-540-54973-0, 0-387-54973-0.
10. W. Meier, E. Pasalic, and C. Carlet. Algebraic attacks and decomposition of Boolean functions. In C. Cachin and J. Camenisch, editors, Eurocrypt 2004, volume 3027 of Lecture Notes in Computer Science, pages 474-491. Springer-Verlag, 2004.
11. W. Meier and O. Staffelbach. Nonlinearity criteria for cryptographic functions. In J.-J. Quisquater and J. Vandewalle, editors, Eurocrypt 1989, volume 434 of Lecture Notes in Computer Science, pages 549-562. Springer-Verlag, 1989.
12. P. Sarkar and S. Maitra. Balancedness and correlation immunity of symmetric Boolean functions. Electronic Notes in Discrete Mathematics, 15:178-183, 2002.
13. M. Videau. On some properties of symmetric Boolean functions. In D.J. Costello and J.B. Hajek, editors, IEEE International Symposium on Information Theory, 2004, Proceedings, page 500. IEEE Press, 2004.
14. J. von zur Gathen and J.R. Roche. Polynomials with two values. Combinatorica, 17(3):345-362, 1997.
15. I. Wegener. The Complexity of Boolean Functions. Wiley, 1987.

# On Highly Nonlinear S-Boxes and Their Inability to Thwart DPA Attacks

Claude Carlet

INRIA, Projet CODES, BP 105 - 78153, Le Chesnay Cedex, France
University of Paris 8 (MAATICAH)
claude.carlet@inria.fr

**Abstract.** Prouff has introduced recently, at FSE 2005, the notion of transparency order of S-boxes. This new characteristic is related to the ability of an S-box, used in a cryptosystem in which the round keys are introduced by addition, to thwart single-bit or multi-bit DPA attacks on the system. If this parameter has sufficiently small value, then the S-box is able to withstand DPA attacks without that ad-hoc modifications in the implementation be necessary (these modifications make the encryption about twice slower). We prove a lower bound on the transparency order of highly nonlinear S-boxes. We show that some highly nonlinear functions, and in particular the S-box of AES, have very bad transparency orders.

## 1 Introduction

Block cipher cryptosystems embedded in cryptographic devices are sensitive to a series of cryptanalyses such as differential and linear attacks. Much is known on the desired characteristics (balancedness, high nonlinearity or high algebraic degree) of their S-boxes which permit an optimal resistance to these attacks. But these cryptosystems and their implementation must also withstand the attacks on the hardware. Indeed, one can obtain information from the side channels in evaluating the timing of operations or their power consumption. The first side channel attack, introduced by Kocher [20], permitted to obtain the whole secret key in several cryptosystems (more precisely, in the implementation of these cryptosystems), thanks to a timing of operations. Since this seminal paper, a large number of very efficient attacks has been performed on various cryptographic implementations (see e.g. [7, 8, 14, 16, 23, 24]), in particular in implementations for smart cards. The *differential power analysis* (DPA) is one of the most powerful such methods. Its efficiency is much greater than that of linear or differential cryptanalyses. For instance, in the case of DES, a DPA attack needs about 2000 bytes of plaintext-ciphertext pairs, whereas linear or differential attacks need terabytes of such pairs (encrypted with a single key, or twice as many encrypted with several keys, this makes them completely unpractical in most situations, and in particular in the case of embedded cryptography, which is the most favourable situation for DPA attacks). Fortunately, countermeasures to DPA attacks exist, that can be added to the implementation to withstand

© Springer-Verlag Berlin Heidelberg 2005

these attacks; for example, adding computations which are not necessary for the encryption itself, or enciphering data so that the attacker has no information on the input to the S-box [9, 12, 14, 28]. But these countermeasures make the code size and the complexity of computation greater. This is a concern in the area of embedded cryptography, becaus e of limited power and memory capability, and it slows down the encryption by a factor of 2, roughly. A potentially better method would be to choose the S-boxes so that they permit a high resistance to linear and differential cryptanalyses and to DPA attacks as well. But is this possible? To study such possibility, E. Prouff, extending in [27] the study made by Guilley et al. [15] for the so-called single-bit DPA, has introduced a new characteristic for S-boxes used in block cryptosystems in which the round keys are introduced by addition: the transparency order. This extension by Prouff to several coordinate functions of the S-box instead of just one (or of a linear combination of the coordinate functions) shows that the transparency order must not be greater that some value, depending on the amount of noise inside the device and on the number of encryptions that a cryptanalyst can obtain with the same key. The introduction of this parameter is interesting, as a first attempt at theoretically characterizing and quantifying the resistance of S-boxes to DPA attacks. Obviously, it would be nice if we could exhibit S-boxes with reasonably high nonlinearities and with low transparency orders; unfortunately, this is still an open problem. Prouff shows that the transparency order of an S-box $F$ is null if the S-box is a (cryptographically useless) affine function of a certain type and that it is the worst possible when the coordinate functions of $F$ are all bent. He also proves that the transparency order of a function satisfying the propagation criterion of high degree has bad value. However, this gives no information on the behavior of realistic S-boxes. We show in the present paper that the most important of those S-boxes currently used in cryptosystems - namely the inverse function, used as S-box in the AES - has a very bad transparency order. This may not mean that the use of inverse S-box is going to diminish because of this. But it proves what was only believed true without proof before: the countermeasures cannot be avoided with this precise S-box. We are able to obtain this result thanks to bounds on the transparency order which relate it to the Walsh spectra of the functions. We calculate the exact transparency order of the Gold functions (which are not used as S-boxes because of their algebraic degree, which is too low since it equals 2) and this permits us to evaluate (at least for these functions) how precise are our bounds.

The paper is organized as follows: in Section 2, we recall some preliminaries on S-boxes (Walsh transform, APN and AB functions). In Section 3, we recall the definition of the transparency order and we prove several lower bounds. Relation (4) shows in particular that the transparency order can be lower bounded by an expression only depending on the Walsh transforms of the coordinate functions of the S-box (recall that the Walsh transform plays also a central role in the evaluation of the nonlinearity of the S-box). In Section 4, we deduce a lower bound on the transparency order of the inverse function (in a finite field of characteristic 2), and in particular of the S-box of the AES. We deduce that it cannot contribute by itself to a resistance to DPA attacks. We also calculate the transparency order of the Gold functions and compare it with the lower bounds obtained in Section 3.

## 2 Preliminaries on S-Boxes

Let $\mathbb{F}_2^n$ be the $n$-dimensional vector space over the field $\mathbb{F}_2$. We call *$n$-variable Boolean function* any function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$ and *$(n,m)$-function* any function $F = (f_1, \dots, f_m)$ from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ (the coordinate functions $f_i$ of $F$ are $n$-variable Boolean functions). $(n,m)$-functions are used as S-boxes (substitution boxes) in block ciphers, often with $n = m$. An $(n,m)$-function is called *balanced* if its output is uniformly distributed over $\mathbb{F}_2^m$, which permits to withstand statistical attacks.

For every $n$-variable Boolean function $f$, the character sum

$$\mathcal{F}(f) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)}$$

and the related *Walsh transform* $W_f(a) = \mathcal{F}(f + l_a)$, where $l_a$ is the linear function $l_a(x) = a \cdot x = a_1x_1 + \ldots + a_nx_n$ (this addition being obviously calculated mod 2), play an important cryptographic role. In particular, the *nonlinearity* $N_f$ of $f$ equals $2^{n-1} - \frac{1}{2}\max_{a \in \mathbb{F}_2^n} |W_f(a)|$. The number $\max_{a \in \mathbb{F}_2^n} |W_f(a)|$ is usually called the *linearity* of $f$ and we shall denote it by $L_f$. It is lower bounded by $2^{n/2}$, because of Parseval's relation $\sum_{a \in \mathbb{F}_2^n} {W_f}^2(a) = 2^{2n}$.

An $n$-variable Boolean function is called *bent* if its nonlinearity equals $2^{n-1} - 2^{n/2-1}$.

Another important parameter related to a Boolean function $f$ is the autocorrelation function $AC_f(a) = \mathcal{F}(D_af)$, where $D_af$ is the derivative of $f$ in the direction of $a$:

$$D_af(x) = f(x) + f(x + a).$$

The Fourier transform of the autocorrelation function, that is, by definition, the function $\widehat{AC_f}(b) = \sum_{a \in \mathbb{F}_2^n} AC_f(a)(-1)^{a \cdot b}$, equals the square of the Walsh transform of $f$:

$$\widehat{AC_f}(b) = {W_f}^2(b).$$

In particular, for $b = 0$, we have:

$$\sum_{a \in \mathbb{F}_2^n} \mathcal{F}(D_af) = {W_f}^2(0).$$

The following relation will be useful in the sequel: let $f$ and $g$ be two $n$-variable Boolean functions, then

$$\sum_{a \in \mathbb{F}_2^n} \mathcal{F}(D_af)\mathcal{F}(D_ag) = 2^{-n} \sum_{a \in \mathbb{F}_2^n} {W_f}^2(a){W_g}^2(a). \tag{1}$$

Indeed, $\sum_{a\in\mathbb{F}_2^n}\mathcal{F}(D_af)\mathcal{F}(D_ag)$ is the value at 0 of the Fourier transform of the function $a\rightarrow\mathcal{F}(D_af)\mathcal{F}(D_ag)$ and it is well-known that the Fourier transform of the Hadamard product of two functions equals $2^{-n}$ times the convolutional product of the Fourier transforms of the functions. Hence, since the Fourier transform of $AC_f$ equals ${W_f}^2$, we have (1).

Any $(n,m)$-function $F$ (and in particular, any Boolean function) can be uniquely represented as a polynomial on $n$ variables with coefficients in $\mathbb{F}_2^m$ of the form:

$$F(x_1,...,x_n)=\sum_{u\in\mathbb{F}_2^n}c(u)\prod_{i=1}^n x_i^{u_i}.$$

This representation is called the *algebraic normal form* of $F$ and its degree $d^\circ(F)$ the *algebraic degree* of the function $F$.

Besides, for $m=n$, $F$ can be identified to a function from the field $\mathbb{F}_{2^n}$ of order $2^n$ to itself, and has then a unique representation as a univariate polynomial of degree smaller than $2^n$ over this field:

$$F(x)=\sum_{i=0}^{2^n-1}c_ix^i,\quad c_i\in\mathbb{F}_{2^n}.$$

For any $k$, $0\le k\le 2^n-1$, the number $w_2(k)$ of nonzero coefficients $k_s\in\{0,1\}$ in the binary expansion $\sum_{s=0}^{n-1}2^sk_s$ of $k$ is called the 2-weight of $k$. The algebraic degree of $F$ is equal to the maximum 2-weight of the exponents $i$ of the polynomial $F(x)$ such that $c_i\neq 0$.

For a function $F:\mathbb{F}_2^n\rightarrow\mathbb{F}_2^n$ and any elements $a,b\in\mathbb{F}_2^n$ we denote

$$\begin{aligned}\delta_F(a,b)&=|\{x\in\mathbb{F}_2^n : F(x+a)+F(x)=b\}|,\\ \lambda_F(a,b)&=\sum_{x\in\mathbb{F}_2^n}(-1)^{b\cdot F(x)+a\cdot x}=W_{b\cdot F}(a).\end{aligned}$$

Note that, for any $a,b\in\mathbb{F}_2^n$, the number $\delta_F(a,b)$ is even. Indeed, if $x_0$ is a solution of $F(x+a)+F(x)=b$ then $x_0+a$ is a solution too.

The function $\lambda_F$ is often called the Walsh transform of $F$. The *nonlinearity* $N_F$ of $F$ is the minimum nonlinearity of all the nonzero linear combinations $b\cdot F$, $b\neq 0$, of its coordinate functions; hence it equals $2^{n-1}-\frac{1}{2}\max_{a,b\in\mathbb{F}_2^n;b\neq 0}|\lambda_F(a,b)|$. The multi-set of the values $\lambda_F(a,b)$, $a,b\in\mathbb{F}_2^n$ does not depend on a particular choice of the inner product in $\mathbb{F}_2^n$. If we identify $\mathbb{F}_2^n$ with $\mathbb{F}_{2^n}$ then we can take $x\cdot y=tr(xy)$, where $tr(x)=x+x^2+...+x^{2^{n-1}}$ is the trace function from $\mathbb{F}_{2^n}$ into $\mathbb{F}_2$.
We also denote

$$\begin{aligned}\varDelta_F&=\{\delta_F(a,b) : a,b\in\mathbb{F}_2^n, a\neq 0\},\\ \varLambda_F&=\{\lambda_F(a,b) : a,b\in\mathbb{F}_2^n, b\neq 0\}.\end{aligned}$$

A function $F:\mathbb{F}_2^n\rightarrow\mathbb{F}_2^n$ is called *almost perfect nonlinear* (APN) if $\varDelta_F=\{0,2\}$. A function $F:\mathbb{F}_2^n\rightarrow\mathbb{F}_2^n$ is called *almost bent* (AB) or *maximum nonlinear* if $\varLambda_F=\{0,\pm 2^{\frac{n+1}{2}}\}$. Obviously, AB functions exist only for $n$ odd.

APN and AB functions are used in cryptography in block ciphers because APN mappings possess the best resistance against the differential cryptanalysis [1] and AB mappings oppose an optimum resistance to both linear [22] and differential attacks.

For affinely equivalent functions $F$ and $F' = L \circ F \circ L'$ (where $L$ and $L'$ are two affine isomorphisms), we have $\Delta_F = \Delta_{F'}$, $\Lambda_F = \Lambda_{F'}$ and if $F$ is a permutation then $\Delta_F = \Delta_{F^{-1}}$, $\Lambda_F = \Lambda_{F^{-1}}$ [5]. Therefore, if $F$ is APN (resp. AB) and $F'$ is affinely equivalent to either $F$ or $F^{-1}$ (if $F$ is a permutation), then $F'$ is also APN (resp. AB).

Table 1 (resp. Table 2) gives all known values of exponents $d$ (up to affine equivalence and up to taking the inverse when a function is a permutation) such that the power function $x^d$ is APN (resp. AB).

Every AB function is APN [6]. The converse is not true, even when $n$ is odd. A counterexample is given by the inverse APN function, which has the algebraic degree $n-1$ while the algebraic degree of any AB function is not greater than $(n+1)/2$ [5].

The inverse function is not APN when $n$ is even, but it is almost APN in the sense that $\Delta_F = \{0, 2, 4\}$. It has been chosen (as elementary block) in the S-box of the AES, with $n = 8$. It has nonlinearity $2^{n-1} - 2^{n/2}$, see [21, 25]. This value is the best known nonlinearity when $n$ is even (see [4] for a list of all known permutations with the same nonlinearity) and knowing whether there exist $(n, n)$-functions with nonlinearity strictly greater than this value is an open question (even for power functions).

Other APN and AB functions have been recently found, which are not equivalent to power functions, see [2].

**Table 1.** Known APN power functions on $\mathbb{F}_{2^n}$

| | Exponents $d$ | Conditions | Proven in |
|---|---|---|---|
| Gold functions | $2^i + 1$ | $gcd(i,n) = 1$, $1 \leq i \leq \frac{n-1}{2}$ | [26] |
| Kasami functions | $2^{2i} - 2^i + 1$ | $gcd(i,n) = 1$, $1 \leq i \leq \frac{n-1}{2}$ | [19],[18] |
| Welch function | $2^t + 3$ | $n = 2t + 1$ | [10] |
| Niho function | $2^t + 2^{\frac{t}{2}} - 1$, $t$ even<br>$2^t + 2^{\frac{3t+1}{2}} - 1$, $t$ odd | $n = 2t + 1$ | [13] |
| Inverse function | $2^{2t} - 1$ | $n = 2t + 1$ | [26] |
| Dobbertin function | $2^{4i} + 2^{3i} + 2^{2i} + 2^i - 1$ | $n = 5i$ | [11] |

**Table 2.** Known AB power functions on $\mathbb{F}_{2^n}$, $n$ odd

| | Exponents $d$ | Conditions | Proven in |
|---|---|---|---|
| Gold functions | $2^i + 1$ | $gcd(i,n) = 1$, $1 \leq i \leq \frac{n-1}{2}$ | [26] |
| Kasami functions | $2^{2i} - 2^i + 1$ | $gcd(i,n) = 1$, $1 \leq i \leq \frac{n-1}{2}$ | [19] |
| Welch function | $2^t + 3$ | $n = 2t + 1$ | [3] |
| Niho function | $2^t + 2^{\frac{t}{2}} - 1$, $t$ even<br>$2^t + 2^{\frac{3t+1}{2}} - 1$, $t$ odd | $n = 2t + 1$ | [17] |

## 3 The Transparency Order

In [27], E. Prouff introduced a new characteristic for S-boxes in block cryptosystems. The *transparency order* of an S-box $F = (f_1, \ldots, f_n)$ on $\mathbb{F}_2^n$ is the number:

$$\mathcal{T}_F = \max_{b \in \mathbb{F}_2^n} \left( |n - 2w_H(b)| - \frac{1}{2^{2n} - 2^n} \sum_{a \in \mathbb{F}_2^{n*}} \left| \sum_{i=1}^{n} (-1)^{b_i} \mathcal{F}(D_a f_i) \right| \right). \quad (2)$$

In this definition (which also exists for $(n, m)$-functions, but we shall study here the case $m = n$ only), the expression inside the brackets is positive for $w_H(b) = 0$ and for $w_H(b) = n$, and it is upper bounded by $n$ for every $b$. Hence, as observed in [27], we have $0 \le \mathcal{T}_F \le n$. As also observed by Prouff, if the coordinate functions $f_i$ of $F$ are bent, then $F$ has obviously worst possible transparency order $n$. However, nothing more is said in [27] on the relationship between the transparency order and the (non)linearities of the $f_i$'s (it seems quite logical that any non-linear S-box will be rather bad against this property, but this has to be proven). We show below four bounds relating the transparency order to the Walsh transforms of the coordinate functions. Bound (3) implies (4) which implies (5) which implies in its turn (6). Bound (5) (resp. bound (6)) shows in particular that, to have a chance of obtaining a good transparency order, two kinds of parameters play a role: the nonlinearities of the coordinate functions (resp. the nonlinearity of the S-box), which would better be not too high, and the sizes of the Walsh supports of these functions and the sizes of the pairwise intersections of these supports.

**Theorem 1.** *Let $F=(f_1, \ldots, f_n)$ be any $(n, n)$-function. For every $i = 1, \ldots, n$, let $S_i$ denote the support of the Walsh transform of $f_i$, that is, the set $\{a \in \mathbb{F}_2^n \mid W_{f_i}(a) \neq 0\}$, and let $L_{f_i}$ denote the linearity of $f_i$ (hence, its nonlinearity equals $2^{n-1} - \frac{1}{2} L_{f_i}$). Then, $\mathcal{T}_F$ is lower bounded by each of the following expressions:*

$$n - \frac{1}{2^n \sqrt{2^n - 1}} \sqrt{\sum_{i=1}^{n} \sum_{a \in \mathbb{F}_2^{n*}} \mathcal{F}^2(D_a f_i) + 2 \sum_{1 \le i < j \le n} \sum_{a \in \mathbb{F}_2^{n*}} \mathcal{F}(D_a f_i) \mathcal{F}(D_a f_j)} \quad (3)$$

$$n - \frac{1}{2^{\frac{3n}{2}} \sqrt{2^n - 1}} \sqrt{\sum_{i=1}^{n} \sum_{a \in \mathbb{F}_2^n} W_{f_i}^4(a) + 2 \sum_{1 \le i < j \le n} \sum_{a \in \mathbb{F}_2^n} W_{f_i}^2(a) W_{f_j}^2(a) - n^2 \, 2^{3n}} \quad (4)$$

$$n - \frac{1}{2^{\frac{3n}{2}} \sqrt{2^n - 1}} \sqrt{\sum_{i=1}^{n} (L_{f_i}^4 \, |S_i|) + 2 \sum_{1 \le i < j \le n} (L_{f_i}^2 \, L_{f_j}^2 \, |S_i \cap S_j|) - n^2 \, 2^{3n}}, \quad (5)$$

*where "$|\ |$" denotes the size. Consequently, denoting by $N_F$ the nonlinearity of $F$, and by $L_F$ its linearity (such that $N_F = 2^{n-1} - \frac{1}{2} L_F$), $\mathcal{T}_F$ is lower bounded by:*

$$n - \frac{1}{2^{3n/2} \sqrt{2^n - 1}} \left( \left( \sum_{i=1}^{n} |S_i| + 2 \sum_{1 \le i < j \le n} |S_i \cap S_j| \right) L_F^4 - n^2 \, 2^{3n} \right)^{1/2}. \quad (6)$$

*Proof*: Applying Cauchy-Schwartz' inequality, we have:

$$\sum_{a\in\mathbb{F}_2^{n\,*}}\left|\sum_{i=1}^{n}(-1)^{b_i}\mathcal{F}(D_af_i)\right| \le \left((2^n-1)\sum_{a\in\mathbb{F}_2^{n\,*}}\left(\sum_{i=1}^{n}(-1)^{b_i}\mathcal{F}(D_af_i)\right)^2\right)^{1/2}.$$

The sum:

$$\sum_{a\in\mathbb{F}_2^{n\,*}}\left(\sum_{i=1}^{n}(-1)^{b_i}\mathcal{F}(D_af_i)\right)^2$$

equals

$$\sum_{i=1}^{n}\sum_{a\in\mathbb{F}_2^{n\,*}}\mathcal{F}^2(D_af_i)+2\sum_{1\le i<j\le n}(-1)^{b_i+b_j}\sum_{a\in\mathbb{F}_2^{n\,*}}\mathcal{F}(D_af_i)\mathcal{F}(D_af_j).$$

Taking for $b$ the null vector or the all-one vector, we get (3).
According to Relation (1), the sum $\sum_{a\in\mathbb{F}_2^{n\,*}}\left(\sum_{i=1}^{n}\mathcal{F}(D_af_i)\right)^2$, that is equal to the expression:
$\sum_{i=1}^{n}\sum_{a\in\mathbb{F}_2^n}\mathcal{F}^2(D_af_i)+2\sum_{1\le i<j\le n}\sum_{a\in\mathbb{F}_2^n}\mathcal{F}(D_af_i)\mathcal{F}(D_af_j)-n^2\,2^{2n}$, equals then:

$$2^{-n}\sum_{i=1}^{n}\sum_{a\in\mathbb{F}_2^n}{W_{f_i}}^4(a)+2^{-n+1}\sum_{1\le i<j\le n}\sum_{a\in\mathbb{F}_2^n}{W_{f_i}}^2(a){W_{f_j}}^2(a)-n^2\,2^{2n}.$$

This proves (4), and we deduce (5) and (6) since $W_{f_i}^2(a)\le L_{f_i}^2$, for every $a$ and for every $i$, and since $L_{f_i}\le L_F$ for every $i$. ◇

**Remarks**:
1. When the $f_i$'s are bent, all of the expressions (3) to (6) equal $\mathcal{T}_F=n$, since for every $i$, $|S_i|$ equals then $2^n$, $L_{f_i}$ equals $2^{n/2}$, and for every $i,j$, $|S_i\cap S_j|$ equals $2^n$.
2. Relation (4) gives

$$\mathcal{T}_F\ge n-\frac{1}{2^{3n/2}\sqrt{2^n-1}}\left(\sum_{a\in\mathbb{F}_{2^n}}\left(\sum_{i=1}^{n}{W_{f_i}}^2(a)\right)^2-n^2\,2^{3n}\right)^{1/2}. \qquad (7)$$

## 4 Power Permutations

Bounds (3), (4), (5) and (6) seem complicated and we can wonder whether they can ever be computed. We shall show that, in the case of power permutations, their complexity decreases and that their computation can be done at least in the cases of Gold functions and of inverse function (see Subsection 4.1).

The coordinate functions of a power function $x^d$ have the form $f_i(x)=tr(b_ix^d)$, where the $b_i$'s are linearly independent. Set $b\ne 0$. Assuming that $d$

is co-prime with $2^n - 1$ (i.e. that the power function is a permutation), the character sum $\sum_{x\in\mathbb{F}_{2^n}}(-1)^{tr(bx^d+ax)}$ equals

$$\sum_{x\in\mathbb{F}_{2^n}}(-1)^{tr\left(b\left(\frac{x}{b^{1/d}}\right)^d+a\left(\frac{x}{b^{1/d}}\right)\right)} = \sum_{x\in\mathbb{F}_{2^n}}(-1)^{tr\left(x^d+\left(\frac{a}{b^{1/d}}\right)x\right)},$$

where $1/d$ denotes the inverse of $d$ mod $2^n - 1$. Hence, denoting the function $tr(bx^d)$ by $f_b$, and the function $tr(x^d)$ by $f$, we have

$$W_{f_b}(a) = W_f\left(\frac{a}{b^{1/d}}\right), \tag{8}$$

and the support of $W_{f_b}$ equals $b^{1/d}S$, where $S$ is the support of $W_f$.

Hence $\sum_{b\in\mathbb{F}^*_{2^n}}\sum_{a\in\mathbb{F}_{2^n}}W^4_{f_b}(a) = (2^n-1)\sum_{a\in\mathbb{F}_{2^n}}W^4_f(a)$.

It is well-known that $\sum_{b\in\mathbb{F}_{2^n}}\sum_{a\in\mathbb{F}_{2^n}}W^4_{f_b}(a)$ equals $2^{2n}$ times the size of the set $\{(x,y,z)\in\mathbb{F}^3_{2^n}\,|\,x^d+y^d+z^d+(x+y+z)^d=0\}$. Indeed, we have

$$\sum_{b\in\mathbb{F}_{2^n}}\sum_{a\in\mathbb{F}_{2^n}}W^4_{f_b}(a) =$$

$$\sum_{b\in\mathbb{F}_{2^n}}\sum_{a\in\mathbb{F}_{2^n}}\sum_{x,y,z,t\in\mathbb{F}_{2^n}}(-1)^{tr(b(x^d+y^d+z^d+t^d)+a(x+y+z+t))} =$$

$$\sum_{x,y,z,t\in\mathbb{F}_{2^n}}\left(\sum_{b\in\mathbb{F}_{2^n}}(-1)^{tr(b(x^d+y^d+z^d+t^d))}\right)\left(\sum_{a\in\mathbb{F}_{2^n}}(-1)^{tr(a(x+y+z+t))}\right),$$

and the sum $\sum_{b\in\mathbb{F}_{2^n}}(-1)^{tr(b(x^d+y^d+z^d+t^d))}$ is null if $x^d+y^d+z^d+t^d\neq 0$ (resp. the sum $\sum_{a\in\mathbb{F}_{2^n}}(-1)^{tr(a(x+y+z+t))}$ is null if $x+y+z+t\neq 0$).

Since the condition $x^d+y^d+z^d+(x+y+z)^d = 0$ is satisfied under the sufficient condition that two elements among $x, y$ and $z$ are equal (the number of such cases equals $3\cdot 2^{2n} - 2^{n+1}$), $\sum_{a\in\mathbb{F}_{2^n}}W^4_f(a)$ is therefore lower bounded by $\frac{1}{2^n-1}\left(2^{2n}\cdot(3\cdot 2^{2n}-2^{n+1})-\sum_{a\in\mathbb{F}_{2^n}}W^4_0(a)\right)=\frac{1}{2^n-1}\left(2^{2n}\cdot(3\cdot 2^{2n}-2^{n+1})-2^{4n}\right) = 2^{3n+1}$, as well as $\sum_{a\in\mathbb{F}_{2^n}}W^4_{f_b}(a)$, for every $b\neq 0$. We deduce:

**Lemma 1.** *If $F=(f_1,\ldots,f_n)$ is a power permutation, then*

$$\sum_{i=1}^n\sum_{a\in\mathbb{F}_{2^n}}W^4_{f_i}(a) =$$

$$\frac{n\,2^{2n}}{2^n-1}\left(|\{(x,y,z)\in\mathbb{F}^3_{2^n}\,|\,x^d+y^d+z^d+(x+y+z)^d=0\}|-2^{2n}\right)\geq$$

$$n\cdot 2^{3n+1}.$$

If $x^d$ is APN, then the condition $x^d+y^d+z^d+(x+y+z)^d=0$ is satisfied if and only if two elements among $x, y$ and $z$ are equal, and $\sum_{a\in\mathbb{F}_{2^n}} W_f^4(a)$ equals $2^{3n+1}$, as well as $\sum_{a\in\mathbb{F}_{2^n}} W_{f_b}^4(a)$, for every $b\neq 0$. Hence $\sum_{i=1}^n \sum_{a\in\mathbb{F}_{2^n}} W_{f_i}^4(a) = n\cdot 2^{3n+1}$.

Let us consider now, for $c\notin\mathbb{F}_2$, the sum $\sum_{a\in\mathbb{F}_{2^n}} {W_f}^2(a){W_{f_c}}^2(a)$ involved in (4). According to (8), it is equal to $\frac{1}{2^n-1}$ times

$$\sum_{b\in\mathbb{F}_{2^n}^*}\sum_{a\in\mathbb{F}_{2^n}} W_{f_b}^2(a)W_{f_b}^2\left(\frac{a}{c^{1/d}}\right)=$$

$$\sum_{b\in\mathbb{F}_{2^n}^*}\sum_{a\in\mathbb{F}_{2^n}} W_{f_b}^2(a)W_{f_{bc}}^2(a)=$$

$$\left(\sum_{b\in\mathbb{F}_{2^n}}\sum_{a\in\mathbb{F}_{2^n}}\sum_{x,y,z,t\in\mathbb{F}_{2^n}}(-1)^{tr(b(x^d+y^d+cz^d+ct^d)+a(x+y+z+t))}-2^{4n}\right).$$

Hence, we have:

**Lemma 2.** *If $F(x)=x^d$ is a power permutation and $f(x)=tr(F(x))$, $f_c(x)=tr(cF(x))$, $c\notin\mathbb{F}_2$, then*

$$\sum_{a\in\mathbb{F}_{2^n}} {W_f}^2(a){W_{f_c}}^2(a)= \tag{9}$$

$$\frac{2^{2n}}{2^n-1}\left(|\{(x,y,z)\in{\mathbb{F}_{2^n}}^3\,|\,x^d+y^d+cz^d+c(x+y+z)^d=0\}|-2^{2n}\right).$$

There does not seem to exist a nice general lower bound on this expression, even when $x^d$ is APN (taking $x=y$ only leads only to the positivity of $\sum_{a\in\mathbb{F}_{2^n}} {W_f}^2(a){W_{f_c}}^2(a)$). We first consider the particular case of the inverse function.

## 4.1 The Inverse Function and the S-Box of the AES

Let $d=2^n-2=-1$ [mod $2^n-1$] ($x^d$ equals $1/x$ if $x\neq 0$ and equals 0 if $x=0$). We know that $x^d$ is APN when $n$ is odd, and is not APN when $n$ is even. Recall that this function is used with $n=8$ as the basic S-box in the AES.

In Appendix 1, we show why it seems impossible to calculate the exact value of the transparency order of the inverse function. So we must use the method initiated in the introduction of Section 4 (see Lemmas 1 and 2).

We study in Appendix 2, for any $c\neq 0$, the solutions of the equation $x^d+y^d+cz^d+c(x+y+z)^d=0$. We obtain:

- if $c=1$, then:
  - if $n$ is odd, the number of solutions of the equation $x^d+y^d+z^d+(x+y+z)^d=0$ equals $2^{2n}+4(2^n-1)+2(2^n-1)(2^n-2)=3\cdot 2^{2n}-2^{n+1}$; we calculated already this number (the function is APN);

- if $n$ is even, the number of solutions of the equation $x^d + y^d + z^d + (x + y + z)^d = 0$ equals $2^{2n} + 8(2^n - 1) + 2(2^n - 1)(2^n - 2) = 3 \cdot 2^{2n} + 2^{n+1} - 4$, since $tr(1) = 0$; this number could have also been calculated by using the results of Nyberg [26].

– if $c \neq 1$, then
- if $tr(c) = tr(1/c) = 0$, the number of solutions of the equation $x^d + y^d + cz^d + c(x + y + z)^d = 0$ equals $2^{2n} + 4(2^n - 1) + 4(2^n - 1) + 2(2^n - 1)(2^{n-1} - 2) = 2 \cdot 2^{2n} + 3 \cdot 2^n - 4$
- if $tr(c) = 0$ and $tr(1/c) = 1$ or if $tr(c) = 1$ and $tr(1/c) = 0$, the number of solutions of the equation $x^d + y^d + cz^d + c(x + y + z)^d = 0$ equals $2^{2n} + 4(2^n - 1) + 2(2^n - 1)(2^{n-1} - 2) = 2 \cdot 2^{2n} - 2^n$
- if $tr(c) = tr(1/c) = 1$, the number of solutions of the equation $x^d + y^d + cz^d + c(x+y+z)^d = 0$ equals $2^{2n} + 2(2^n - 1)(2^{n-1} - 2) = 2 \cdot 2^{2n} - 5 \cdot 2^n + 4$.

Note that, in the case $c \neq 1$, the greatest number that we have obtained is $2 \cdot 2^{2n} + 3 \cdot 2^n - 4$. According to Relation (9), we deduce that Bound (4) gives, if $n$ is odd:

$$\mathcal{T}_{\mathcal{F}} \geq$$

$$n - \left( \frac{n(2 \cdot 2^{2n} - 2^{n+1}) + n(n-1)(2^{2n} + 3 \cdot 2^n - 4)}{2^n(2^n - 1)^2} - \frac{n^2}{(2^n - 1)} \right)^{1/2} =$$

$$n - \frac{1}{2^{n/2}(2^n - 1)} \left(4n^2(2^n - 1) + n(2^{2n} - 5 \cdot 2^n + 4)\right)^{1/2} \approx n - \sqrt{\frac{n}{2^n}}$$

and if $n$ is even:

$$\mathcal{T}_{\mathcal{F}} \geq$$

$$n - \left( \frac{n(2 \cdot 2^{2n} + 2^{n+1} - 4) + n(n-1)(2^{2n} + 3 \cdot 2^n - 4)}{2^n(2^n - 1)^2} - \frac{n^2}{(2^n - 1)} \right)^{1/2} =$$

$$n - \frac{1}{2^{n/2}(2^n - 1)} \left(4n^2(2^n - 1) + n(2^{2n} - 2^n)\right)^{1/2} \approx n - \sqrt{\frac{n}{2^n}}.$$

Hence, the inverse function has bad transparency order! In particular, in the case of the S-box of the AES ($n = 8$), our bound gives that $\mathcal{T}_{\mathcal{F}} \geq 7.8$, which is close to $n = 8$.

## 4.2 The Gold Functions

For $F(x) = x^{2^i+1}$ ($\gcd(i, n)$=1, $n$ odd), $f_b(x) = tr(bx^{2^i+1})$ ($b \neq 0$) and for $a \neq 0$, we have $D_a f_b(x) = tr(bax^{2^i} + ba^{2^i}x + ba^{2^i+1}) = tr(((ba)^{2^{n-i}} + ba^{2^i})x + ba^{2^i+1})$. Hence, $\mathcal{F}(D_a f_b)$ equals $\pm 2^n$ if $(ba)^{2^{n-i}} + ba^{2^i} = 0$ and is null otherwise. We have $(ba)^{2^{-i}} + ba^{2^i} = 0$ if and only if $ba + b^{2^i}a^{2^{2i}} = 0$, that is, if and only if $b^{2^i-1}a^{2^{2i}-1} = 1$, or equivalently $ba^{2^i+1} = 1$, since $\gcd(i, n)$=1. Hence, for every $a \neq 0$, there exists exactly one $b$ such that $\mathcal{F}(D_a f_b) \neq 0$, and therefore at most one $i$ such that $\mathcal{F}(D_a f_i) \neq 0$. We deduce that $\mathcal{T}_F$ equals in fact $n - \frac{1}{2^{2n}-2^n} \sum_{a \in \mathbb{F}_2^{n*}} \sum_{i=1}^{n} |\mathcal{F}(D_a f_i)| = n - \frac{1}{2^{2n}-2^n} \sum_{i=1}^{n} \left( \sum_{a \in \mathbb{F}_2^{n*}} |\mathcal{F}(D_a f_i)| \right)$. Note that this observation is valid whatever is the evenness of $n$. Here, $n$ is odd, so that $F$ is a permutation. For every $b \neq 0$, there exists then a unique $a \neq 0$ such that $ba^{2^i+1} = 1$ and we deduce $\mathcal{T}_F = n - \frac{n2^n}{2^{2n}-2^n} = n - \frac{n}{2^n-1}$. We see that the transparency order of Gold functions is bad too. These functions had already the drawback of having low degree.

The fact that we could calculate the exact value of $\mathcal{T}_F$ in the case of Gold AB functions is an opportunity of seeing whether, at least in this case, our bound (6) is good or not. It is well known that the support of the Walsh transform of the function $tr(x^{2^i+1})$ ($n$ odd, $\gcd(i,n) = 1$) equals $H = \{a \in \mathbb{F}_{2^n} \,|\, tr(a) = 1\}$. Indeed, since function $tr(x^{2^i+1})$ is quadratic (i.e. has degree 2), for every $a \in \mathbb{F}_{2^n}$, the function $tr(x^{2^i+1}) + tr(ax)$ is unbalanced if and only if its restriction to the kernel of its associated symplectic form, that is, the vectorspace $E = \{x \in \mathbb{F}_{2^n} \,|\, \forall y \in \mathbb{F}_{2^n}, tr(x^{2^i+1}) + tr(y^{2^i+1}) + tr((x+y)^{2^i+1}) = 0\}$, is constant. We have $E = \{x \in \mathbb{F}_{2^n} \,|\, x^{2^i} + x^{2^{n-i}} = 0\} = \{x \in \mathbb{F}_{2^n} \,|\, x^{2^{2i}} + x = 0\} = \{0\} \cup \{x \in \mathbb{F}_{2^n}{}^* \,|\, x^{2^{2i}-1} = 1\} = \{0, 1\}$. It is a simple matter to see that, for every $k \neq j$, we have $|b_j^{1/(2^i+1)}\, H \cap b_k^{1/(2^i+1)}\, H| = 2^{n-2}$, since the $b_j$'s are nonzero and pairwise distinct. We deduce that Relation (6) gives

$$\begin{aligned}\mathcal{T}_F &\geq n - \frac{1}{2^{3n/2}\sqrt{2^n-1}} \left( \left( n\, 2^{n-1} + n(n-1)\, 2^{n-2} \right) 2^{2n+2} - n^2\, 2^{3n} \right)^{1/2} \\ &= n - \frac{\sqrt{n}}{\sqrt{2^n-1}}.\end{aligned}$$

The difference between $\mathcal{T}_F = n - \frac{n}{2^n-1}$ and $n - \frac{\sqrt{n}}{\sqrt{2^n-1}}$ is negligible with respect to $\mathcal{T}_F$.

## 5 Conclusion

We were able to show that the transparency orders of two highly nonlinear S-boxes (including the S-box of the AES) are bad. This confirms the intuition that nonlinear mappings used as S-boxes may be unable to avoid using heavy countermeasures to DPA attacks (and the resulting penalties on efficiency). But it remains to show that the other functions included in tables 1 and 2 (for instance) have also bad transparency orders.

## Acknowledgement

We wish to thank E. Prouff for useful discussions.

## References

1. E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *Journal of Cryptology*, vol. 4, No.1, pp. 3-72, 1991.
2. L. Budaghyan, C. Carlet and A. Pott. New Classes of Almost Bent and Almost Perfect Nonlinear Polynomials. Proceedings of the Workshop on Coding and Cryptography 2005, Bergen, pp. 306-315, 2005.

3. A. Canteaut, P. Charpin and H. Dobbertin. Binary $m$-sequences with three-valued crosscorrelation: A proof of Welch's conjecture. *IEEE Trans. Inform. Theory*, 46 (1), pp. 4-8, 2000.
4. A. Canteaut, P. Charpin, and H. Dobbertin. Weight divisibility of cyclic codes, highly nonlinear functions on GF($2^m$) and crosscorrelation of maximum-length sequences. *SIAM Journal on Discrete Mathematics*, 13(1), pp. 105–138, 2000.
5. C. Carlet, P. Charpin and V. Zinoviev. Codes, bent functions and permutations suitable for DES-like cryptosystems. *Designs, Codes and Cryptography*, 15(2), pp. 125-156, 1998.
6. F. Chabaud and S. Vaudenay. Links between differential and linear cryptanalysis, *Advances in Cryptology -EUROCRYPT'94, Lecture Notes in Computer Science*, Springer-Verlag, New York, 950, pp. 356-365, 1995.
7. S. Chari, C. Jutla, J. Rao and P. Rohatgi. Towards sound approaches to counteract power analysis attacks. CRYPTO'99, *Advances in Cryptology, Lecture Notes in Computer Science* 1666, pp. 398-412, 1999.
8. C. Clavier, J.-S. Coron and N. Dabbous. Differential power analysis in the presence of hardware countermeasures. CHES 2000, *Lecture Notes in Computer Science* 1965, pp. 252-263, 2000.
9. J.-S. Coron and L. Goubin. On Boolean and Arithmetic Masking against Differential Power Analysis. CHES00, pp. 231–237, 2000.
10. H. Dobbertin. Almost perfect nonlinear power functions over $GF(2^n)$: the Welch case, *IEEE Trans. Inform. Theory*, 45, pp. 1271-1275, 1999.
11. H. Dobbertin. Almost perfect nonlinear power functions over $GF(2^n)$: a new case for $n$ divisible by 5. D. Jungnickel and H. Niederreiter Eds. *Proceedings of Finite Fields and Applications FQ5*, Augsburg, Germany, Springer, pp. 113-121, 2000.
12. J. Golic and C. Tymen. Multiplicative masking and power analysis of AES. CHES02, pp. 198–212, 2002
13. H. Dobbertin. Almost perfect nonlinear power functions over $GF(2^n)$: the Niho case, *Inform. and Comput.*, 151, 57-72, 1999.
14. L. Goubin and J. Patarin. DES and differential power analysis - the duplication method. CHES'99, *Lecture Notes in Computer Science* 1717, pp. 158-172, 1999.
15. S. Guilley, P. Hoogvorst and R. Pascalet. Differential power analysis model and some results. Smart Card Research ann Advanced Applications VI - Cardis 2004, *Kluwer Academic Publishers*, pp. 127-142, 2004.
16. A. A. Hasan. Power analysis attacks and algorithmic approaches to their countermeasures for Koblitz cryptosystems. CHES 2000, *Lecture Notes in Computer Science* 1965, pp. 93-108, 2000.
17. H. Hollmann and Q. Xiang. A proof of the Welch and Niho conjectures on cross-correlations of binary $m$-sequences. *Finite Fields and Their Applications 7*, pp. 253-286, 2001.
18. H. Janwa and R. Wilson. Hyperplane sections of Fermat varieties in $P^3$ in char. 2 and some applications to cyclic codes. *Proceedings of AAECC-10, Lecture Notes in Computer Science*, vol. 673, Berlin, Springer-Verlag, pp. 180-194, 1993.
19. T. Kasami. The weight enumerators for several classes of subcodes of the second order binary Reed-Muller codes. *Inform. and Control*, 18, pp. 369-394, 1971.
20. P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. CRYPTO'96, *Advances in cryptology, Lecture Notes in Computer Science* 1109, pp. 104-113, 1996.
21. G. Lachaud and J. Wolfmann. The Weights of the Orthogonals of the Extended Quadratic Binary Goppa Codes. *IEEE Trans. Inform. Theory*, vol. 36, pp. 686-692, 1990.

22. M. Matsui. Linear cryptanalysis method for DES cipher. *Advances in Cryptology-EUROCRYPT'93, Lecture Notes in Computer Science*, Springer-Verlag, pp.386-397, 1994.
23. R. Mayer Sommer. Smartly analysing the simplicity and the power of simple power analysis on smartcards. CHES 2000, *Lecture Notes in Computer Science* 1965, pp. 78-92, 2000.
24. T. Messerges, E. Dabbish and R. Sloan. Power analysis attacks on smartcards. CHES'99, *Lecture Notes in Computer Science* 1717, pp. 144-157, 1999.
25. K. Nyberg. On the construction of highly nonlinear permutations. *Advances in Cryptology, EUROCRYPT' 92, Springer Verlag, Lecture Notes in Computer Science* 658, pp. 92-98, 1993.
26. K. Nyberg. Differentially uniform mappings for cryptography, *Advances in Cryptology, EUROCRYPT'93, Lecture Notes in Computer Science*, Springer-Verlag, New York, 765, pp. 55-64, 1994.
27. E. Prouff. DPA attacks and S-boxes. FSE 2005, *Lecture Notes in Computer Science* 3557, pp. 424-441, 2005.
28. E. Trichina and D. DeSeta and L. Germani. Simplified Adaptive Multiplicative Masking for AES. CHES'02, pp. 187–197, 2002.

## Appendix 1

In this appendix, we see whether it is possible to calculate the exact value of $\mathcal{T}_F$ when $F(x)$ is the inverse function $x^{-1}$, equal to $\frac{1}{x}$ if $x \neq 0$ and to 0 if $x = 0$. For every $a, b \neq 0$, we have $D_a f_b(x) = tr\left(\frac{ab}{x(x+a)}\right)$ if $x \neq 0, x \neq a$ and $D_a f_b(x) = tr\left(\frac{b}{a}\right)$ if $x = 0$ and if $x = a$.

Hence, $\mathcal{F}(D_a f_b)$ equals $\sum_{x\in\mathbb{F}_{2^n}} (-1)^{tr\left(abx^{-1}(x+a)^{-1}\right)} - 2 + 2(-1)^{tr(b/a)}$, that is, by changing variable $x$ into $ax$: $\sum_{x\in\mathbb{F}_{2^n}} (-1)^{tr\left(a^{-1}b(x^2+x)^{-1}\right)} - 2 + 2(-1)^{tr(a^{-1}b)}$. Since $x^2 + x$ ranges uniformly over the hyperplane $\{u \in \mathbb{F}_{2^n} \mid tr(u) = 0\}$ when $x$ ranges over $\mathbb{F}_{2^n}$, we deduce that $\mathcal{F}(D_a f_b)$ equals $2\sum_{u\in\mathbb{F}_{2^n} \mid tr(u)=0} (-1)^{tr\left(a^{-1}bu^{-1}\right)} - 2 + 2(-1)^{tr(a^{-1}b)}$, that is, equals $\sum_{u\in\mathbb{F}_{2^n}} [(-1)^{tr\left(a^{-1}bu^{-1}\right)} + (-1)^{tr\left(u+a^{-1}bu^{-1}\right)}] - 2 + 2(-1)^{tr(a^{-1}b)}$, that is, changing $u$ into $ab^{-1}u$:

$$\mathcal{F}(D_a f_b) = \sum_{u\in\mathbb{F}_{2^n}} (-1)^{tr\left(u^{-1}\right)} + \sum_{u\in\mathbb{F}_{2^n}} (-1)^{tr\left(ab^{-1}u+u^{-1}\right)} - 2 + 2(-1)^{tr(a^{-1}b)}.$$

Since $x^{-1}$ is a permutation, the first of these two sums is null. The second one is known under the name of *Kloosterman* sum. It is proved in [21] that, when $ab^{-1}$ ranges over $\mathbb{F}_{2^n}$, the set of the values of this sum equals the set of all the integers congruent with -1 modulo 4, in the range $[-2^{\frac{n}{2}+1}, 2^{\frac{n}{2}+1}]$. But the distribution of these values is not known and this gives therefore no information on the possible value of the expression inside the brackets in (2).

## Appendix 2

Let us consider the equation $x^d + y^d + cz^d + c(x + y + z)^d = 0$ for any $c \neq 0$. If $x = y$, then it is satisfied. *This makes* $2^{2n}$ *solutions.* We study now the solutions such that $x \neq y$.

*Case 1*: if $z = 0$ or $z = x + y$, then the equation reduces to $x^d + y^d + c(x + y)^d = 0$.
- If $x = 0$ (and $y \neq 0$) or $y = 0$ (and $x \neq 0$), then it is satisfied if $c = 1$ and it is not satisfied if $c \neq 1$.
- If $x \neq 0$ and $y \neq 0$, it is equivalent to $\frac{1}{x} + \frac{1}{y} + \frac{c}{x+y} = 0$, that is, $x^2 + y^2 + cxy = 0$, or equivalently $\left(\frac{x}{cy}\right)^2 + \frac{x}{cy} + \frac{1}{c^2} = 0$. Thus, if $tr\left(\frac{1}{c^2}\right) = 0$, that is, if $tr\left(\frac{1}{c}\right) = 0$, then the equation admits two solutions in $x$, for every $y \neq 0$; note that these two solutions satisfy $x \neq 0$ and $x \neq y$, since $c$ is nonzero. *This makes altogether* $2\ [2\ (2^n - 1) + 2\ (2^n - 1)] = 8\ (2^n - 1)$ *solutions if* $c = 1$ *and* $tr\left(\frac{1}{c}\right) = 0$ *(that is, if* $n$ *is even),* $2\ [2\ (2^n - 1)] = 4\ (2^n - 1)$ *if* $c = 1$ *and* $tr\left(\frac{1}{c}\right) = 1$ *(that is, if* $n$ *is odd) or if* $c \neq 1$ *and* $tr\left(\frac{1}{c}\right) = 0$ *and none if* $c \neq 1$ *and* $tr\left(\frac{1}{c}\right) = 1$.
*Case 2*: if $z \neq 0$, $z \neq x + y$ and $x = 0$ or $y = 0$ - say $y = 0$ (and $x \neq 0$), then the equation reduces to $\frac{1}{x} + \frac{c}{z} + \frac{c}{x+z} = 0$, or equivalently $xz + z^2 + cx^2 = 0$, that is, $\left(\frac{z}{x}\right)^2 + \frac{z}{x} + c = 0$. This last equation admits two solutions $z$, for every $x \neq 0$, if $tr(c) = 0$ and none otherwise. Note that the two solutions, if they exist, satisfy $z \neq 0$ and $z \neq x + y = x$, since $c$ is nonzero. *This makes altogether* $2\ (2\ (2^n - 1)) = 4\ (2^n - 1)$ *solutions if* $tr(c) = 0$ *and none otherwise.*
*Case 3*: if $z \neq 0$, $z \neq x + y$, $x \neq 0$ and $y \neq 0$, then the equation is equivalent to $\frac{1}{x} + \frac{1}{y} + \frac{c}{z} + \frac{c}{x+y+z} = 0$, that is, $(x + y + z)(yz + xz + cxy) + cxyz = 0$, that is, $(x + y)(cxy + (x + y)z + z^2) = 0$. Since $x \neq y$, this is equivalent to

$$\left(\frac{z}{x+y}\right)^2 + \frac{z}{x+y} + \frac{cxy}{x^2+y^2} = 0. \tag{10}$$

Two cases concerning $x$ and $y$ can occur:

- if $tr\left(\frac{cxy}{x^2+y^2}\right) = 1$, then Equation (10) has no solution.
- if $tr\left(\frac{cxy}{x^2+y^2}\right) = 0$, then Equation (10) has two solutions $z$. Note that, since $x$ and $y$ are distinct and nonzero, and since $c$ is nonzero, these two solutions satisfy $z \neq 0$ and $z \neq x + y$.

Let us determine the number of ordered pairs $(x, y)$, with $x$ and $y$ distinct and nonzero, such that $tr\left(\frac{cxy}{x^2+y^2}\right) = 0$. We have $\frac{xy}{x^2+y^2} = \frac{x}{x+y} + \left(\frac{x}{x+y}\right)^2$, and $\frac{x}{x+y}$ ranges uniformly over $\mathbb{F}_{2^n} \setminus \mathbb{F}_2$ when $(x, y)$ ranges over ${\mathbb{F}_{2^n}^*}^2 \setminus \{(x, x); x \in \mathbb{F}_{2^n}^*\}$; hence $\frac{xy}{x^2+y^2}$ ranges uniformly over $\{u \in \mathbb{F}_{2^n}^* \mid tr(u) = 0\}$ when $(x, y)$ ranges over ${\mathbb{F}_{2^n}^*}^2 \setminus \{(x, x); x \in \mathbb{F}_{2^n}^*\}$ (more precisely, every element $\{u \in \mathbb{F}_{2^n}^* \mid tr(u) = 0\}$ equals $\frac{xy}{x^2+y^2}$ for $\frac{(2^n-1)(2^n-2)}{2^{n-1}-1} = 2^{n+1} - 2$ ordered pairs $(x, y)$). Thus, if $c = 1$ then the condition $tr\left(\frac{cxy}{x^2+y^2}\right) = 0$ is satisfied for all of the $(2^n - 1)(2^n - 2)$ ordered pairs $(x, y)$, and if $c \notin \mathbb{F}_2$, it is satisfied for $(2^{n+1} - 2)(2^{n-2} - 1) = (2^n - 1)(2^{n-1} - 2)$ ordered pairs. *This makes altogether* $2(2^n - 1)(2^n - 2)$ *solutions if* $c = 1$ *and* $2(2^n - 1)(2^{n-1} - 2)$ *if* $c \notin \mathbb{F}_2$.

Summarizing, this gives what is stated at Subsection 4.1.

# How to Construct Universal One-Way Hash Functions of Order $r$

Deukjo Hong[1], Jaechul Sung[2], Seokhie Hong[1], and Sangjin Lee[1]

[1] Center for Information Security Technologies(CIST), Korea University, Seoul, Korea
{hongdj, hsh, sangjin}@cist.korea.ac.kr
[2] Department of Mathematics, University of Seoul, Seoul, Korea
jcsung@uos.ac.kr

**Abstract.** At ASIACRYPT 2004, Hong et al. introduced the notion of UOWHFs of order $r > 0$. A UOWHF has the order $r$ if it is infeasible for any adversary to win the game for UOWHF where the adversary is allowed $r$ adaptive queries to the hash function oracle before outputting his target message. They showed that if a UOWHF has the order $r$, its some-round MD (Merkle-Damgård) or some-level TR (TRee) extension is a UOWHF. Since MD and TR extensions do not require additional key values except the key of compression functions for hashing, their result means that the order of UOWHFs can be useful for minimizing the total key length. In this paper we study how to construct such UOWHFs of order $r$. As the first step, we observe Bellare-Rogaway UOWHF and Naor-Yung UOWHF. It is shown that Bellare-Rogaway UOWHF has the order 0 and that Naor-Yung UOWHF has the order 1. We generalize the construction of Naor-Yung UOWHF based on a one-way permutation to that of the UOWHF of order $r$.

**Keywords :** Hash Function, Collision Resistant Hash Function (CRHF), Universal One-Way Hash Function (UOWHF), Higher Order Universal One-Way Hash Function.

## 1 Introduction

The notion of universal one-way hash function (UOWHF) was introduced by Naor and Yung [6]. It is a family of functions $H = \{h_K\}_{K \in \mathcal{K}}$ with $h_K : \{0,1\}^n \to \{0,1\}^m$, for which the following game is infeasible: the adversary chooses $x$ as a target message, then receives a key $K \in \mathcal{K}$ selected uniformly at random and wins if he can find $x'$ such that $x \neq x'$ and $h_K(x) = h_K(x')$.

UOWHFs have been studied with skepticism of collision-resistant hash functions (CRHFs) that the CRHF is too strong assumption to design. Bellare and Rogaway [1] showed that the Merkle-Damgård extension, which is the most popular method to be used to extend a CRHF with a finite domain to one with a larger domain, cannot be applicable to UOWHFs by giving an example of a UOWHF with a finite domain whose 2-round MD (Merkle-Damgård) extension

© Springer-Verlag Berlin Heidelberg 2005

is not a UOWHF. Their example means that it cannot be avoided that the longer message is hashed, the longer key is required in any extension for UOWHFs. So, later work have been concentrated on minimizing key length. Bellare and Rogaway suggested two extensions for UOWHFs with linear structure, LH and XLH, and two extensions for UOWHFs with tree structure, TH and XTH.

Shoup [11] proposed a nice improvement of XLH extension, and Mironov [5] proved that it achieves minimal key length among all XLH-like extension. Recent work by Sarkar [8, 9, 10] and Lee et al. [4] have provided different extensions with parallel structures with varying trade-off between degree of parallelism and key length.

## 1.1 CRHFs and UOWHFs

UOWHF is a strictly weaker primitive than CRHF. Simon [12] showed that there is an oracle relative to which UOWHFs exist but not CRHFs. He also stated that it is not likely that a convincing construction of a collision-resistant hash function can be built on nothing more than the assumption of a one-way permutation. However, in the point that Naor and Yung [6] suggested a construction of a UOWHF based on a one-way permutation, it is more plausible to recognize a family of functions as a UOWHF than a CRHF.

Extending a UOWHF has the problem that the key length increases according to the length of a message. Such problem does not exist in case of CRHFs since MD or TR (TRee) extension does not increase the key size.

## 1.2 Order of Universal One-Way Hash Functions

Hong et al. [3] introduced the notion of UOWHFs of order $r$. A UOWHF $H$ of order $r$ is a family of hash functions $\{h_K\}_{K\in\mathcal{K}}$ such that it is feasible for any adversary to win the following game: a key $K$ is selected uniformly at random from $\mathcal{K}$, the adversary asks $r$ adaptive queries to the hash function oracle $h_K(\cdot)$, chooses $x$ as a target message, then receives the key $K$ and wins if he can find $x'$ such that $x \neq x'$ and $h_K(x) = h_K(x')$. According to this terminology, the standard definition of UOWHF is a UOWHF of order 0. They showed that if a UOWHF $H$ has the order $r$, then its $(r+1)$-round MD extension is a UOWHF and that if a UOWHF $H = \{h_K : \{0,1\}^{dn} \rightarrow \{0,1\}^n\}_{K\in\mathcal{K}}$ has order $r = (d^l - d)/(d-1)$, then its $l$-level TR extension is a UOWHF. Note that MD and TR constructions do not require additional random key values except the key of compression functions for hashing. So, these results allow us to reduce the key length required for extending a UOWHF of order 0 if we consider the order of the compression function as a UOWHFs.

For example, let $H = \{h_K : \{0,1\}^{n+m} \rightarrow \{0,1\}^n\}_{K\in\mathcal{K}}$ be a UOWHF, where $\mathcal{K} = \{0,1\}^k$. Assume that we use XLH extension for extending $H$. If we don't consider the order of $H$, we should use $(k + Ln/m)$-bit key to hash a $L$-bit message, where $L$ is a multiple of $m$. However, if the order of $H$ is known as $r$, we can see that $(r+1)$-round MD extension $F = \{f_K : \{0,1\}^{n+(r+1)m} \rightarrow \{0,1\}^n\}_{K\in\mathcal{K}}$ of $H$ is a UOWHF. So, the amount of required key bits for hashing

**Table 1.** The comparison of the shortest key length required by the extension methods with a linear structure when the compression UOWHF $H$ has order 0 and $r > 0$. Here $k$ is the key length of $H$ and $h_K : \{0,1\}^{n+m} \to \{0,1\}^n$. $L$ is the length of the message to be hashed (measured in bits) and a multiple of $m$.

| Extension | Key Length (order 0) | Key Length (order $r$) | Reference |
|---|---|---|---|
| LH | $\frac{Lk}{m}$ | $\lceil \frac{Lk}{m(r+1)} \rceil$ | [1] |
| XLH | $k + \frac{Ln}{m}$ | $\lceil \frac{1}{r+1} \left(k + \frac{Ln}{m}\right) \rceil$ | [1] |
| Shoup | $k + \lceil \log_2 \frac{Ln}{m} \rceil$ | $\lceil \frac{1}{r+1} \left(k + \lceil \log_2 \frac{Ln}{m} \rceil\right) \rceil$ | [11] |

a $L$-bit message by using XLH extension is decreased to $\lceil (k + Ln/m)/(r+1) \rceil$. Table 1 compares the key length required for extending a UOWHF of order 0 and of order $r$ with sequential extension methods such as LH, XLH, and Shoup's method. Note that a UOWHF is no more than one of order 0 if we don't consider its order.

Notice that we treated extensions with linear structure for simplicity but we can give a similar (but little more complicated) argument about extensions with tree structure.

## 1.3 Motivation and Contribution

As mentioned in the previous subsection, the order of UOWHFs can be useful for minimizing the total key length required for extension. According to the results in [3], applying such extension as LH, XLH, etc. directly to a UOWHF is wasting random key materials. Note that many researchers have worked hard to build extending methods with optimal key length. Our work has been launched from the question: does there exist a construction of UOWHFs of any order $r > 0$? In this paper, we will suggest a provably secure and generalized construction of UOWHFs of order $r > 0$, as Naor and Yung [6] suggested a one-way-permutation-based construction of a UOWHF.

As the first step of our work, we check the order of Bellare-Rogaway UOWHF $H^{\mathsf{BR}}$ and Naor-Yung UOWHF $H^{\mathsf{NY}}$. We show that $H^{\mathsf{BR}}$ has order 0. The construction of $H^{\mathsf{BR}}$ is not based on other primitives, but this result is interesting in the point that it is a well-matched example with the contraposition of a theorem in [3] that the order of the UOWHF is less than $r-1$ if $r$-round MD extension of a UOWHF is not a UOWHF. Note that in [1] it has been already shown that 2-round MD extension of $H^{\mathsf{BR}}$ is not a UOWHF.

We also prove that $H^{\mathsf{NY}}$ has order 1. Therefore, according to work in [3], 2-round MD extension of $H^{\mathsf{NY}}$ is a UOWHF. This implies that we can reduce the key length required for extending $H^{\mathsf{NY}}$ by half if we consider its order. We generalize $H^{\mathsf{NY}}$ to a provably secure construction of a UOWHF with order $r > 0$ based on a one-way permutation, $H^{\mathsf{NY}_r}$. $H^{\mathsf{NY}_r}$ has the key length of $(r+1)n$ bits when the input length is $n$, while $H^{\mathsf{NY}}$ has the key length of $2n$. There is a tradeoff between order and key length in the sense of the key length for extending a UOWHF. Assume that an extension method requires the additional key length of $t$ bits for extending a UOWHF of order 0 in order to hash a message. Then, the total key length for extending $H^{\mathsf{NY}_r}$ is $(r+1)n+\lceil\frac{t}{r+1}\rceil$. This relation between $t$ and $r$ can be useful for determining the optimal order for some schemes. For example, in the environment where the message length is fixed, $t$ is also fixed. In such environment, the key length is minimized if we use the UOWHF of order $r$ such that $(r+1)^2 \cong \frac{t}{n}$.

Roughly according to another result in [3], the larger the order is, the closer to a CRHF a UOWHF is. However, we show that for any $r$, there exists a UOWHF of order $r$ which is not a CRHF.

## 2 Preliminaries

### 2.1 Notations

Let $\{0,1\}^n$ be the set of all the strings of length $n$. When $a$ and $b$ are bit strings, $a||b$ denotes the concatenation of $a$ and $b$. When $b$ is a value, $a \longleftarrow b$ means that $b$ is computed and then $a$ is set to $b$. Similarly, when both $U$ and $V$ are sets, $U \longleftarrow V$ means that $U$ is put to $V$. When $U$ is a set and $u$ is an element of $U$, $u \overset{\$}{\longleftarrow} U$ means that $u$ is uniformly at random from $U$. When $A$ is an algorithm or a program, $A(x) \longrightarrow a$ means that $A$ on the input $x$ outputs $a$.

### 2.2 Definitions of Cryptographic Hash Functions

We give formal definitions of CRHF, UOWHF, UOWHF of order $r$, and OWP (One-Way Permutation).

**Definition 1 (CRHF).** *Let $H = \{h_K\}_{K\in\mathcal{K}}$ be a family of functions with $h_K : \mathcal{M} \longrightarrow \mathcal{C}$ for $K \in \mathcal{K}$. $H$ is $(t,\varepsilon)$-CRHF if any adversary $A$ cannot win the following game with the probability $\varepsilon$ and within the running time $t$.*

```
Game^CRHF(H, A)
1 :    $K \overset{\$}{\longleftarrow} \mathcal{K}$
2 :    $A(K) \longrightarrow (x, x')$
3 :    if $x \neq x'$ and $h_K(x) = h_K(x')$, then output "$A$ wins"
4 :    else, output "$A$ loses"
```

**Definition 2 (UOWHF).** *Let $H = \{h_K\}_{K\in\mathcal{K}}$ be a family of functions with $h_K : \mathcal{M} \longrightarrow \mathcal{C}$ for $K \in \mathcal{K}$. $H$ is $(t,\varepsilon)$-UOWHF if any adversary $A = (A_1, A_2)$ cannot win the following game with the probability $\varepsilon$ and within the running time $t$.*

```
Game^UOWHF(H, A)
1 :    $K \overset{\$}{\longleftarrow} \mathcal{K}$
2 :    $A_1(null) \longrightarrow (x, State)$
3 :    $A_2(K, x, State) \longrightarrow x'$
4 :    if $x \neq x'$ and $h_K(x) = h_K(x')$, then output "$A$ wins"
5 :    else, output "$A$ loses"
```

We can exchange the order of Step 1 and 2 in $\mathbf{Game}^{\mathrm{UOWHF}}(H, A)$ since the two steps are independent and the order of them does not effect Step 3. So, the notion of the conventional UOWHFs is equivalent to that of UOWHFs of order 0.

**Definition 3 (UOWHF of order $r$).** *Let $H = \{h_K\}_{K \in \mathcal{K}}$ be a family of functions with $h_K : \mathcal{M} \longrightarrow \mathcal{C}$ for $K \in \mathcal{K}$. $H$ is $(t, \varepsilon)$-UOWHF($r$) if any adversary $A = (A_1, A_2)$ cannot win the following game with the probability $\varepsilon$ and within the running time $t$.*

```
Game^{UOWHF(r)}(H, A)
100 :   $K \xleftarrow{\$} \mathcal{K}$
200 :   $Q \longleftarrow \varnothing$
300 :   if $r > 0$ do
310 :       for $i = 1, \cdots, r$ do
311 :           $A_1(Q) \longrightarrow x_i$
312 :           $y_i \longleftarrow h_K(x_i)$
313 :           $Q \longleftarrow Q \cup \{(x_i, y_i)\}$
400 :   $A_1(Q) \longrightarrow (x, State)$
500 :   $A_2(K, x, State) \longrightarrow x'$
600 :   if $x \neq x'$ and $h_K(x) = h_K(x')$, then output "$A$ wins"
700 :   else, output "$A$ loses"
```

Note that $A_1$ never gets the key $K$ and has just $r$ adaptive accesses to $h_K(\cdot)$ which behaves as like an oracle. The order of $h$ means the number of adaptive queries which the adversary can ask to the hash function oracle $h_K(\cdot)$ when it has no information of the key.

Finally, we introduce the definition of a one-way permutation. In the definition of a one-way function or permutation, we are not forced to consider the key space [7].

**Definition 4 (OWP: One-Way Permutation).** *A permutation $f : \{0,1\}^n \rightarrow \{0,1\}^n$ is $(t, \varepsilon)$-OWP if any adversary $A$ cannot win the following game with the probability $\varepsilon$ and within the running time $t$.*

```
Game^{OWP}(f, A)
1 :   $x \xleftarrow{\$} \{0,1\}^m$
2 :   $A(f(x)) \longrightarrow x'$
3 :   if $x = x'$, then output "$A$ wins"
4 :   else, output "$A$ loses"
```

## 2.3 MD and TR Extensions

Let $H = \{h_K\}_{K \in \mathcal{K}}$ be a family of functions with $h_K : \{0,1\}^{n+m} \rightarrow \{0,1\}^n$. Then $r$-round MD extension of $H$, $\mathrm{MD}_r[H]$ is $\{\mathrm{MD}_r[h_K] : \{0,1\}^{n+rm} \rightarrow \{0,1\}^n\}_{K \in \mathcal{K}}$.

Let $x \in \{0,1\}^{n+rm}$ be $x_0||x_1||\cdots||x_r$ where $x_0 \in \{0,1\}^n$ and $x_i \in \{0,1\}^m$ for $i = 1, \cdots, r$. Then, for $K \in \mathcal{K}$, $\mathrm{MD}_r[h_K](x)$ is computed as follows.

```
Algorithm MD_r[h_K](x)
1 :  $y_0 \longleftarrow x_0$
2 :  for $i = 1, ..., r$ do
3 :      $y_i \longleftarrow h_K(y_{i-1}||x_i)$
4 :  output $y_r$
```

Let $H = \{h_K\}_{K \in \mathcal{K}}$ be a family of functions with $h_K : \{0,1\}^{dm} \rightarrow \{0,1\}^m$. Then $l$-level TR extension of $H$, $\mathrm{TR}_l[H]$ is $\{\mathrm{TR}_l[h_K] : \{0,1\}^{d^l m} \rightarrow \{0,1\}^m\}_{K \in \mathcal{K}}$. Let $x \in \{0,1\}^{d^l m}$ be $x_0^1||x_0^2||\cdots||x_0^{d^l}$ where $x_0^i \in \{0,1\}^m$ for $i = 1, \cdots d^l$. Then, for $K \in \mathcal{K}$, $\mathrm{TR}_l[h_K](x)$ is computed as follows.

```
Algorithm TR_l[h_K](x)
1 :  for $i = 1, ..., l$ do
2 :      for $j = 1, ..., d^{l-i}$ do
3 :          $x_i^j \longleftarrow h_K(x_{i-1}^{(j-1)d+1}||\cdots||x_{i-1}^{jd})$
4 :  output $x_l^1$
```

## 2.4 Review of Previous Work

In this subsection we review the results in [3]. The following theorems state the relationships among UOWHFs of various order and CRHFs. Let $T_H$ be the worst-case time to compute $h_K(\cdot)$.

**Theorem 1 (In [3]).** *Let $H = \{h_K\}_{K \in \mathcal{K}}$ be a family of functions with $h_K : \{0,1\}^m \longrightarrow \{0,1\}^c$ and $m > c$. Then,*

1. *$H$ is a $(t, \varepsilon)$-UOWHF $\Leftrightarrow$ $H$ is a $(t, \varepsilon)$-UOWHF$(0)$.*
2. *For any $r \geq 0$, $H$ is a $(t', \varepsilon)$-UOWHF$(r+1)$ $\Rightarrow$ $H$ is a $(t, \varepsilon)$-UOWHF$(r)$, where $t = t' - \Theta(T_H + m + c)$.*
3. *For any $r \geq 0$, $H$ is a $(t', \varepsilon)$-CRHF $\Rightarrow$ $H$ is a $(t, \varepsilon)$-UOWHF$(r)$, where $t = t' - \Theta(r)(T_H + m + c)$.*

According to the above theorem, UOWHFs of order $r > 1$ are in the place between CRHF and conventional UOWHF (of order 0), and as $r$ increases, the notion of UOWHF$(r)$s is closer to the notion of CRHFs. However, we show that the two notions do not meet. For any $r > 0$, we can construct a UOWHF which is not a CRHF but has the order $r$. Let $H = \{h_K\}_{K \in \mathcal{K}}$ be a UOWHF$(r)$ where $\mathcal{K} = \{0,1\}^k$ and $h_K : \{0,1\}^{k+m} \longrightarrow \{0,1\}^k$. Then we can construct the following family of functions: $H' = \{h'_K\}_{K \in \mathcal{K}}$ where for $K, x \in \{0,1\}^k$ and $y \in \{0,1\}^m$, $h'_K(x||y)$ is defined as follows.

$$h'_K(x||y) = \begin{cases} 0||h_K(x||y) & \text{if } x \neq K, \\ 1||K & \text{if } x = K. \end{cases} \tag{1}$$

Clearly, $H'$ is not a CRHF, but still UOWHF$(r)$.

**Theorem 2.** *Let $H$ be a $(t, \varepsilon)$-UOWHF$(r)$. Then, $H'$ defined above is a $(t', \varepsilon')$-UOWHF$(r)$ where $\epsilon' = \frac{2^k}{2^k - r}\epsilon - \frac{1}{2^k(2^k - r)}$ and $t = t' - \Theta(rT_H)$.*

*Proof.* See Appendix.

The following theorems are main contributions of [3] together with introduction of the notion of UOWHF of order $r$.

**Theorem 3 (In [3]).** *Let $H = \{h_K\}_{K\in\mathcal{K}}$ be a family of functions with $h_K : \{0,1\}^{c+m} \to \{0,1\}^c$. If $H$ is a $(t', \varepsilon')$-UOWHF$(r)$ then,* $\mathrm{MD}_{r+1}[H]$ *is a $(t, \varepsilon)$-UOWHF, where $\varepsilon = (r+1)\varepsilon'$ and $t = t' - \Theta(r)(T_H + m + c)$.*

**Theorem 4 (In [3]).** *Let $H = \{h_K\}_{K\in\mathcal{K}}$ be a family of functions with $h_K : \{0,1\}^{dc} \to \{0,1\}^c$. If $H$ is a $(t', \varepsilon')$-UOWHF$(r)$ and $r = (d^l - d)/(d-1)$. Then* $\mathrm{TR}_l[H]$ *is a $(t, \varepsilon)$-UOWHF, where $\varepsilon = (r+1)\varepsilon'$, and $t' = t + \Theta(d^l)(T_H + dc)$.*

## 3 Order of Bellare-Rogaway UOWHF

Bellare and Rogaway [1] gave an example of a UOWHF whose 2-round MD extension is not a UOWHF. This example is as follows. Let $H^0 = \{h^0_K\}_{K\in\mathcal{K}}$ be a $(t, \varepsilon)$-UOWHF where $\mathcal{K} = \{0,1\}^k$ and for $K \in \mathcal{K}$, $h^0_K : \{0,1\}^{c+k+m} \longrightarrow \{0,1\}^c$. Then $H^{\mathsf{BR}} = \{h^{\mathsf{BR}}_K\}_{K\in\mathcal{K}}$ with $h^{\mathsf{BR}}_K : \{0,1\}^{c+k+m} \longrightarrow \{0,1\}^{c+k}$ is defined as follows. For $K \in \{0,1\}^k, x \in \{0,1\}^c, y \in \{0,1\}^k$, and $z \in \{0,1\}^m$, let

$$h^{\mathsf{BR}}_K(x||y||z) = \begin{cases} h^0_K(x||y||z)||K & \text{if } y \neq K \\ 1^c||1^k & \text{if } y = K. \end{cases}$$

The following is proved in [1].

**Proposition 1.** *$H^{\mathsf{BR}}$ has the following properties.*

1. *$H^{\mathsf{BR}}$ is $(t', \varepsilon - 2^{-k+1})$-UOWHF, where $t' = t - \Theta(m + 2k)$*
2. *There is an adversary that wins the* $\mathbf{Game}^{\mathrm{UOWHF}}(MD_2[H^{\mathsf{BR}}], \cdot)$*, within the time $t'' = \Theta(m+k)$ and with the success probability $\varepsilon'' = 1 - 2^{-k}$.*

This example has made many cryptographers abandon applying MD extension to UOWHFs and research other extending methods. We show that this counter-example satisfies the contraposition of the statement of Theorem 3. The point is that $H^{\mathsf{BR}}$ is a UOWHF but $MD_2[H^{\mathsf{BR}}]$ is not. According to the contraposition, the order of $H^{\mathsf{BR}}$ is not 1 and so its order should be zero.

**Theorem 5.** *The order of $H^{\mathsf{BR}}$ is zero.*

*Proof.* The proof is easy. We can make an adversary $A = (A_1, A_2)$ which works in the $\mathbf{Game}^{\mathrm{UOWHF}(1)}(H^{\mathsf{BR}}, A)$ and has a very simple strategy. $A_1$ asks a query which is selected uniformly at random from $\{0,1\}^{c+k+m}$. Once $A_1$ is given the answer by the hash function oracle $h^{\mathsf{BR}}_K(\cdot)$, it can know the key $K$ with the probability $1 - 2^{-k}$. $A_1$ selects and outputs $x||K||z$ as a target message where $x, z \in \{0,1\}^k$. $A_2$ on the input $(K, x||K||z)$ selects and outputs $x'||K||z'$ as a sibling message such that $x \neq x'$ or $z \neq z'$. By the definition of $H^{\mathsf{BR}}$, the two messages collide under $h^{\mathsf{BR}}_K$. Hence the order of $H^{\mathsf{BR}}$ is less than 1, so it is zero. □

## 4 Order of Naor-Yung UOWHF

In [6] a construction of a UOWHF based on a one-way permutation is introduced. This example shows that if a one-way permutation exists then a UOWHF also exists. It is constructed as follows.

Let $f : \{0,1\}^k \longrightarrow \{0,1\}^k$ be a one-way permutation and $\mathsf{Chop} : \{0,1\}^k \longrightarrow \{0,1\}^{k-1}$ be a function which simply chops the least significant bit. $G = \{g_K\}_{K \in \mathcal{K}}$ is a family of functions where $\mathcal{K} = \{0,1\}^{2k}$, and for nonzero $a \in \{0,1\}^k$ and $b, x \in \{0,1\}^k$, $g_{a||b}(x)$ is defined as

$$g_{a||b}(x) = \mathsf{Chop}(ax + b).$$

It is considered that all the computations are in the finite field $\mathbb{F}_{2^k}$. Then Naor-Yung UOWHF $H^{\mathsf{NY}}$ is defined as $\{h_K^{\mathsf{NY}} = g_K \circ f\}_{K \in \mathcal{K}}$.

Note that $G$ has collision accessibility: Given that $g_{a||b}(x_1) = g_{a||b}(x_2)$ it is possible to generate $a, b \in \{0,1\}^k$ within a reasonable time such that $g_{a||b}(x_1) = g_{a||b}(x_2)$ with equal probability over all functions in $G$ which obey the restriction of the same function value for $x_1$ and $x_2$. It is proved that $H^{\mathsf{NY}}$ is a UOWHF from the properties of $f$ and $G$ in [6]. We generalize and formalize the definition of the collision accessibility to be suitable for our work as follows.

**Definition 5 (($t, r$)-Collision-Accessible).** *Let $E$ be a family of functions. $E$ is $(t, r)$-collision-accessible if for given $r$ distinct points $(x_1, y_1), \cdots, (x_r, y_r)$ and a pair of inputs $(x, x')$ it is possible to generate $e \in E$ within time $t$ such that $e(x_1) = y_1, \cdots, e(x_r) = y_r$ and $e(x) = e(x')$ with equal probability over all functions in $E$ which obey the restriction that $e(x_1) = y_1, \cdots, e(x_r) = y_r$ and $e(x) = e(x')$.*

From this definition we can claim the following.

**Lemma 1.** *$G$ is $(t, 1)$-collision-accessible, where $t = O(k^3)$.*

*Proof.* It is sufficient to show that $a$ and $b$ are easily determined for a point $(x_1, y_1)$ and a collision $(x, x')$. For the collision $(x, x')$, the following equation should hold.

$$\mathsf{Chop}(ax + b) = \mathsf{Chop}(ax' + b).$$

By cancellation, the above equation is changed as follows.

$$\mathsf{Chop}(a(x - x')) = 0.$$

Since $x \neq x'$, the above equation can be written like this:

$$a(x - x') = 1.$$

So, $a$ is uniquely determined as $(x - x')^{-1}$. Then, $b$ is determined such that $\mathsf{Chop}(ax_1 + b) = y_1$ for the pair of query-answer $(x_1, y_1)$. The time for generating $a$ and $b$ is dominated by the time for the computation of $(x - x')^{-1}$, so $t = O(k^3)$. □

Let $F$ be a $(t', \varepsilon)$-OWP. Now, we show that the order of $H^{\mathsf{NY}}$ is 1.

**Theorem 6.** *$H^{\mathsf{NY}}$ is a $(t'', \varepsilon')$-UOWHF(1) for $t'' = t' - O(k^3 + T_f)$ and $\varepsilon' = \frac{2^k}{2^k-1}\varepsilon - \frac{1}{2^k-1}$ where $T_f$ is the worst-case time to compute $f(\cdot)$.*

*Proof.* We assume that there is an adversary $A = (A_1, A_2)$ which works in the $\mathbf{Game}^{\mathrm{UOWHF(1)}}(H^{\mathsf{ny}}, A)$. An adversary $B$ which works in the $\mathbf{Game}^{\mathrm{OWP}}(f, B)$ can be built as like Fig. 1.

```
Game^OWP(f, B)
10 :   $w \xleftarrow{\$} \{0,1\}^k$
20 :   $z \longleftarrow f(w)$
30 :   $B(z)$ do
31 :       $A_1(null)$ asks a query $x_1$
32 :       $y_1 \longleftarrow \{0,1\}^{k-1}$
33 :       $A_1(\{x_1, y_1\}) \longrightarrow (x, State)$
34 :       if $f(x) = z$ then $x' \longleftarrow x$ and output $x'$
35 :       Choose $K \in \{0,1\}^{2k}$ such that
                $g_K(f(x_1)) = y_1$ and $g_K(f(x)) = g_K(z)$
36 :       $A_2(K, x, State) \longrightarrow x'$
37 :       output $x'$
40 :   if $x' = w$ then output "$B$ wins"
50 :   else, output "$B$ loses"
```

**Fig. 1.** $\mathbf{Game}^{\mathrm{OWP}}(f, B)$: $B$ has an adversary $A = (A_1, A_2)$ as a subroutine, which works in $\mathbf{Game}^{\mathrm{UOWHF(1)}}(H^{\mathsf{NY}}, A)$.

The $\mathbf{Game}^{\mathrm{OWP}}(f, B)$ starts with giving $z = f(w)$ to the adversary $B$ where $w$ is selected uniformly at random from $\{0,1\}^k$. $B$ on the input $z$ runs the adversary $A$ and simulates the oracle of $A_1$. For the query $x_1$ of $A_1$, $B$ returns the answer $y_1$ which is selected uniformly at random from $\{0,1\}^k$. When $A_1$ outputs a target message $x$ in Step 33, if $f(x) = z$ then $B$ stops simulating $\mathbf{Game}^{\mathrm{UOWHF(1)}}(H^{\mathsf{NY}}, A)$ and outputs $x$. If $f(x) \neq z$ then $B$ chooses $K = a||b \in \{0,1\}^{2k}$. Since $w$ and $y_1$ were chosen randomly, it is guaranteed that $K$ is determined with uniform probability from lemma 1. $A_2$ on the input $(K, x, State)$ outputs $x'$. Note that each $H_K^{\mathsf{NY}}$ is a 2-1 function. If Step 36 is successful, i.e $h_K^{\mathsf{NY}}(x) = h_K^{\mathsf{NY}}(x')$ but $x \neq x'$, then $f(x') = z$ and so $x' = w$. Hence, the probability that $B$ wins is as follows.

$$\begin{aligned}\Pr[B \text{ wins}] &= \Pr[f(x) = z] + \Pr[A \text{ wins}] \cdot \Pr[f(x) \neq z] \\ &= 2^{-k} + \frac{2^k - 1}{2^k} \cdot \Pr[A \text{ wins}].\end{aligned}$$

The running time of $B$ is that of $A$ plus the overhead of Step 10, 20, 32, and 35. The overhead is about $O(k^3 + T_f)$. □

## 5 Construction of UOWHFs of Order $r$

We introduce a construction of a UOWHF of order $r > 0$ by generalizing $H^{\mathsf{NY}}$. We consider a family of functions $G = \{g_K\}_{K \in \mathcal{K}}$ where $K = \{0,1\}^{(r+1)k}$. For each $a_r \neq 0$ and $a_{r-1}, \cdots, a_0, x \in \{0,1\}^k$, $g_{a_r||\cdots||a_1||a_0}(x)$ is defined by

$$g_{a_r||\cdots||a_1||a_0}(x) = \mathsf{Chop}(a_r x^r + \cdots + a_1 x + a_0).$$

All the computations are in $\mathbb{F}_{2^k}$.

**Lemma 2.** *$G$ is $(t, r)$-collision-accessible, where $t = O(r^3k^3)$.*

*Proof.* The proof is similar to that of lemma 1. Let $(x_1, y_1), \cdots, (x_r, y_r)$ be $r$ distinct points and a pair of inputs $(x, x')$. We choose $r$ 1-bits $b_1, \cdots, b_r$ uniformly at random from $\{0,1\}$. For the $r$ points, we should find $K = a_r||\cdots||a_0$ such that the following equations hold.

$$\begin{aligned} a_r x_1^r + \cdots + a_1 x_1 + a_0 &= y_1||b_1, \\ &\vdots \\ a_r x_r^r + \cdots + a_1 x_r + a_0 &= y_r||b_r. \end{aligned}$$

There exists an index $i \in [0, r]$ such that every $a_j \neq a_i$ is represented by an affine transform of $a_i$. It means that the entropy of $K = a_r||\cdots||a_0$ decreases from $2^{(r+1)k}$ to $2^k$. Then, the other constraint,

$$\mathsf{Chop}(a_r x^r + \cdots + a_0) = \mathsf{Chop}(a_r x'^r + \cdots + a_0)$$

can be re-written by $a_i$ as follows.

$$\mathsf{Chop}(\alpha a_i + \beta) = 0$$

where $\alpha$ and $\beta$ are constants in $\mathbb{F}_{2^k}$. Two cases can be considered for the above equation. The former case is $\alpha a_i + \beta = 0$, and the latter case is $\alpha a_i + \beta = 1$. $a_i$ can be determined in both cases within the running time $O(r^3k^3)$ which is dominated by the time for computing a matrix multiplication. □

Let $f : \{0,1\}^k \longrightarrow \{0,1\}^k$ be a $(t', \varepsilon)$-UOWHF. Then, a family of functions $H^{\mathsf{NY}_r}$ is constructed as $\{h_K^{\mathsf{NY}_r} = g_K \circ f\}_{K \in \mathcal{K}}$. We show that the order of $H^{\mathsf{NY}_r}$ is $r$.

**Theorem 7.** *$H^{\mathsf{NY}_r}$ is a $(t'', \varepsilon')$-UOWHF($r$) for $t'' = t' - O(r^3k^3 + T_f)$ and $\varepsilon' = \frac{2^k(2r-1)}{2^k-1}\varepsilon - \frac{2r-1}{2^k-1}$ where $T_f$ is the worst-case time to compute $f(\cdot)$.*

*Proof.* The proof is similar to Theorem 6. Assume that there exists an adversary $A$ which works in $\mathbf{Game}^{\mathrm{UOWHF}(r)}(H^{\mathsf{NY}_r}, A)$. An adversary $B$ which works in $\mathbf{Game}^{\mathrm{OWP}}(f, B)$ can be built from $A$ as like Fig. 2.

In Step 340, if $f(x) = z$ for the target message $x$ which $A_1$ outputs then $B$ stops the simulation of $\mathbf{Game}^{\mathrm{UOWHF}(r)}(H^{\mathsf{NY}_r}, A)$ and outputs $x$. If $f(x) \neq z$

```
Game^OWP(f, B)
100 :   $w \xleftarrow{\$} \{0,1\}^k$
200 :   $z \longleftarrow f(w)$
300 :   $B(z)$ do
310 :       $Q \longleftarrow \varnothing$
320 :       for $i = 1, \cdots, r$ do
321 :           $A_1(Q)$ asks $i$-th query $x_i$
322 :           $y_i \xleftarrow{\$} \{0,1\}^{k-1}$
323 :           $Q \longleftarrow Q \cup \{(x_i, y_i)\}$
330 :       $A_1(Q) \longrightarrow (x, State)$
340 :       if $f(x) = z$ then $x' \longleftarrow x$ and output $x'$
350 :       Choose $K \in \{0,1\}^{(r+1)k}$ such that
                $g_K(f(x_1)) = y_1, \cdots, g_K(f(x_r)) = y_r$,
                and $g_K(f(x)) = g_K(z)$
360 :       $A_2(K, x, State) \longrightarrow x'$
370 :       output $x'$
400 :   if $x' = w$ then output "$B$ wins"
500 :   else, output "$B$ loses"
```

**Fig. 2.** $\mathbf{Game}^{\mathrm{OWP}}(f, B)$: $B$ has an adversary $A = (A_1, A_2)$ as a subroutine, which works in $\mathbf{Game}^{\mathrm{UOWHF}(r)}(H^{\mathsf{NY}_r}, A)$.

then $B$ keeps the simulation and outputs what $A_2$ produces. So, the probability that $B$ wins is as follows.

$$\begin{aligned}\Pr[B \text{ wins}] &= \Pr[f(x) = z] + \Pr[A \text{ wins}] \cdot \Pr[f(x') = z | A \text{ wins}] \cdot \Pr[f(x) \neq z] \\ &= 2^{-k} + \Pr[A \text{ wins}] \cdot \Pr[f(x') = z | A \text{ wins}] \cdot \frac{2^k - 1}{2^k}.\end{aligned}$$

Since $G_K$ is a composition of a polynomial with degree $r$ and $\mathsf{Chop}$, there exist at most distinct $2r - 1$ inputs of $H_K^{\mathsf{NY}_r}$ colliding with $x$. Since $f(w)$ is a root of the equation $a_r X^r + \cdots a_1 X + a_0 = G_K(z)$ and $w$ is chosen uniformly at random, $\Pr[f(x') = z | A \text{ wins}]$ is greater than or equal to $\frac{1}{2r-1}$.

The running time of $B$ is that of $A$ plus the overhead, where the overhead is $O(k^3 + T_f)$. □

## 6 Conclusions

In this paper, we showed the existence of UOWHFs of order $r > 0$ by studying how to construct them. As the first step, we researched the orders of known UOWHF constructions — Bellare-Rogaway and Naor-Yung constructions. It was shown that the order of Bellare-Rogaway UOWHF is 0, and that the order Naor-Yung UOWHF is 1. We constructed a UOWHF of order $r > 0$ by generalizing Naor-Yung UOWHF. With this generalized construction, we give the answer 'yes' for the question: does there exist a construction of UOWHFs of any order $r > 0$ ?, which was asked in the beginning of the paper. This generalized construction is not creative, but it is an interesting result which shows that for any proper $r > 0$, UOWHFs of order $r$ exists as long as one-way permutations do.

This result obviously contributes to optimal usage of known UOWHF extending methods. UOWHFs of order $r > 0$ can be extended to some-round MD or some-level TR without additional key values, and such extensions can be used as a new compression function. So, if we choose the optimal order $r$ for a particular environment (e.g. the message length is fixed) and construct a UOWHF of order $r$, then the total key length required for hashing messages can be almost minimized. If someone find a more efficient method to construct a UOWHF of order $r > 0$ whether its security is based on any other cryptographic primitives or not, it will be a very interesting result.

## Acknowledgement

This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment).

## References

1. M. Bellare and P. Rogaway, "Collision-resistant hashing: Towards making UOWHFs practical," *Advances in Cryptology – CRYPTO'97*, LNCS 1294, B.S. Kaliski Jr. ed., Springer-Verlag, pp. 470–484, 1997.
2. R. Chen and E. Biham, "Near Collisions of SHA-0," *Advances in Cryptology – CRYPTO 2004*, LNCS 3152, M. Franklin ed., Springer-Verlag, pp. 290–305, 2004.
3. D. Hong, B. Preneel, and S. Lee, "Higher Order Universal One-Way Hash Functions," *Advances in Cryptology – ASIACRYPT 2004*, LNCS 3329, P. J. Lee ed., Springer-Verlag, pp. 201–213, 2004.
4. W. Lee, D. Chang, S. Lee, S. Sung, and M. Nandi, "New Parallel Domain Extenders for UOWHF," *Advances in Cryptology – ASIACRYPT 2003*, LNCS 2894, C. S. Laih ed., Springer-Verlag, pp. 208–227, 2003.
5. I. Mironov, "Hash Functions: From Merkle-Damgård to Shoup," In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, LNCS 2045, G. Goos, J. Hartmanis and J. van Leeuwen ed., Springer-Verlag, pp. 166-181, 2001.
6. M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," *Proceedings of the 21st Annual Symposium on Theory of Computing*, ACM, pp. 33–43, 1989.
7. P. Rogaway and T. Shrimpton, "Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance," *Fast Software Encryption 2004*, LNCS 3017, B. Roy and W. Meier ed., Springer-Verlag, pp. 371–388, 2004.
8. P. Sarkar, "Constuction of UOWHF: Tree Hashing Revisited," Cryptology ePrint Achive, http://eprint.iacr.org/2002/058.

9. P. Sarkar, "Domain Extenders for UOWHF: A Generic Lower Bound om Key Expansion and a Finite Binary Tree Algorithm," Cryptology ePrint Archive, http://eprint.iacr.org/2003/009.
10. P. Sarkar, "Masking Based Domain Extenders for UOWHFs: Bounds and Constructions," *Advances in Cryptology – ASIACRYPT 2004*, LNCS 3329, P. J. Lee ed., Springer-Verlag, pp. 187–200, 2004.
11. V. Shoup, "A composite theorem for universal one-way hash functions," *Advances in Cryptology – EUROCRYPT 2000*, LNCS 1807, B. Preneel ed., Springer-Verlag, pp. 445–452, 2000.
12. D. Simon, "Finding collisions on a one-way street: can secure hash functions be based on general assumptions?," *Advances in Cryptology – EUROCRYPT'98*, LNCS 1403, K. Nyberg ed., Springer-Verlag, pp. 334–345, 1998.
13. Y. Zheng, T. Matsumoto, H. Imai, "Connections among several versions of one-way hash functions," *Trans. IEICE E*, Vol. E73, No. 7, pp. 1092–1099, July 1990.

## A Proof of Theorem 2

*Proof.* Assume that there is an adversary $A = (A_1, A_2)$ which plays in the $\mathbf{Game}^{\mathrm{UOWHF}(r)}(H', A)$. We use $A$ to build an adversary $B = (B_1, B_2)$ which plays in the $\mathbf{Game}^{\mathrm{UOWHF}(r)}(H, B)$, depicted in Fig. 3.

```
Game^{UOWHF(r)}(H, B)
100 :  $K \xleftarrow{\$} \mathcal{K}$
200 :  $B_1$ sets $Q_A$ and $Q_B$ to be $\varnothing$
300 :  for $i = 1, ..., r$ do
310 :      $B_1(Q_B)$ runs $A_1(Q_A)$ to get $A_1$'s query $x_i||y_i$
               where $x_i \in \{0,1\}^k$ and $y_i \in \{0,1\}^m$
320 :      $B_1(Q_B)$ asks a query $x_i||y_i$ to $h_K(\cdot)$
330 :      $z_i \longleftarrow h_K(x_i||y_i)$
340 :      $Q_B \longleftarrow Q_B \cup \{(x_i||y_i, z_i)\}, Q_A \longleftarrow Q_A \cup \{(x_i||y_i, 0||z_i)\}$
400 :  $A_1(Q_A) \longrightarrow (x||y, State_A), B_1(Q_B) \longrightarrow (x||y, State_B)$
           where $State_B = State_A$
500 :  $B_2(K, x||y, State_B)$ do
510 :      If there is an index $i$ such that $x_i = K$,
               then $x'||y' \xleftarrow{\$} \{0,1\}^{k+m}$ and output $x'||y'$
520 :      $A_2(K, x||y, State_A) \longrightarrow x'||y'$
530 :      output $x'||y'$
600 :  if $x||y \neq x'||y'$ and $h_K(x||y) = h_K(x'||y')$ then output "$B$ wins"
700 :  else, output "$B$ loses"
```

**Fig. 3.** $\mathbf{Game}^{\mathrm{UOWHF}(r)}(H, B)$: $B$ has an adversary $A = (A_1, A_2)$ as a subroutine, which works in $\mathbf{Game}^{\mathrm{UOWHF}(r)}(H', A)$

In the beginning of $\mathbf{Game}^{\mathrm{UOWHF}(r)}(H, B)$, $B_1$ runs $A_1$ to obtain its queries for $i = 1, ..., r$. If $A_1$ gives a query $x_i||y_i$ then $B_1$ send it to the hash function oracle $h_K(\cdot)$. When $B_1$ gets $z_i = h_K(x_i||y_i)$, $B_1$ returns $0||z_i$ to $A_1$ as the answer for $A_1$'s $i$-th query. $B_1$'s target message is the same as $A_1$'s target message $x||y$. Since $Q_A = \{(x_1||y_1, 0||z_1), \cdots, (x_r||y_r, 0||z_r)\}$ while $Q_B = \{(x_1||y_1, z_1), \cdots, (x_r||y_r, z_r)\}$, the adversary $A$ supposes that every $x_i$ is not equal to $K$. However, if there is an index $i$ such that $x_i = K$, then $B_2(K, x||y, State_B)$ outputs a random value from $\{0,1\}^{k+m}$. Thus, in this case, the probability that $x'||y'$ collides with $x||y$ for $h_K$ is $2^{-k}$. Otherwise, $B_2(K, x||y, State_B)$ outputs $A_2(K, x||y, State_A)$'s result. Let $\mathsf{E}$ be the event that there is an index $i$ such that $x_i = K$ and $\bar{\mathsf{E}}$ be the complementary event of $\mathsf{E}$. Then, the probability that $B$ wins is like this:

$$\Pr[B \ \text{ wins}] = \Pr[\mathsf{E}] \cdot 2^{-k} + \Pr[\bar{\mathsf{E}}] \cdot \Pr[A \ \text{ wins}]. \tag{2}$$

We are left with bounding $\Pr[\mathsf{E}]$. Let $\mathsf{D}_1$ be the event that all $x_i$'s are same, and let $\mathsf{D}_2$ be the event that all $x_i$'s are distinct. Then, the probability that $\mathsf{E}$ occurs is bounded as follows.

$$\Pr[\mathsf{E}|\mathsf{D}_1] \leq \Pr[\mathsf{E}] \leq \Pr[\mathsf{E}|\mathsf{D}_2]. \tag{3}$$

It is easy to understand that $\Pr[\mathsf{E}|\mathsf{D}_1] = 2^{-k}$ and $\Pr[\mathsf{E}|\mathsf{D}_2] = 1 - r \cdot 2^{-k}$. So, (2) can be bounded below by $2^{-2k} + r \cdot 2^{-k} \cdot \Pr[A \ \text{ wins}]$.

The running time of $B$ is at most that of $A$ plus the overhead $\Theta(rT_H)$. □

# Towards Optimal Double-Length Hash Functions

Mridul Nandi

Applied Statistics Unit,
Indian Statistical Institute,
Kolkata, India
mridul_r@isical.ac.in

**Abstract.** In this paper we design several double length hash functions and study their security properties in the random oracle model. We design a class of double length hash functions (and compression functions) which includes some recent constructions [4, 6, 10]. We also propose a secure double length hash function which is as efficient as the insecure concatenated classical hash functions [7].

## 1 Introduction

An $n$-bit compression function or hash function is said to be "ideal" or "maximally secure" if the best collision attack requires $\Omega(2^{n/2})$ many queries which is same as the complexity of the *birthday attack*. To increase the security level, one can design $2n$-bit compression functions and hash functions (also known as *double length compressions or hash functions* respectively). Potentially one can expect a security level of $\Omega(2^n)$ for a $2n$-bit hash function. But the trivial concatenated hash functions $H \parallel G$ is not secure where one of the $H$ and $G$ is a classical hash function [7]. In this paper we design several double length compression functions and double length hash functions based on single length compression functions. More precisely we consider the following problem;

**Problem :** *Given a secure compression function, $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ (or $s$ compression functions $f_1, \cdots, f_s : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$), $m > 0$, design a secure compression function $F : \{0,1\}^{2n} \times \{0,1\}^{N-2n} \to \{0,1\}^{2n}$ and a hash function $H : \{0,1\}^{\leq L} \to \{0,1\}^{2n}$, where $N > 2n$* and $L$ is sufficiently large.

Designing a double length hash function from a single length compression or hash function is also important in the hardware point of view. As crypto-hardware are expensive, the construction which allows an existing hardware would be meaningful while we are looking for more security. There were several attempts to construct a secure block cipher based double length compression functions. Again, most of these have several collision and preimage attacks much better than the birthday attack [5, 6, 8, 10, 17]. Recently, Lucks [4, 10] designed a secure double length compression function. A similar designed is proposed by Hirose [6] by using a secure block ciphers of the form $E : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^n$. But the efficiency of their design is fairly low. A more efficient double

© Springer-Verlag Berlin Heidelberg 2005

length compression function based on three independent random compression functions has been proposed by Nandi *et. al.* [13]. But the security of the compression function is not maximum. So it would be interesting to design both maximally secure and efficient double length hash function.

### 1.1 Our Contribution

In this paper we design several new double length hash functions and compute their security level and the rate (measurement of efficiency).

Our first design is a generalization of Lucks's [10] and Hirose's [6] constructions. Given a permutations $p(\cdot)$ on the set of all $N$-bit strings and a compression function $f : \{0,1\}^N \to \{0,1\}^n$, define $f^p(X) = f(X) \parallel f(p(X))$. We show that the double length function $f^p$ is maximally secure provided the permutation $p$ does not have any fixed point.

Next, we study the security level for the double length hash function defined by the classical iteration of a compression function $f^p$ defined as above. We show that, along with secure compression functions there are many more compression functions which extends to a secure double length hash functions. Thus, we have a wide class of maximally secure double length hash functions. Lucks and Hirose's construction belong to this class.

Then we design a construction similar to the concatenated hash function. We show the collision security (or preimage security) level of the double length hash function $\Omega(2^n/in^{i-1})$ (or $\Omega(2^{2n}/in^{i-1})$ respectively), where $i$ is the number of iterations of underlying compression function to invoke a double length compression function. It determines the efficiency of the hash function.

## 2 Preliminaries

In this section we briefly recall some preliminaries of hash functions. We first give a brief introduction of classical hash functions. Then we explain random oracle model and the behavior of adversary in the random oracle model. Next, we explain Joux's attack and its application to the collision attack on concatenated hash function. Finally we state the recent constructions of double length hash functions and we compare their efficiencies.

### 2.1 The Classical Iterated Hash Function

We briefly explain a simplified version of Merkle-Damgård method [3, 11]. Let $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ be an underlying compression function. Given a message $M \in \{0,1\}^{\leq L} = \cup_{i=0}^{L}\{0,1\}^i$, where $L = 2^m - 1$ and an *initial value* $h_0 \in \{0,1\}^n$ we apply the following padding rule;

– *padding rule*: Choose smallest $i \geq 0$ such that $|M| + i + 1$ is multiple of $m$. Let $\langle |M| \rangle$ be the $m$-bit binary representation of $|M|$. We write $M \parallel 10^i \parallel \langle |M| \rangle = m_1 \parallel m_2 \cdots \parallel m_l$ for some positive integer $l$ and $|m_i| = m, 1 \leq i \leq l$.

Each $m_i$ is known as a *message block* of the underlying compression function. We define the *classical iterated hash function* $H^f(M)$, or simply $H(M)$, by the following method;

$$H(M) = h_l, \text{ where } h_i = f(h_{i-1} \parallel m_i),\ 1 \leq i \leq l.$$

the $h_i$'s are known as *intermediate hash values*, $i \neq 0, l$. For simplicity, we ignore the padding rule.

We use the notation $h \rightarrow_x h'$ (a labeled arc) to mean $f(h, x) = h'$, where $|h| = |h'| = n$ and $|x| = m$. Thus, the computation of $H(M)$ can be represented by a labeled path from $h_0$ to $h_l$ as follows;

$$h_0 \rightarrow_{m_1} h_1 \rightarrow_{m_2} h_2 \cdots h_{l-1} \rightarrow_{m_l} h_l \quad \text{or} \quad h_0 \Rightarrow_M h_l.$$

Thus, $h_0 \Rightarrow_M h_l$ if and only if $H(M) = h_l$.

## 2.2 The Random Oracle Model

Let $\mathcal{F}^{D \rightarrow R}$ be a set of all functions from $D$ to $R$. A randomly chosen function $f$ from $\mathcal{F}^{D \rightarrow R}$ is known as a *random function* [2]. One can define a random function in the following equivalent way and equivalence can be checked in a straightforward way;

**Definition 1. (Random Function)**
*A* random function*, $f$ from $D$ to $R$, takes values as random variables, such that for any $x \in D$, $f(x)$ has uniform distribution on $R$ and for any $k > 0$ and $k$ distinct elements $x_1, \cdots x_k \in D$, the random variables $f(x_1), \cdots, f(x_k)$ are independently distributed.*

**Proposition 1.** *Let $f : D \rightarrow R$ be a random function and $\{X_1, Y_1\} \neq \{X_2, Y_2\}$ be two subsets of $D$. Then* $\mathsf{Pr}\ [\ f(X_1) = f(Y_1)] = 1/|R|$ *and* $\mathsf{Pr}\ [\ f(X_1) = f(Y_1)$ *and* $f(X_2) = f(Y_2)] = 1/|R|^2$.

**Proof.** Since $X_1 \neq Y_1$, $f(X_1)$ is uniformly distributed on $R$ given $f(Y_1)$ and hence $\mathsf{Pr}\ [\ f(X_1) = f(Y_1)] = 1/|R|$.

For the second part of the proposition, without loss of generality, let us assume that $X_1 \notin \{X_2, Y_2\}$. Thus $f(X_1)$ is uniformly distributed on $R$ given the random variables $f(Y_1), f(X_2)$ and $f(Y_2)$. Thus, the conditional probability $\mathsf{Pr}$ $[f(X_1) = f(Y_1) \mid f(Y_1),\ f(X_2),\ f(Y_2)]$ is $1/|R|$ and hence so is unconditional. So we have,

$$\begin{aligned} & \mathsf{Pr}\ [f(X_1) = f(Y_1) \text{ and } f(X_2) = f(Y_2)\ ] \\ &= \mathsf{Pr}\ [f(X_1) = f(Y_1)\ \mid\ f(X_2) = f(Y_2)\ ] \times \mathsf{Pr}\ [\ f(X_2) = f(Y_2)\ ] \\ &= \mathsf{Pr}\ [\ f(X_1) = f(Y_1)\ ] \times \mathsf{Pr}\ [\ f(X_2) = f(Y_2)\ ] \\ &= 1/|R|^2. \end{aligned}$$ □

We call $f_1, \cdots, f_s$ *independent random functions* if $f_i$'s are chosen independently and randomly from the set $\mathcal{F}^{D \rightarrow R}$. We state an equivalent definition of independent random functions.

**Definition 2. (Independent Random Functions)**
*We say a family of random functions $f_1, \cdots, f_s : D \to R$ are* independent random functions *if for any $s$ subsets $\{x_1^1, \cdots, x_{k_1}^1\}$, $\cdots$ , $\{x_1^s, \cdots, x_{k_s}^s\}$, the random vectors $(f_1(x_1^1)$, $\cdots$, $f_1(x_{k_1}^1))$, $\cdots$, $(f_s(x_1^s)$, $\cdots$, $f_s(x_{k_s}^s))$ are independently distributed.*

**The Adversary in the Random Oracle Model :** When a hash function $H(\cdot)$ is designed based on a random compression function $f$, an attack algorithm is an oracle algorithm, $\mathcal{A}^f$, with the oracle $f$. Thus, adversary can ask several queries of $f$ adaptively and based on the query-response pairs adversary finally outputs a message or a pair of messages depending on the nature of the attack. It can choose $x_1, \cdots x_q$ adaptively and get responses $y_1, \cdots, y_q$, where $y_i = f(x_i)$. We can think that $y_i$ as a realization of the random variable $f(x_i)$ which is observed by the adversary. Define the complete list of query-response pairs $\mathcal{Q} = ((x_1, y_1), \cdots, (x_q, y_q))$ by *view* of the adversary. Any output produced by the adversary should only depend on the view. Moreover, if the adversary is finding collisions for a hash function, $H(\cdot)$, based on the compression function, $f(\cdot)$, and it outputs a pair of distinct messages $M \neq N$ then the values of $H(M)$ and $H(N)$ should be computed from the view. When we have two or more underlying compression functions $f_1, f_2, \cdots$, we have a set of lists of pairs $\{\mathcal{Q}_1, \mathcal{Q}_2, \cdots\}$ called view of the adversary, where $\mathcal{Q}_i$ is the view due to the random compression function $f_i$, $i \geq 1$.We define the complexity of an attack algorithm by size of the view to be required to have non negligible probability of success or advantage [1]. The minimum complexity of an attack algorithm is a measurement of security of the hash function.

## 2.3 The Birth-Day Attack

A set $\{M_1, \cdots, M_r\}$ is said to be an *r-way collision set* of $g : D \to R$, if $g(M_1) = \cdots = g(M_r)$. The above event is known as a *multicollision.*

**BirthdayAttack($g, q, r$) :**

1. Choose $x_1, \cdots, x_q$ randomly from the domain $D$ and compute $y_i = g(x_i)$ for $1 \leq i \leq q$.
2. Return a subset (if any) $C \subseteq \{x_1, \cdots, x_q\}$ of size $r$ such that $C$ is an $r$-way multicollision subset for the function $g$. Otherwise return the output "failure".

The next proposition gives an estimate of the complexity of the birthday attack in finding an $r$-way collision with significant probability. See [15, 16] for a detail discussion.

**Proposition 2. (Complexity of the Birthday Attack)** [14]
*For a random function $g : D \to \{0,1\}^n$, the birthday attack with complexity $q$ finds an $r$-way collision with probability $O(q^r/2^{(r-1)n})$. Thus, the birthday attack requires $\Omega(2^{n(r-1)/r})$ queries to find an $r$-way collision with significant probability. For $r = 2$, the birthday attack requires $O(2^{n/2})$ queries.*

## 2.4 Joux's Multicollision Attack

In a recent paper by Joux [7], it was shown that there is a $2^r$-way collision attack for the classical iterated hash function based on a compression function, $f : \{0,1\}^{m+n} \to \{0,1\}^n$, where the attack has complexity $\mathrm{O}(r\,2^{n/2})$. This complexity is much less than $\Omega(2^{\frac{n(2^r-1)}{2^r}})$, which is the complexity for the birthday attack (see Proposition 2).

Here is the basic idea of the attack. Consider a set of vertices $V = \{0,1\}^n$. We use the notation $h \to_M h'$ (a labeled arc) to mean $f(h, M) = h'$. Here, $|h| = |h'| = n$ and $|M| = m$. The strategy is to first find $r$ successive collisions (see Figure 1) by performing $r$ successive birthday attacks, as follows:

$$\begin{gathered}
f(h_0, m_1) = f(h_0, n_1) = h_1 \text{ (say), where } m_1 \neq n_1 \\
f(h_1, m_2) = f(h_1, n_2) = h_2 \text{ (say), where } m_2 \neq n_2 \\
\vdots \\
f(h_{r-1}, m_r) = f(h_{r-1}, n_r) = h_r \text{ (say), where } m_r \neq n_r.
\end{gathered}$$

For $1 \leq i \leq r$, we apply BirthdayAttack$(f(h_{i-1}, \cdot), 2^{n/2}, 2)$ to find $m_i \neq n_i$ such that $f(h_{i-1}, m_i) = f(h_{i-1}, n_i)$. Thus the set

$$\{x_1 \parallel \cdots \parallel x_r : x_i = m_i \text{ or } n_i, 1 \leq i \leq r\}$$

is a $2^r$-way collision set. The complexity of the attack is $\mathrm{O}(r\,2^{n/2})$. Figure 1 is a diagram illustrating the attack.

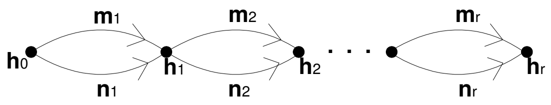


**Fig. 1.** Graphical representation of Joux's multicollision attack

**Applications of Multicollision Attacks :** A natural and efficient approach to produce large output hash values is the concatenation of several smaller output hash values. For example, given two classical iterated hash functions, $H$ and $G$, one can define a hash function $H(M) \parallel G(M)$. This idea has been frequently used because it is efficient and simple to implement. However, due to the attacks of Joux [7], there exists a collision attack that is more efficient than the birthday attack. The complexity of the attack is roughly the maximum of the complexity of the multicollision birthday attack on $H$ and the complexity of the standard birthday attack on $G$.

We briefly describe the attack (see [7] for more details). Let $H$ and $G$ have output hash values of $n_H$ and $n_G$ bits in length, respectively.

1. By using Joux's multicollision attack, find $2^{n_G/2}$ messages which have common output hash value (say $h^*$) on $H$.

2. Find two messages, say $M$ and $N$ where $M \neq N$, which are members of the set of $2^{n_G/2}$ messages found in step 1, such that they have same output hash value (say $g^*$) on $G$. Note that we expect to be able to find a collision on an $n_G$-bit function from a set of $2^{n_G/2}$ messages using the standard birthday attack.

Thus, we have $H(M) \parallel G(M) = H(N) \parallel G(N) = h^* \parallel g^*$. The overall complexity of this attack is $\mathrm{O}(n_G\, 2^{n_H/2} + 2^{n_G/2})$. Note that we only assume that $H$ is a classical iterated hash function; $G$ can be any hash function at all.

## 2.5 Rate Function (Efficiency Measurement) of Known Designs

We have underlying compression functions $f_1, f_2, \cdots f_k : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$. We design a double length compression function, $F : \{0,1\}^N \to \{0,1\}^{2n}$, based on $f_1, f_2, \cdots f_k$. We define a measurement of the efficiency of the compression function, $F(\cdot)$, called the rate function of $F$. Roughly, it measures the number of message blocks that are hashed per underlying compression function.

**Definition 3. (Rate Function)**
*Let a double length compression function, $F$, be based on $f_1, \cdots, f_k$. Define the rate function of $F$ by $\frac{N-2n}{m \times s}$, where $s$ is the number of invocations of all $f_i$'s are required to compute $F(X)$, $X \in \{0,1\}^N$.*

Since $F$ is a compression function, $N > 2n$. Thus, the rate function is always positive. When the rate function is constant, we only use the term *rate* instead of rate function. We define rate of the classical iterated hash function by the rate of the underlying compression function.

*Example 1.* The underlying double length compression function of the concatenated hash function $H^{f_1} \parallel H^{f_2}$ is $F(H_1, H_2, M) = f_1(H_1, M) \parallel f_2(H_2, M)$, where $|H_1| = |H_2| = n$ and $|M| = m$. The rate function of $F$ is 1/2.

*Example 2.* **(Nandi et. al. [13])** Let $f_i : \{0,1\}^{2n} \to \{0,1\}^n$ be three underlying compression functions, $1 \leq i \leq 3$. Define, $F : \{0,1\}^{3n} \to \{0,1\}^{2n}$, where $F(x,y,z) = (f_1(x,y) \oplus f_2(y,z)) \parallel (f_2(y,z) \oplus f_3(z,x))$ with $|x| = |y| = |z| = n$. The rate of this compression function is 1/3. The best collision attack on $F$ requires $\Omega(2^{2n/3})$ many queries of $f_i$'s in the random oracle model [13].

*Example 3.* **(Lucks [10])** Let $f : \{0,1\}^{n+m} \to \{0,1\}^n$ be an underlying compression function, $m > n$. Define $F(H_1, H_2, M) = f(H_1, H_2, M) \parallel f(H_2, H_1, M)$, $|H_1| = |H_2| = n$ and $|M| = m - n$. The rate function of the compression function is $\frac{n+m-2n}{2m} = \frac{1}{2} - \frac{n}{2m}$. When $m = 2n$, the rate of the compression function is $\frac{1}{4}$. The compression function is not secure. But we show later (also see [10]) that the classical hash function based on it is maximally secure in the random oracle model.

## 3 A Class of Double Length Compression Functions

Fix a compression function, $f : \{0,1\}^{n+m} \to \{0,1\}^n$, and define a class of double length compression functions

$$\mathcal{C} = \{f^p = f(\cdot) \parallel f(p(\cdot)) : p \text{ is a "simple" permutation on } \{0,1\}^{n+m}\}.$$

By a simple permutation we mean both the permutations, $p$ and $p^{-1}$, are easy to compute. One can also consider a compression function $f^{p_1,p_2}(X) = f(p_1(X)) \parallel f(p_2(X))$ where $|X| = n+m$ for two simple permutations $p_1$ and $p_2$. Since $p_1$ and $p_2$ are simple, it is sufficient to study the security properties of $f^p$ where $p = p_2 \circ p_1^{-1}$. All these compression function have rate $\frac{1}{2} - \frac{n}{2m}$ (as in Example 3).

### 3.1 Security Analysis of the Compression Functions from $\mathcal{C}$

In this section, we study the security properties of the compression functions from the class, $\mathcal{C}$, in the random oracle model of $f$. Let us first consider the Example 3. In this example, $F = f^p$, where $p(H_1, H_2, M) = H_2 \parallel H_1 \parallel M$, $|H_1| = |H_2| = n$ and $|M| = m - n$. By using the birthday attack, find $H, G$ and $M_1, M_2$, such that $(H, M_1) \neq (G, M_2)$ and $f(H, H, M_1) = f(G, G, M_2)$. Now, it is easy to check that $f^p(H, H, M_1) = f^p(G, G, M_2)$. Here, we need $O(2^{n/2})$ many queries.

The reason for having the above attacks is that the permutation $p$ has many "fixed points". $X$ is called a *fixed point* of a function $p(\cdot)$, if $p(X) = X$. We write $\mathcal{F}_p$ for the set of all fixed points of $p$. In the above example, $\mathcal{F}_p = \{H \parallel H \parallel M : |H| = n, |M| = m - n\}$ is the set of fixed points of the permutation $p$ and $|\mathcal{F}_p| > 2^n$. Thus, one can apply birthday attack to find a collision (or a preimage) on the compression function $f$ from the fixed point set. Similar attack can be done for any compression function based on a permutation with more that $2^n$ many fixed points. In the light of the above discussion, one should use a permutation, $p$, which does not have many fixed points. In fact, there are many permutations where the set of fixed points are the empty set. We give two classes of examples of that kind, in below.

*Example 4.* For $A \in \{0,1\}^N \setminus \{\mathbf{0}\}$, define a permutation $p : \{0,1\}^N \to \{0,1\}^N$ such that $p(X) = X \oplus A$. It is easy to check that $\mathcal{F}_p$ is empty.

*Example 5.* We can map any $N$-bit string to an integer modulo $2^N$. We use " $+$ " to denote addition modulo $2^N$. Let $p(X) = X + A$ where $A \neq 0$. Note that, $p(X) \neq X$ for all $X$. Moreover, if $A \neq 2^{N-1}$ then the fixed point of $p(p(X)) = X + 2A$ (in notation, $p^2$) is also empty.

Suppose, $f^p$ is a double length compression function based on a permutation, $p$, where $\mathcal{F}_p$ is the empty set. Then a collision, $f^p(X) = f^p(Y)$ with $X \neq Y$ implies $f(X) = f(Y)$ and $f(p(X)) = f(p(Y))$. Thus, $\{X, Y\}$ and $\{p(X), p(Y)\}$ are collision sets of $f$. Now, we have the following two cases.

- **Case-1 :** Two collision sets are identical $\{X, Y\} = \{p(X), p(Y)\}$. Since $p$ does not have any fixed point, we have $Y = p(X)$ and $X = p(Y)$. Thus, we should have a collision set $\{X, p(X)\}$, where $p(X) \neq X$ and $p(p(X)) = X$. Let $\Omega(K_1(n))$ (or in short $K_1$) be the complexity of the best attack to find a collision set of the form $\{X, p(X)\}$.
- **Case-2 :** $\{X, Y\} \neq \{p(X), p(Y)\}$. Let $\Omega(K_2(n))$ (or in short $K_2$) be the complexity of the best attack to find two distinct collision sets of the form $\{X, Y\}$ and $\{p(X), p(Y)\}$.

Thus a collision on $f^p$ reduces to the one of the above two events and hence the complexity of best collision attack is $\min\{K_1, K_2\}$. If $p^2$ does not have any fixed point then we can exclude the first case also and the complexity of the best collision attack is $K_2(n)$. We summarize the above discussion as follows;

**Proposition 3.** *The complexity of the best collision attack on $f^p$ is* min $\{\Omega(K_1(n)),\ \Omega(K_2(n))\}$ *where $p$ is a permutation with no fixed point and $K_1$ and $K_2$ are defined as above. Moreover, if the permutation $p^2$ does not have any fixed point (like in the Example 5) then the best collision attack on $f^p$ is $\Omega(K_2(n))$.*

Now we give some evidences why $K_1$ and $K_2$ would be large for a good compression function $f$. Suppose an adversary tries to find two collision sets $\{X, Y\} \neq \{p(X), p(Y)\}$. After finding a collision set $\{X, Y\}$, he does not have any freedom to choose for the second collision set and he is forced to check whether $\{p(X), p(Y)\}$ is a collision set or not. Thus $K_2$ would be large and may be close to $2^n$ for a good underlying compression function.

Next, an adversary tries to find a collision set $\{X, p(X)\}$. For each message $X$, $p(X)$ is completely determined (also vice-versa) and hence the adversary has to check equality of two values, $f(X)$ and $f(p(X))$, instead of comparing several values like in the birthday attack. Thus we expect $K_1$ to be large. In the random oracle model of $f$, we can prove that, $K_1(n) = K_2(n) = 2^n$. We first introduce a new notion called *computable message.*

**Definition 4. (Computable Message)**
*Let the double length compression function, $F$, be based on the compression functions, $f_1, \cdots, f_k$. Let $\mathcal{Q}_j = \{(x_1^j, y_1^j), \cdots, (x_{q_j}, y_{q_j})\}$ be the* view *of $f_j$, $1 \leq j \leq k$ and let $\mathcal{Q} = (\mathcal{Q}_1, \cdots, \mathcal{Q}_k)$. We say, an input $X$ is* computable message *of $F$ with respect to the view $\mathcal{Q}$, if the value of $F(X)$ can be determined from $\mathcal{Q}$.*

For example, when $F = f^p$, an input $X$ is computable message of $F$ with respect to $\{(x_1, y_1), \cdots, (x_q, y_q)\}$, view of $f$, if $X = x_i$ and $p(X) = x_j$ for some $i, j \in [1, q]$. Thus, $f^p(X) = f(x_i) \parallel f(x_j) = y_i \parallel y_j$, which can be computed from $\mathcal{Q}$.

**Theorem 1.** *Under the assumption of the random oracle model of $f$, $K_1(n) = K_2(n) = 2^n$. Thus, for any permutation $p$ where $\mathcal{F}_p$ is the empty set, any attack algorithm finding collisions requires $\Omega(2^n)$ many queries of $f$ in the random oracle model of $f$.*

**Proof.** If an adversary can ask at most $q$ many queries then he can have at most $q$ many computable messages and hence at most $\binom{q}{2}$ 2-sets $\{X, Y\}$. Hence the probability that the adversary finds $X \neq Y$ with $\{X, Y\} \neq \{p(X), p(Y)\}$ such that $f(X) = f(Y)$ and $f(p(X)) = f(p(Y))$ is at most $\binom{q}{2}/2^{2n}$ (by using Proposition 1). Thus, to have a significant success probability, $q$ should be $\Omega(2^n)$.

Similarly, from a set of $q$ queries one can get $\mathrm{O}(q)$ many pairs of the form $(X, p(X))$, where $X$ and $p(X)$ both are computable. For fixed $X$, $\mathsf{Pr}\ [f(X) = f(p(X))] = 1/2^n$ provided $p(X) \neq X$ (see Proposition 1). Thus success probability is at most $q/2^n$ and hence $q = \Omega(2^n)$ for significant success probability. □

*Remark 1.* Let $p$ be a permutation where $|\mathcal{F}_p| << 2^{n/2}$ such that there are no two elements $X \neq Y \in \mathcal{F}_p$ with $f(X) = f(Y)$. We can prove Theorem 1 if the permutation $p$ satisfies the above condition instead of the condition given in the Theorem.

## 4 A Class of Double Length Hash Functions

Now we study the double length hash functions defined by the classical iteration of the compression functions $f^p$ stated in Sect. 3.

**Definition 5.** *Let $p$ be a permutation on the set of $(n+m)$-bits strings, $m \geq n$. Define $\mathcal{F}_p[2n] = \{Z \in \{0,1\}^{2n} : \exists\ M \in \{0,1\}^{m-n}$ such that $Z \parallel M \in \mathcal{F}_p\ \}$. It is a projection of $\mathcal{F}_p$ onto the the first $2n$-bits of it. We say the permutation, $p(\cdot)$, is* good *if $|\mathcal{F}_p[2n]| = O(2^n)$. In particular, when $p$ does not have any fixed point it is also a good permutation.*

Now we define the following attack. Find $M$ and $H \notin \mathcal{F}_p[2n]$, such that $f^p(H, M) \in \mathcal{F}_p[2n]$, where $|M| = m - n$ and $|H| = 2n$. Let the complexity of the best attack be $\Omega(K_3(n))$ (or in short $K_3$).

**Proposition 4.** *The classical hash function, $H^{f^p}$, based on a good permutation and an initial value $H_0 \notin \mathcal{F}_p[2n]$ has collision security* $\min\{K_1, K_2, K_3\}$.

**Proof.** Let $(M, M')$ be a collision pair of $H^{f^p}$ and $H_0 \notin \mathcal{F}_p[2n]$. We denote $H_i$ and $G_i$ for internal hash values while computing the final hash value for messages $M = M_1 \parallel M_2 \cdots$ and $M' = M'_1 \parallel M'_2 \cdots$ respectively. One of the following holds:

1. There is an $i$ such that $H_i \notin \mathcal{F}_p[2n]$ but $f^p(H_i \parallel M_{i+1}) \in \mathcal{F}_p[2n]$ or there is a $j$ such that $G_j \notin \mathcal{F}_p[2n]$ but $f^p(G_j \parallel M'_{j+1}) \in \mathcal{F}_p[2n]$. To achieve this we need $\Omega(K_3)$ many queries.
2. There are $H_i, G_j \notin \mathcal{F}_p[2n]$ such that $X = (H_i, M_{i+1}) \neq (G_j, M') = Y$ and $f^p(X) = f^p(Y)$. Since $H_i, G_j \notin \mathcal{F}_p[2n]$, $p(X) \neq X$ and $p(Y) \neq Y$. Thus either $\{X, Y\} \neq \{p(X), p(Y)\}$ are collision two sets or $\{X, p(X)\}$ is a collision set for the compression function $f$. To achieve this adversary requires $\min\{K_1, K_2\}$ many queries of $f$.

Combining both the adversary needs $\min\{K_1, K_2, K_3\}$ queries. □

**Theorem 2.** *For a good permutation $p$ and random compression function $f$, $K_3(n) = 2^n$ and hence $H^{f^p}$ is maximally secure against collision attack.*

**Proof.** We have already seen that after $q$ many queries the adversary can have at most $q$ many computable messages for $f^p$. Given a computable message $H \parallel M$ with $H \notin \mathcal{F}_p[2n]$, we have $p(H \parallel M) \neq H \parallel M$ and hence $f^p(H \parallel M)$ is uniformly distributed over the set $\{0,1\}^{2n}$. But $|\mathcal{F}_p[2n]| < 2^n$ since the permutation $p(\cdot)$ is good. Thus we have, $\mathsf{Pr}\ [f^p(H \parallel M) \in \mathcal{F}_p[2n]] \leq 1/2^n$. Since we have at most $q$ computable message the success probability of the adversary is less than $q/2^n$. This proves the fact that $K_3(n) = 2^n$ under the random oracle model of $f(\cdot)$. By Theorem 1 and Proposition 4, $H^{f^p}$ is maximally secure under the random oracle model of $f$. □

## 5 An Efficient Double Length Hash Function

Let the compression function be $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. For $i > 1$, define $f^{(i)} : \{0,1\}^{(i+1)n} \to \{0,1\}^n$ by using the classical iteration. Thus, for $x_0 \parallel \cdots \parallel x_i$ with $|x_j| = n$, $0 \leq j \leq i$ and $h_0 = x_0$,

$$f^{(i)}(x_0 \parallel \cdots \parallel x_i) = h_i, \text{ where } h_j = f(h_{j-1}, x_j),\ 1 \leq j \leq i.$$

We call $f^{(i)}$ is the *$i$-iterated compression function.* Now we can observe that the multicollision on this compression function is not as easy as the classical hash function, since we restrict the number of message blocks. Any $r^i$-way collision on $f^{(i)}$ reduces to at least $r$-way collision on the underlying compression function $f$ (by using pigeon-hole principle). Thus, if we assume that $(r+1)$-way collision on $f$ is infeasible then we can have at most $r^i$-way collision on $f^{(i)}$. Recall that, in the random oracle model of $f$, $r$-way collision requires $\Omega(2^{n(r-1)/r})$ queries. Now we summarize this by the following lemma.

**Lemma 1.** *$(r^i+1)$-way collision on $f^{(i)}$ reduces to at least $(r+1)$-way collision on $f$. In particular, when $f$ is a random function, the complexity of $(r^i+1)$-way collision attack on $f^{(i)}$ is $\Omega(2^{nr/(r+1)})$ and the complexity of $(n^i+1)$-way collision attack on $f^{(i)}$ is $\Omega(2^n)$.*

Like the concatenation of two independent hash functions we can define the concatenation of two independent $i$-iterated compression functions. Thus, given two independent compression functions, $f_1$ and $f_2$, we can define a double length compression function, $F_i(X) = f_1^{(i)}(X) \parallel f_2^{(i)}(X)$, $|X| = n(i+1)$. Obviously, in this construction, we need to assume $i \geq 2$. Otherwise, for $i = 1$, it does not compress the input. Now we can study the security property of this concatenated compression function in the random oracle model.

**Lemma 2.** *If $f$ is a random function then for any two distinct $(i+1)$-block inputs $X$ and $Y$, $\mathsf{Pr}\,[f^{(i)}(X) = f^{(i)}(Y)] \leq i/2^n$. If $f_1$ and $f_2$ are two independent random functions then $\mathsf{Pr}\ [F_i(X) = F_i(Y)] = i^2/2^{2n}$.*

**Proof.** Let $j$ be the round number where collision of $f$ occurs. Call this event by $C_j$. Thus, $f^{(i)}(X) = f^{(i)}(Y)$ implies $\cup_{j=1}^{i} C_j$. Now, $\mathsf{Pr}\ [C_j] = \mathsf{Pr}\ [f(X_j) = f(Y_j)$ and $X_j \neq Y_j] = 1/2^n$, $X_j$ and $Y_j$ denote the input of $f$ at $j^{th}$ invocation for messages $X$ and $Y$ respectively. Thus, $\mathsf{Pr}\ [\cup_{j=1}^{i} C_j] \leq i/2^n$.

$$\begin{aligned}\mathsf{Pr}\ [F_i(X) = F_i(Y)] &= \mathsf{Pr}\ [f_1^{(i)}(X) = f_1^{(i)}(Y), f_2^{(i)}(X) = f_2^{(i)}(Y)] \\ &= \mathsf{Pr}\ [f_1^{(i)}(X) = f_1^{(i)}(Y)] \times \mathsf{Pr}\ [f_2^{(i)}(X) = f_2^{(i)}(Y)] \\ &\leq i^2/2^{2n}.\end{aligned}$$

The second equality follows from the fact that $f_1$ and $f_2$ are independent random functions and the last inequality is immediate from the first half of the Lemma. □

Thus to find the collision probability for any adversary we need to compute the number of pairs $(X, Y)$ it can get from any possible set of queries. Note that the adversary should compute the $F$-values of both $X$ and $Y$. Now we state the computable message which means the message whose hash value can be computed from the set of queries the adversary made. We fix $i \geq 2$.

**Definition 6. (Computable message)** *Let $\mathcal{Q}_j$ be the set of query response tuples for the random function $f_j$, $j = 1, 2$. $X$ is said to be a computable message for $f_j^{(i)}$ (also for $F_i$) with respect to $\mathcal{Q}_j$ if the value of $f_j^{(i)}(X)$ ( or $F_i(X)$) can be computed from $\mathcal{Q}_j$ (or $\mathcal{Q}_1 \cup \mathcal{Q}_2$ respectively).*

More precisely, if $X = x_0 \parallel \cdots \parallel x_i$ then $X$ is computable for $f_1^{(i)}$ with respect to $\mathcal{Q}_1$ if $(x_0 \parallel x_1, h_1)$, $(h_1 \parallel x_2, h_2)$,$\cdots$,$(h_{i-1} \parallel x_i, h_i) \in \mathcal{Q}_1$. Thus the $f_1^{(i)}$-value of $X$ is $h_i$. Similarly one can define computable messages for $f_2^{(i)}$. A message $X$ is computable with respect to $\mathcal{Q}_1 \cup \mathcal{Q}_2$ for the compression function $F_i$, if $X$ is computable for both $f^{(i)}$ with respect to $\mathcal{Q}_j$, $j = 1, 2$.

Let $q$ be the number of queries. We assume that $q = \mathrm{o}(2^n)$. Thus there is no $n$-way collision on both $f_1$ and $f_2$. Note that, the complexity of $n$-way collision on a random function is $\Omega(2^{n(n-1)/n}) = \Omega(2^n)$. Thus we can have at most $n^{i-1}$-way collision on $f_1^{(i-1)}$ or $f_2^{(i-1)}$. The number of computable messages for $F_i$ is at most $qn^{i-1}$. Thus, the total number of pairs of the form $(X, Y)$ where $X \neq Y$ are $(i+1)$-block inputs and both $X$ and $Y$ are computable messages is at most $q^2 n^{2(i-1)}/2$. Thus, the probability that we have a collision among these pairs is bounded by $i^2 q^2 n^{2(i-1)}/2^{2n+1}$. To have non-negligible probability we need $q = \Omega(2^n/in^{i-1})$. Thus we have the following theorem :

**Theorem 3.** *If $f_1$ and $f_2$ are two independent random functions then the complexity for finding a collision on $F_i$ requires $\Omega(2^n/(in^{i-1}))$ queries.*

**Efficiency of the compression function.** The rate function of the compression function, $F_i$, is $((i+1)n - 2n)/2ni = \frac{1}{2} - \frac{1}{2i}$. Thus, the rate of the compression function is close to $1/2$ provided $i$ is large. So we have a trade-off between the security level and the efficiency.

For $s \geq 2$, define a double length hash function $H : (\{0,1\}^n)^* \to \{0,1\}^{2n}$. We can define the hash function on arbitrary domain by applying some standard padding rule. Let $M = m_1 \parallel \cdots \parallel m_l$ be $l$-block message, $|m_i| = n$ for each $i$. Let $l = (s-1)b+r$, where $0 \leq r < s-1$. Thus, we divide the message $M = M_1 \parallel \cdots \parallel M_b \parallel M_{b+1}$, where $|M_i| = (s-1)n$, $1 \leq i \leq b$ and $|M_{b+1}| = rn$. In case of $r = 0$ we do not have any message block $M_{b+1}$. Let $H_0$ be an initial two block message that is $|H_0| = 2n$. Now define the hash function $H(H_0, M) = F_s^{(b)}(M_1 \parallel \cdots \parallel M_{b+1})$ if $r = 0$, otherwise $H(H_0, M) = F_r(F_s^{(b)}(M_1 \parallel \cdots \parallel M_{b+1}) \parallel M_{b+1})$.

Thus, the hash function is the classical iterated hash function using two underlying compression functions $F_s$ and $F_{r+1}$. Thus any collision on $H$ reduces to a collision on one of the compression functions. Thus we have the following theorem;

**Theorem 4.** *For any $s \geq 2$, collision on $H^{(s)}$ requires $\Omega(2^n/s^2 n^{s-1})$ complexity.*

## 6 (2nd) Preimage Security Analysis

Similar to the previous section we can study the (2nd) preimage security. Recall that we say a message $X$ is computable from the set of queries $\mathcal{Q}$ if $f^{(i)}(X)$ can be computed from the set $\mathcal{Q}$. We have already observed that if $q$ is the maximum number of queries and at most $r$-way collision is possible then we can have $qr^{i-1}$ computable messages. Now given $M$, $F_i(M)$ is a $2n$-bit random string. We also have observed that $\Pr[F(M) = F(N)] = i^2/2^{2n}$, where $M \neq N$. So, if $q = o(2^n)$ then the number of computable messages for $N$ is at most $n^{i-1}q$. Thus, there will be a computable message $N \neq M$ such that $F_i(M) = F_i(N)$ is bounded by $qn^{i-1}/i^2 2^{2n}$. Thus complexity for any attack algorithm of 2nd preimage attack is $\Omega(2^{2n}/i^2 n^{i-1})$.

Note that, in preimage or collision attack on $F_i$, we do not count the complexity for multicollision attack. In fact, it is likely that if the number of computable messages is $qr^{i-1}$ then the number of queries is at least $q2^{n(r-1)/r}$. Thus one can try to prove the following statement in the random oracle model;

*The complexity of the best collision (or preimage) attack on the above double length compression function $F_i$ is $\Omega(2^n/i)$ (or $\Omega(2^{2n}/i)$ respectively).*

## 7 Conclusion

We have studied several new double length compression functions. We first introduced a class of double length compression function which contains recently known constructions [6, 10]. We studied their security levels in the random oracle model. We also designed a double length compression function $F_i$ of rate close to 1/2 (the rate of concatenated hash function). The design is very much similar to the concatenated hash functions. It has almost maximal security level. In fact, we believe that the complexity of the best collision attack on the above double length compression function $F_i$ is $\Omega(2^n/i)$. It would be interesting to prove our belief. A possible future research would be to design efficient as well as secure double length hash functions.

## References

1. M. Bellare. *A Note on Negligible Function.* Journal of Cryptology, Springer-Verlag, vol **15**, No. 4, pp. 271-284, September 2002.
2. R. Canetti, O. Goldreich and S. Halvei. *The random oracle methodology, revisited.* $30^{th}$ Annual ACM Symposium on Theory of Computing (STOC), pp. 209-218, 1998.
3. I. B. Damgård. *A Design Principle for Hash Functions.* Advances in Cryptology - Crypto'89, Lecture Notes in Computer Sciences, vol **435**, Springer-Verlag, pp. 416-427, 1989.
4. H. Finney. *More problems with hash functions.* The cryptographic mailing list, 24 Aug 2004. Available at http://lists.virus.org/cryptography-0408/msg00124.html.
5. M. Hattori, S. Hirose and S. Yoshida. *Analysis of Double Block Lengh Hash Functions.* 9th IMA International Conference Cryptographi and Coding, 2003, Lecture Notes in Computer Science, vol **2898**, 2003.
6. S. Hirose. *Provably Secure Double-Block-Length Hash Functions in a Black-Box Model*, 7th International Conference on Information Security and Cryptology, 2004.
7. A. Joux. *Multicollision on Iterated Hash Functions. Applications to Cascaded Constructions.* Advances in Cryptology - Crypto'04, Lecture Notes in Computer Science, vol **3152**, 2004.
8. L. Knudsen, X. Lai and B. Preneel. *Attacks on fast double block length hash functions.* Journal of Cryptology, vol **11**, no-1, winter, 1998.
9. L. Knudsen and B. Preneel. *Construction of Secure and Fast Hash Functions Using Nonbinary Error-Correcting Codes.* IEEE transactions on information theory, vol **48**, No. 9, Sept-2002.
10. S. Lucks. *Design principles for Iterated Hash Functions.* ePrint Archive Report, 2004. Available at http://eprint.iacr.org/2004/253.
11. R. Merkle. *One Way Hash Functions and DES.* Advances in Cryptology - Crypto'89, Lecture Notes in Computer Sciences, Vol. **435**, Springer-Verlag, pp. 428-446, 1989.
12. C. H. Meyer and M. Schilling. *Secure program load with manipulation detection code.* Proceedings Securicom, pp. 111-130, 1988.
13. M. Nandi, W. Lee, K. Sakurai and S. Lee. *Security Analysis of a* 2/3*-rate Double Length Compression Function in The Black-Box Model.* Fast Software Encryption'05, 2005.
14. M. Nandi and D. R. Stinson. *Multicollision Attacks on Generalized Hash Functions.* Cryptology ePrint Archive, 2004. Available at http://eprint.iacr.org/2004/330.
15. D. R. Stinson. *Cryptography : Theory and Practice*, Second Edition, CRC Press, Inc.
16. D. R. Stinson. *Some observations on the theory of cryptographic hash functions.* ePrint Archive Report, 2001. Available at http://eprint.iacr.org/2001/020/.
17. T. Satoh, M. Haga and K. Kurosawa. *Towards Secure and Fast Hash Functions.* IEICE Trans. vol **E82-A**, No. 1 January, 1999.

# Near Optimal Algorithms for Solving Differential Equations of Addition with Batch Queries[⋆]

Souradyuti Paul and Bart Preneel

Katholieke Universiteit Leuven,
Dept. ESAT/COSIC, Kasteelpark Arenberg,
10, B–3001, Leuven-Heverlee, Belgium
{Souradyuti.Paul, Bart.Preneel}@esat.kuleuven.ac.be

**Abstract.** Combination of *modular addition* (+) and *exclusive-or* ($\oplus$) is one of the widely used symmetric cipher components. The paper investigates the strength of *modular addition* against *differential cryptanalysis* (DC) where the differences of inputs and outputs are expressed as *XOR*. In particular, we solve two very frequently used equations (1) $(x + y) \oplus (x + (y \oplus \beta)) = \gamma$ and (2) $(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma$, known as the *differential equations of addition* (DEA), with a set of *batch* queries. In a companion paper, presented at ACISP'05, we improved the algorithm by Muller (at FSE'04) to design optimal algorithms to solve the equations with *adaptive* queries. However, a *nontrivial* solution with *batch* queries has remained open. The major contributions of this paper are (*i*) determination of lower bounds on the required number of *batch* queries to solve the equations and (*ii*) design of two algorithms which solve them with queries close to optimal. Our algorithms require $2^{n-2}$ and 6 queries to solve (1) and (2) where the lower bounds are $\frac{3}{4}\cdot 2^{n-2}$ (theoretically proved) and 4 (based on extensive experiments) respectively ($n$ is the bit-length of $x, y, \alpha, \beta, \gamma$). This exponential lower bound is an important theoretical benchmark which certifies (1) as *strong* against DC. On the other hand, the constant number of batch queries to solve (2) discovers a major weakness of *modular addition* against DC.

Muller, at FSE'04, showed a key recovery attack on the Helix stream cipher (presented at FSE'03) with $2^{12}$ *adaptive chosen plaintexts* (ACP). At ACISP 2005, we improved the data complexity of the attack to $2^{10.41}$. However, the complexity of the attack with *chosen plaintexts* (CP) was unknown. Using our results we recover the secret key of the Helix cipher with only $2^{35.64}$ *chosen plaintexts* (CP) which has so far been the only CP attack on this cipher (under the same assumption as that of Muller's attack). Considering the abundant use of this component, the results seem useful to evaluate the security of many block ciphers against DC.

**Keywords:** Differential Cryptanalysis, Addition, Lower bound, Binary Tree.

⋆ This work was supported in part by the Concerted Research Action (GOA) Mefisto 2000/06 and Ambiorix 2005/11 of the Flemish Government and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

© Springer-Verlag Berlin Heidelberg 2005

## 1 Introduction

**Addition and XOR.** Mixing two different group operations is a common technique adopted by the designers of symmetric ciphers to make cryptanalysis difficult. The main reason behind the wide use of mixing *modular addition* (or simply *addition*) with XOR is their high speed on modern machines and their non-linearity over GF(2). Helix [6], IDEA [9], Mars [3], RC6 [14], Twofish [15] and the MD-family of hash functions are a few applications of *addition* and XOR. Plenty of research has also been spent on analyzing behaviors of addition, XOR and their mixing. Staffelbach and Meier worked on determination of the probability distribution of the carry for integer addition [16]. Wallén investigated the linear approximations of modular addition [18]. Lipmaa *et al.* dealt with the equation $(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma$ and its dual to investigate the differential properties [10, 11]. Paul and Preneel showed that the satisfiability of an arbitrary set *differential equations of addition* is in the complexity class P and designed algorithms to solve them efficiently [13].

**Batch and Adaptive Queries.** Cryptologic analogue of a query is a plaintext (or a ciphertext depending on the mode of attack). Similarly, attacks based on *batch* and *adaptive* queries are equivalent to *chosen plaintext* (CP) attack and *adaptive chosen plaintext* (ACP) attack respectively. A CP attack is more practical than the corresponding ACP attack because, in the latter case, we assume the attacker to be powerful enough to submit queries adaptively, i.e., the next query is computed based on the answers to the previous queries. In other words, an ACP attack requires two computing oracles while a CP attack needs only one. There is a large number of research papers launching CP and ACP attacks on many practical ciphers. For example, the boomerang attack introduced by Wagner is an ACP attack [17] and, therefore, less practical. The slide attacks by Biryukov and Wagner can be implemented using both CP and ACP but with different amount of data and time [5]. The time-memory trade-off attack by Hellman [7] and the differential attack on DES by Biham and Shamir [4] are CP attacks; so they are attributed more practical importance. In this paper we will deal with a very frequently encountered set of equations in secret key cryptography, known as *differential equations of addition* (explained in the next paragraph), to solve them with a set of *batch* queries (equivalent to a CP attack).

**Summary of the Results.** *Addition mod* $2^n$ is extensively used as a block cipher component. *Differential Cryptanalysis* (DC) is one of the most powerful attacks against block ciphers [4]. In this paper we investigate the security of *addition* under DC. In particular, we deal with the following two equations where differences of inputs and outputs of *addition* are expressed as *exclusive-or*

$$(x + y) \oplus (x + (y \oplus \beta)) = \gamma\,, \tag{1}$$

$$(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma\,. \tag{2}$$

These equations are known as *differential equations of addition* (DEA). In course of cryptanalysis of MD5 in 1992, Berson observed the *hardness* to analyze *modular addition* for large $n$ when differences are expressed as XOR [2]. The apparent difficulty to analyze combination of *addition* and XOR seems to justify its widespread application in symmetric cryptography. The solution to the above equations with *adaptive* queries (an $(\alpha, \beta)$ is a query) was first given in [12]. In the companion paper [13], we provided optimal algorithms which required $(n-1)$ and 3 *adaptive queries* in the worst case to solve (1) and (2) respectively. However, a *nontrivial* solution with a set of *batch* queries has remained elusive (the problem is elaborated in Section 3). As argued before, solution with *adaptive queries* is less practical because of the assumption of more powerful adversary which uses two oracles. This paper solves the equations with *batch* queries where only one oracle is needed. Our algorithm solves (1) with $2^{n-2}$ queries where a lower bound on the number of queries is $\frac{3}{4} \cdot 2^{n-2}$ (for all $n > 3$), i.e., the lower bound is optimal up to a constant factor of $\frac{4}{3}$. This exponential lower bound constitutes an important theoretical reference point which endorses the equations's strong resistance against DC. On the other hand, (2) has been solved with only 6 (for all $n > 2$) queries which is two more than a conjectured lower bound (note that the total number of queries is $2^{2n}$) – this fact shows a major cryptographic weakness of *addition* under DC and therefore, this component should be used with caution. In practical cryptanalysis, using these results we are successful to recover the secret key of a recently proposed stream cipher Helix [6], which was a candidate for consideration in the 802.11i standard, with $2^{35.64}$ chosen plaintexts which has so far been the only CP attack on this cipher (the earlier attacks were ACP attacks with data complexities $2^{12}$ and $2^{10.41}$ [12, 13]). In view of plenty of applications of *addition* and XOR, our analyses seem suitable to evaluate cryptographic strength of many ciphers. In addition, solutions to the above equations emerge as typical tasks in many branches of mathematics and computers science such as combinatorics, Boolean algebra, computational complexity. For example, the results may be useful to solve equations involving *modular multiplication* and $T$-functions [8]. Last but not the least, the technique used to derive all the results of this article, is purely combinatorial and easier than the traditional algebraic methods that use Gröbner bases to solve multivariate polynomial equations [1].

## 2 Notation

The $i$th bit of an $n$-bit integer $l$ is denoted by $l_i$ ($l_0$ denotes the least significant bit or the 0th bit of $l$). The operation *addition modulo* $2^n$ over $\mathbb{Z}_{2^n}$ can be viewed as a binary operation over $\mathbb{Z}_2^n$ (we denote this operation by '+') using the bijection that maps $(l_{n-1}, \cdots, l_0) \in \mathbb{Z}_2^n$ to $l_{n-1}2^{n-1} + \cdots + l_0 2^0 \in \mathbb{Z}_{2^n}$. The symbols '$\oplus$' and '$\wedge$' denote the operations *bit-wise exclusive-or* and *bit-wise and* of two $n$-bit integers respectively. We will denote $a \wedge b$ by $ab$. Throughout the paper, $[p, q]$ denotes a set containing all integers between the integers $p$ and $q$ including both of them. Unless otherwise stated, $n$ denotes a positive integer. The size of a set $S$ is denoted by $|S|$.

## 3 The Problem: Solving DEA with Batch Queries

In Sect. 1, we have already provided the motivation for solving the following two *differential equations of addition* (DEA) over $\mathbb{Z}_2^n$,

$$(x+y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma, \tag{3}$$

$$(x+y) \oplus (x + (y \oplus \beta)) = \gamma\,, \tag{4}$$

where $x$, $y$ are the *only fixed* unknown variables. The bit-length of each of $x, y, \alpha, \beta, \gamma$ is $n$. In the present context, solving (3) should be understood to be solving the set of all $2^{2n}$ equations generated by ranging $(\alpha,\, \beta)$ with the corresponding $\gamma$ for a fixed unknown $(x,\, y)$ (for (4) the number of all equations is $2^n$). Such an $(\alpha,\, \beta)$ is known as a *query.* As pointed out in [13] also, the number of solutions satisfying all $2^{2n}$ equations is no more than that of any subset of the equations and therefore, solving these $2^{2n}$ equations reduces the search space of the secret $(x,\, y)$ to the minimum. The most captivating question that follows immediately is that whether it is possible to solve a subset of the $2^{2n}$ equations and obtain the same solution as that of the entire $2^{2n}$ equations. In cryptographic applications, minimizing the number of equations to solve DEA's can be translated into reduction of the data complexities of many attacks [12], [13]. Next, we formalize the problem for easy understanding of many results. In the companion paper [13], we built an adversarial model that worked with *adaptive queries.* In the subsequent sections, we adapt the previous formalism to an attack model which uses *batch queries.*

### 3.1 The Power of the Adversary

The power of the adversary is described below.

1. The adversary has unrestricted computational power and an infinite amount of memory.
2. The adversary submits a *set* of queries $\{(\alpha,\, \beta)\}$ in a *batch,* to an honest oracle[1] which computes the $\gamma$'s using the fixed unknown $(x,\, y)$ in (3) and returns them to the adversary. The fixed $(x,\, y)$ is defined to be the *seed* of the oracle.

An oracle with seed $(x,\, y)$ is viewed as a mapping $O_{xy} : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$ and defined by

$$O_{xy} = \{(\alpha,\, \beta,\, \gamma) \,|\, (\alpha,\, \beta) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n,\, \gamma = (x+y) \oplus ((x \oplus \alpha) + (y \oplus \beta))\}\,. \tag{5}$$

The above adversarial model for (3), can be tailored for (4) by setting $(\alpha,\, \beta) \in \{0\}^n \times \mathbb{Z}_2^n$ and the mapping $O_{xy} : \{0\}^n \times \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$ .

The model is suitable for a *chosen message attack* where the adversary submits queries to an oracle in batch and based on the replies she computes unknown values.

[1] An honest oracle correctly determines $\gamma$.

## 3.2 The Useful Set $\tilde{D}$ and the Solution Set $\tilde{D}$-Consistent

$O_{xy}$, defined in (5), generates a family of mappings $\mathcal{F} = \{O_{xy} \mid (x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n\}$. If $D \in \mathcal{F}$ then $D$ is called a *character set*. Note that $|D| = 2^{2n}$ and therefore, a *character set* uniquely represents a set of $2^{2n}$ equations if we deal with (3). The aim of the adversary is to find all $(x, y)$'s satisfying these $2^{2n}$ equations from a subset of the *character set* $D$. The set of all such satisfiable $(x, y)$'s is called $D$-satisfiable. If we work with (4) then $|D| = 2^n$.

**Equivalent Task.** Applying the following transformation on a *character set* $D$ we compute $\tilde{D}$,

$$\tilde{D} = \{(\alpha, \beta, \tilde{\gamma} = \alpha \oplus \beta \oplus \gamma) \mid (\alpha, \beta, \gamma) \in D\}.$$

We call $\tilde{D}$ a *useful set*. Note $|\tilde{D}| = 2^{2n}$ if (3) is considered. An element $(\alpha, \beta, \tilde{\gamma}) \in \tilde{D}$ corresponds to the following equation

$$(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) \oplus \alpha \oplus \beta = \tilde{\gamma}.$$

Let $\tilde{D}$-consistent denote the set of all $(x, y)$'s satisfying all $2^{2n}$ equations corresponding to $\tilde{D}$. It can be shown that

$$D\text{-satisfiable} = \tilde{D}\text{-consistent}.$$

Therefore, the task is equivalent to determination of $\tilde{D}$-consistent from a subset of the *useful set* $\tilde{D}$. Note that there is a bijection between $D$ and $\tilde{D}$. If we deal with (4) then $|\tilde{D}| = 2^n$.

**Equivalent Oracle Output.** The oracle output $\gamma$ on query $(\alpha, \beta)$ will be adjusted to $\tilde{\gamma} = \alpha \oplus \beta \oplus \gamma$ for easy understanding of many deductions.

**Rules of the Game.** Below, we describe the rules followed by the adversary who determines the set $\tilde{D}$-consistent. The essence of the whole problem is brought out in the following points.

1. The adversary starts with no information about $x$ and $y$ except their *bit-length n*.
2. The adversary submits a set of queries $(\alpha, \beta)$'s in a batch, irrespective of the seed $(x, y)$. The oracle returns to the adversary the set of $\tilde{\gamma}$'s corresponding to the queries and the chosen $(x, y)$.
3. The adversary fails if, with the submitted queries, she is unable to compute $\tilde{D}$-consistent for some $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$.

*We search for an algorithm that determines $\tilde{D}$-consistent, for all $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$, with the same set of submitted queries. Furthermore, there is an additional challenge to reduce the number of required queries as much close as possible to the minimum.*

## 4 Previous Work: The Number of Solutions

Before embarking on designing algorithms to solve the equations, we first establish the number of all solutions for different seeds $(x,\, y)$'s, i.e., the size of $\tilde{D}$-consistent (see Sect. 3.2 for the definition). We follow a series of steps (Sect. 4.1 and 4.2) leading up to the formulation of $\tilde{D}$-consistent in Sect. 4.3. We shall heavily use the results of this section to obtain two important contributions of the paper: *(i)* lower bounds on the number of queries (described in Sect. 5) and *(ii)* the proofs of correctness of our algorithms which computes $\tilde{D}$-consistent (explained in Sect. 6). Due to lack of space the results of this section are provided without proofs and examples. See the companion paper [13] for them.

### 4.1 Step 1: Relation Among Input Bits

Let $A \subseteq \tilde{D}$ where $\tilde{D}$ is a *useful set*. Take an arbitrary element $(\alpha,\, \beta,\, \tilde{\gamma}) \in A$ $(n > 1)$. Observe that $\tilde{\gamma}_{i+1}$ can be computed using only the *preceding* bits $x_i$, $y_i$, $c_i$, $\alpha_i$, $\beta_i, \tilde{\gamma}_i$, $\forall\, i \in [0,\, n-2]$, from the following three equations

$$\tilde{\gamma}_{i+1} = c_{i+1} \oplus \tilde{c}_{i+1},\; c_{i+1} = x_i y_i \oplus x_i c_i \oplus y_i c_i,\; \tilde{c}_{i+1} = \tilde{x}_i \tilde{y}_i \oplus \tilde{x}_i \tilde{c}_i \oplus \tilde{y}_i \tilde{c}_i$$

where $c_i$ is the carry at the $i$th position of $(x + y)$, $\tilde{x}_i = x_i \oplus \alpha_i$, $\tilde{y}_i = y_i \oplus \beta_i$ and $\tilde{c}_i = c_i \oplus \tilde{\gamma}_i$. Table 1 lists the values of $\tilde{\gamma}_{i+1}$ as computed from all values of $x_i$, $y_i$, $c_i$, $\alpha_i$, $\beta_i$, $\tilde{\gamma}_i$.

**Table 1.** The values of $\tilde{\gamma}_{i+1}$ corresponding to the values of $x_i$, $y_i$, $c_i$, $\alpha_i$, $\beta_i$, $\tilde{\gamma}_i$. A row and a column are denoted by R($l$) and Col($k$)

| $(x_i, y_i, c_i)$ | $(\alpha_i,\, \beta_i, \tilde{\gamma}_i)$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | (0,0,0) | (0,0,1) | (0,1,0) | (0,1,1) | (1,0,0) | (1,0,1) | (1,1,0) | (1,1,1) | R(0) |
| (0,0,0) | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | R(1) |
| (1,1,1) | | | | | | | | | |
| (0,0,1) | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | R(2) |
| (1,1,0) | | | | | | | | | |
| (0,1,0) | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | R(3) |
| (1,0,1) | | | | | | | | | |
| (1,0,0) | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | R(4) |
| (0,1,1) | | | | | | | | | |
| Col(0) | Col(1) | Col(2) | Col(3) | Col(4) | Col(5) | Col(6) | Col(7) | Col(8) | |

### 4.2 Step 2 : Computation of Parameters $G_i$, $S_{i,\,0}$ and $S_{i,\,1}$ from $A$

We now determine an important quantity, denoted by $G_i$, for nonempty $A \subseteq \tilde{D}$. In $G_i$, we store the $i$th and $(i+1)$th bits of $\tilde{\gamma}$ and the $i$th bit of $\alpha$ and $\beta$ for all $(\alpha,\, \beta,\, \tilde{\gamma}) \in A$. We call $G_i$ the $i$th core of $A$. More formally (suppose $n > 1$),

$$G_i = \{(\alpha_i,\, \beta_i,\, \tilde{\gamma}_i,\, \tilde{\gamma}_{i+1}) \,|\, (\alpha,\, \beta,\, \tilde{\gamma}) \in A\}, \quad i \in [0,\, n-2]\,. \tag{6}$$

**Meaning of the expression "$G_i \Rightarrow (x_i, y_i, c_i)$" for a known $G_i$.** Let $|G_i| = g$. Take an element $(\alpha_i, \beta_i, \tilde{\gamma}_i, \tilde{\gamma}_{i+1}) \in G_i$. In Table 1, find the row(s) of the fourth coordinate $\tilde{\gamma}_{i+1}$ in the column specified by the first three coordinates $(\alpha_i, \beta_i, \tilde{\gamma}_i)$ in R(0) and put them in set $F_{i1}$. Find $F_{i1}, \cdots F_{ig}$ for all $g$ elements of $G_i$. Let $F_i = \bigcap_j F_{ij}$ and R(x)$\in F_i$. If $(x_i, y_i, c_i)$ is in Col(0)×R(x) then we say $G_i \Rightarrow (x_i, y_i, c_i)$. If $F_i = \phi$ then no such $(x_i, y_i, c_i)$ exists.

**How to compute $S_{i,j}$ using $G_i$.** $S_{i,j} = \{(x_i, y_i) \,|\, G_i \Rightarrow (x_i, y_i, c_i = j)\}$. Now, we show a fundamental relation between $S_{i,0}$ and $S_{i,1}$ that will be used to obtain several results.

**Proposition 1.** *For all nonempty set $A \subseteq \tilde{D}$ and all $n > 1$, $|S_{i,0}| = |S_{i,1}|$ $\forall\, i \in [0, n-2]$.*

We set,

$$|S_{i,0}| = |S_{i,1}| = S_i \quad \forall\, i \in [0, n-2]. \tag{7}$$

## 4.3 Step 3 (final): Formulation of the Size of $A$-Consistent

Let $A \subseteq \tilde{D}$. The definition of $A$-consistent which denotes the set of all $(x, y)$'s satisfying the equations represented by $A$ is a natural extension of the definition of $\tilde{D}$-consistent. The general formula for the number of solutions for any given set of DEA (the set may not contain all possible equations) is shown in the following proposition.

**Proposition 2.** *Let $A \neq \phi$ and $S$ denote $|A$-consistent$|$. Then,*

$$S = \begin{cases} 0 & \text{if } \tilde{\gamma}_0 = 1 \text{ for some } (\alpha, \beta, \tilde{\gamma}) \in A, \\ 4 \cdot \prod_{i=0}^{n-2} S_i & \text{if } \tilde{\gamma}_0 = 0, \forall (\alpha, \beta, \tilde{\gamma}) \in A \text{ and } n > 1, \\ 4 & \text{if } \tilde{\gamma}_0 = 0, \forall (\alpha, \beta, \tilde{\gamma}) \in A \text{ and } n = 1. \end{cases}$$

*The $S_i$'s are defined in (7).*

$|\tilde{D}$-consistent$|$ for (4) and (3) are obtained in Theorem 1 and Theorem 2 as special cases of Proposition 2 where $\tilde{D} = A$.

**Theorem 1.** *Let the position of the least significant '1' of $x$ in the equation*

$$(x + y) \oplus (x + (y \oplus \beta)) = \gamma$$

*be $t$ and $x, y, \beta, \gamma \in \mathbb{Z}_2^n$. Let a* useful set *$\tilde{D}$ be given. Then $|\tilde{D}$-consistent$|$ is*
*(i) $2^{t+3}$ if $n - 2 \geq t \geq 0$,*
*(ii) $2^{n+1}$ otherwise (including the case when $x = 0$).*

**Theorem 2.** *Let a* useful set *$\tilde{D}$ be given for the equation*

$$(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma$$

*with $x, y, \alpha, \beta, \gamma \in \mathbb{Z}_2^n$. Then $|\tilde{D}$-consistent$|$=4.*

## 5 Lower Bounds on the Number of Queries

Armed with the results derived in the previous section, we are now ready to establish one of the crucial results of this article, i.e., lower bounds on the number of queries, submitted in a batch, to solve the said equations. As it is already clear from the previous discussion that the trivial method is to submit all possible queries and then solve it; the challenge lies with a *nontrivial* solution which uses less number of queries. It is always regarded as an important theoretical benchmark as to how far it is possible to reduce the number of queries. The significance of a lower bound is that no algorithm can solve the equations with queries less than it.

The condition for a lower bound is determined by the fact that, if $A \subseteq B \subseteq \tilde{D}$ then $|\tilde{D}$-consistent$| \leq |B$-consistent$| \leq |A$-consistent$|$. The condition is stated in the following theorem whose proof is given in the associate paper[13].

**Theorem 3.** *We consider the equation*

$$(x + y) \oplus (x + (y \oplus \beta)) = \gamma \,,$$

*where the position of the least significant '1' of $x$ is $t$ with $n - 3 \geq t \geq 0$. Let all the submitted queries and the oracle outputs be stored in the set $A$ (note that $\phi \subset A \subseteq \tilde{D}$ where $\tilde{D}$ is a* useful *set). Suppose that there is no query $(0,\, \beta)$ for which the oracle output is $\tilde{\gamma}$ with $\tilde{\gamma}_{n-2} = 1$. Then $|A$-consistent$| > |\tilde{D}$-consistent$|$.*

Instead of establishing a lower bound for the entire seed space $\mathbb{Z}_2^n \times \mathbb{Z}_2^n$, we derive a lower bound for a subset of $\mathbb{Z}_2^n \times \mathbb{Z}_2^n$, denoted by $V_{n,\,0}$ which is the collection of all $(x,\, y)$'s with $x_0 = 1$ (that is, the position of the least significant '1' of $x$ is zero). Note that $|V_{n,\,0}| = 2^{2n-1}$. It is easy to conclude that the lower bound derived for $V_{n,\,0}$ is also a lower bound for the entire seed space $\mathbb{Z}_2^n \times \mathbb{Z}_2^n$.

**Theorem 4.** *A lower bound on the number of queries $(0,\, \beta)$'s, submitted in a batch, to solve*

$$(x + y) \oplus (x + (y \oplus \beta)) = \gamma$$

*where $(x,\, y) \in V_{n,\,0}$ is (i) $3 \cdot 2^{n-4}$ if $n \geq 4$, (ii) 2 if $n = 3$, (iii) 1 if $n = 2$ and (iv) 0 if $n = 1$.*

**Proof.** *(i)* When $n \geq 4$. Let all the submitted queries and the oracle outputs be stored in the set $A$ ($\phi \subset A \subseteq \tilde{D}$ where $\tilde{D}$ is a *useful* set) and $|A$-consistent$| = |\tilde{D}$-consistent$|$ for all $(x,\, y) \in V_{n,\,0}$. By Theorem 3, a *necessary* condition is that there must exist at least one query $(0,\, \beta)$ for which the oracle output is $\tilde{\gamma}$ with $\tilde{\gamma}_{n-2} = 1$ otherwise $|A$-consistent$| > |\tilde{D}$-consistent$|$. We shall henceforth denote a query $(0,\, \beta)$ by $\beta$.

We first encode the bit-string of a query $\beta$ as the edges and the corresponding output $\tilde{\gamma}$ as the nodes (denoted by circles) on a path of the full binary tree as shown in Fig. 1. The possible values of $\beta_i$ are denoted as the edges of the tree between the depth $i$ and the depth $(i + 1)$ (the root of the tree is at depth 0).

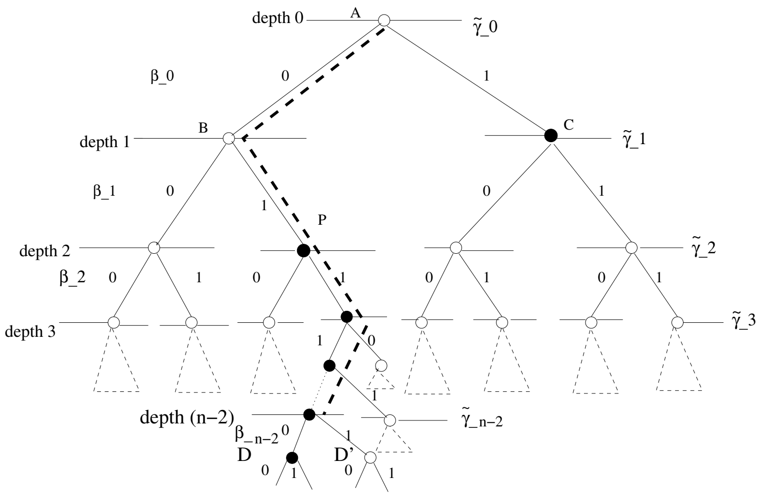


**Fig. 1.** An arbitrary path $P$ in the subtree (black node indicates value 1 and white node 0)

Similarly the possible values of $\tilde{\gamma}_i$ can be assigned to the nodes at the depth $i$. Note that all possible $2^n$ queries are encoded in the tree.

*The Approach.* We shall isolate a subtree and show that, if an arbitrary path in that subtree is *not* present as the prefix of one of the submitted queries then there exists a seed $(x, y) \in V_{n,\,0}$ such that, on all other queries, the outputs are $\tilde{\gamma}$'s with $\tilde{\gamma}_{n-2} = 0$. Therefore a lower bound is the number of all paths present in that particular subtree.

*Node Assignment Rule.* The rule shows how to select the values of $\tilde{\gamma}$'s for all the queries. In the nodes of the *entire* tree we now put the values of $\tilde{\gamma}$. We select an *arbitrary* path $P$ (which is a prefix of a query) in the subtree whose leaf nodes are at the depth $(n-2)$ and whose first two edges are $(0,\ 1)$ and $(1,\ 0)$ and $(1,\ 1)$ (see Fig. 1). Note that the values at the nodes $B$ and $C$ will be 0 and 1 respectively because $x_0 = 1$. Now we put 1 in all nodes on the path $P$ from the depth 2 till the depth $(n-2)$. The two child nodes $D$ and $D'$ of the last node on $P$ are assigned 0 and 1 arbitrarily. All other nodes in the tree are assigned 0. The intuition that such an assignment rule gives a valid solution is derived from an observation in Table 1 (see Sect. 4.1) that the matrix cut off by rows R(1), R(2) and R(3) and columns Col(2), Col(3) and Col(4) has only diagonal elements 1 (this fact is used to prove Lemma 1).

*Proof Continued.* Suppose $P$ is not a prefix of any query in $A$. As shown in Fig. 1, all nodes at depth $(n-2)$, except the one on the path $P$, are assigned zeros. Therefore, there is no query $\beta$ in $A$ such that the corresponding output $\tilde{\gamma}$ has $\tilde{\gamma}_{n-2} = 1$. This leads to a contradiction. Therefore, there must be a query in $A$ whose prefix is $P$. Now $P$ is an arbitrary path in the subtree constructed above. Now the total number of paths (or prefixes of queries) in the subtree is $3 \cdot 2^{n-4}$. The following lemma completes the proof.

**Lemma 1.** *For any arbitrary $P$ with the first two edges* $(0,\ 1)$ *or* $(1,\ 0)$ *or* $(1,\ 1)$ *in the tree constructed above, all queries and their outputs encoded in the tree according to the* Node Assignment Rule*, produce a valid solution* $(x,\ y) \in V_{n,\ 0}$.

**Proof.** For any arbitrary $P$, the core $G_i$'s $(0 \leq i \leq n-2)$, computed from the values of $\tilde{\gamma}$'s and $\beta$'s (according to the *Node Assignment Rule*), are of one of the following forms

$$G_0 = \{(0,\ 0,\ 0,\ 0),\ (0,\ 1,\ 0,\ 0)\}\,,$$
$$G_i = \{(0,\ 0,\ 0,\ 0),\ (0,\ 1,\ 0,\ 0),\ (0,\ 0,\ 1,\ a_i),\ (0,\ 1,\ 1,\ b_i)\}\,, \qquad 1 \leq i \leq n-2$$

where $a_i,\ b_i \in [0,\ 1]$ and $a_i = 1 \oplus b_i$. Now each $S_i > 0$ (obtained from Table 1 using the $G_i$'s). Therefore, the number of valid solutions $S = 4 \cdot \prod_{i=0}^{n-2} S_i > 0$ (see Proposition 2). In fact the number solutions is 8 (verification of a part of Theorem 1). □□
The proofs of *(ii)*, *(iii)* and *(iv)* are immediate from Table 1. □

*Lower bounds for the equation* $(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma$. In this case, lower bounds on the number of *batch* queries, for $n = 2$ and 3, are 2 and 3 respectively. This can be proved by searching through all possible $x$, $y$, $\alpha$, $\beta$ and $\tilde{\gamma}$ exhaustively. However, the situation becomes intractable when $n \geq 4$ when the number of all possible $x$, $y$, $\alpha$, $\beta$ and $\tilde{\gamma}$ for only 3 queries becomes extremely large ($2^{60}$ for $n = 4$). We did extensive experiments with many test vectors and found that three queries were insufficient to solve the equations when $n \geq 4$. We state the following conjecture.

*Conjecture 1.* A lower bound on the number of queries $(\alpha,\ \beta)$, submitted in a batch, to solve

$$(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma$$

is 4 if $n \geq 4$.

## 6 Algorithms

In this section, we present two algorithms Algorithm 1 and Algorithm 2 to solve (4) and (3) with a set of *batch* queries. The inputs to the algorithms are the *bit-length* $n$, the oracle $O$ and the Table 1. The outputs are the $G_i$'s (defined in Sect. 4.2) computed from a set of queries and their replies. Our target is to select a subset of all possible queries such that the set of solutions derived from the $G_i$'s is the same as $\tilde{D}$-consistent, for all $(x,\ y)$'s. Algorithm to compute the actual solution set from the set of $G_i$'s is described in Sect. 2.5 of [13]. Below we discuss the motivation and correctness of the algorithms; as we exclude many details, the pseudocode covers all cases.

**Discussion: Algorithm 1** *(sketch).* The number of queries required by the algorithm is $2^{n-2}$ which is one fourth of all possible $2^n$ queries (note that a lower bound is $\frac{3}{4} \cdot 2^{n-2}$). The two *for loop*'s (in steps 8-17 and 10-13) are the most

**Algorithm 1.** Algorithm to solve the equation $(x + y) \oplus (x + (y \oplus \beta)) = \gamma$

```
Require: Oracle $O$, $n$, Table $T$
Ensure: The core $G_i$'s
 1: If $n \le 0$ then exit with a comment "Invalid Input";
 2: If $n = 1$ then return an empty set $\phi$ indicating all 4 possible solutions and exit;
 3: $\beta = (1, 1, \cdots, 1, 1)_n$;        /*The first query*/
 4: $\tilde{\gamma} = O(\beta)$;        /*Oracle output*/
 5: $Q = \{\beta\}$ and $A = \{(0, \beta, \tilde{\gamma})\}$;        /*Collecting query and output*/
 6: If $n = 2$ then Go to Step 20;
 7: If $n > 3$     /*If $n = 3$, the execution automatically jumps to Step 18)*/
 8:         For all $t \in [1, n-3]$ in increasing order
 9:                 {Initialize $Q' = \phi$;
10:                 For all $\beta \in Q$
11:                         {$\beta' = (1, 1, \cdots, \beta'_t = 0, \beta_{t-1}, \cdots, \beta_0)$;        /*New query*/
12:                          $O(\beta') = \tilde{\gamma}'$;        /*Oracle output*/
13:                          $Q' = Q' \cup \{\beta'\}$ and $A = A \cup \{(0, \beta', \tilde{\gamma}')\}$;}
14:                 $\beta' = (1, 1, \cdots, \beta'_t = 1, 0, \cdots, 0)$;        /*New query*/
15:                 $O(\beta') = \tilde{\gamma}'$;        /*Oracle output*/
16:                 $Q' = Q' \cup \{\beta'\}$ and $A = A \cup \{(0, \beta', \tilde{\gamma}')\}$;   /*Collecting output*/
17:                 $Q = Q \cup Q'$;}
18: $\beta' = (1, 1, \cdots, \beta'_{n-2} = 1, 0, \cdots, 0)$;        /*last query*/
19: $O(\beta') = \tilde{\gamma}'$ and $A = A \cup \{(0, \beta', \tilde{\gamma}')\}$;        /*Collecting Oracle output*/
20: Return the core $G_i$'s for all $i \in [0, n-2]$ computed from $A$.
```

important parts of the algorithm. For easy understanding of the algorithm, let us see how the algorithm works when the position of the least significant '1' of $x$ is zero (i.e., $x_0 = 1$). Note that we have to submit queries such that the $G_i$'s $(0 \le i \le n-2)$ obtained from them correspond to $S_0 = 2$ and $S_i = 1 \forall i \in [1, n-2]$ (Proposition 2, Theorem 1). In the $t$th iteration of the bigger loop we submit a set of queries which ensures that $G_t = \{(0,1,1,a), (0,0,1,b), (0,1,0,c)\}$ which implies that $S_t = 1$. The $t$th iteration also produces at least one output $\tilde{\gamma}$ with $\tilde{\gamma}_{t+1} = 1$ which will be used in the next loop. If there is no output with $\tilde{\gamma}_{t+1} = 1$ then $S_{t+1} > 1$ and hence the algorithm fails (see Theorem 3). The proof of correctness of the algorithm when $x_0 \neq 1$ is similar to the above argument.

**Discussion: Algorithm 2** *(sketch).* The number of queries required by the algorithm is 6 which is two more than the best known lower bound (also note that the number of all possible queries is $2^{2n}$). The proof of correctness of this algorithm is by showing that the submitted queries produce $S_i = 1$ for all $i \in [0, n-2]$ (Proposition 2, Theorem 2). Six queries are submitted in steps 3, 5, 8, 10, 12, 14. Now we consider only the first two queries in steps 3 and 5. For these queries, $G_i = \{(1,\ 0,\ \tilde{\gamma}_i,\ \tilde{\gamma}_{i+1}),\ (0,\ 1,\ \tilde{\gamma}'_i,\ \tilde{\gamma}'_{i+1})\}$ if $i$ even. Note that, in this case, if $\tilde{\gamma}_i = \tilde{\gamma}'_i$ then $S_i = 1$ otherwise $S_i = 2$ (from Table 1). Also observe that $G_i = \{(1,\ 0,\ \tilde{\gamma}_i,\ \tilde{\gamma}_{i+1}),\ (1,\ 0,\ \tilde{\gamma}'_i,\ \tilde{\gamma}'_{i+1})\}$ if $i$ odd. In this case, if $\tilde{\gamma}_i \neq \tilde{\gamma}'_i$ then $S_i = 1$ otherwise $S_i = 2$. Now, we detect a combinatorial relation in the precomputed Table 1 which establishes that, if $S_i = 2$ then $S_{i-1} = 1$ (leaving out the proof). The 3rd and the 4th queries

**Algorithm 2.** Algorithm to solve the equation $(x+y) \oplus ((x+\alpha)+(y \oplus \beta)) = \gamma$

```
Require: Oracle $O$, $n$, Table $T$
Ensure: the core $G_i$'s
 1: If $n \leq 0$ then exit with a comment "Invalid Input";
 2: If $n = 1$ then return an empty set $\phi$ indicating all 4 possible solutions and exit;
 3: $(\alpha[1], \beta[1]) = ((11 \cdots 11)_n, (00 \cdots 00)_n)$;/*First Query*/
 4: $\tilde{\gamma}[1] = O(\alpha[1], \beta[1])$;    /*Oracle Output*/
 5: $(\alpha[2], \beta[2]) = ((\cdots 101010)_n, (\cdots 010101)_n)$;/*Second Query*/
 6: $\tilde{\gamma}[2] = O(\alpha[2], \beta[2])$;    /*Oracle Output*/
 7: If $n = 2$ then Go to Step 16;
 8: $(\alpha[3], \beta[3]) = ((\cdots 101010)_n, (\cdots 000000)_n)$;/*Third Query*/
 9: $\tilde{\gamma}[3] = O(\alpha[3], \beta[3])$;    /*Oracle Output*/
10: $(\alpha[4], \beta[4]) = ((\cdots 111111)_n, (\cdots 010101)_n)$;/*Fourth Query*/
11: $\tilde{\gamma}[4] = O(\alpha[4], \beta[4])$;    /*Oracle Output*/
12: $(\alpha[5], \beta[5]) = ((\cdots 000000)_n, (\cdots 010101)_n)$;/*Fifth Query*/
13: $\tilde{\gamma}[5] = O(\alpha[5], \beta[5])$;    /*Oracle Output*/
14: $(\alpha[6], \beta[6]) = ((\cdots 101010)_n, (\cdots 111111)_n)$;/*Sixth Query*/
15: $\tilde{\gamma}[6] = O(\alpha[6], \beta[6])$;    /*Oracle Output*/
16: $A = \{(\alpha[i], \beta[i], \tilde{\gamma}[i]) \mid$ for all $i$'s$\}$
17: Return the core $G_i$'s for all $i \in [0, n-2]$ computed from $A$.
```

are generated from the second query assuming $S_i = 2$ for some odd $i$'s. We change all the even numbered bits of the second query $(\alpha[2], \beta[2])$. It can be shown that making $(\alpha[2]_i, \beta[2]_i) = (0,0)$ and $(1,1)$ for all even $i$'s ensures that $S_i = 1$ for all odd $i$'s. Exactly the same way the 5th and the 6th queries are generated from the second query $(\alpha[2], \beta[2])$ assuming that $S_i = 2$ for some even $i$'s. Now we change the odd numbered bits of $(\alpha[2], \beta[2])$ to $(0,0)$ and $(1,1)$ to ensure that all $S_i = 1$ for all even $i$'s. Therefore, the number of solutions derived from these 6 queries is $S = 4 \cdot \prod_{i=0}^{n-2} S_i = 4$ as suggested in Theorem 2.

## 7 Cryptographic Applications

**Security of Modular addition against DC for Batch Queries.** In view of the large scale application of modular addition in symmetric cryptography, our results seem effective in the evaluation of security of cipher components which mixes two different group operations addition and XOR. The fact that only 6 *queries* submitted in a batch (which is a more practical attack scenario than that with adaptive queries) are sufficient to reduce the search space of the secret $(x, y)$ from $2^{2n}$ to only 4 (for all $n \geq 4$) should be recognized as a warning to the designers (see Algorithm 2). On the other hand the high exponential lower bound on the number of queries for another differential equation of addition $(3 \cdot 2^{n-4}$ for all $n \geq 4)$ underlines an important theoretical reference point which advocates it for being *relatively strong* under DC (see Theorem 4). One direct application of our results in practical cryptanalysis is described below. At this moment, we are not aware of other applications of the results, yet it can very likely be used to evaluate cryptographic strengths of many modern ciphers which use modular multiplication combined with modular addition and XOR.

**Cryptanalysis of Helix.** Helix, proposed by Ferguson *et al.* [6], is a stream cipher with a combined MAC functionality. This cipher was a candidate for consideration in the 802.11i standard. The main component of the primitive is combination of *addition* and XOR. The fact that the internal state of Helix depends on the plaintext allows for cryptanalysis with *chosen plaintexts* (CP) and *adaptive chosen plaintexts* (ACP). Muller mounted an ACP attack which recovers the secret key of the Helix cipher with $2^{12}$ plaintexts. In [13], the data complexity of the attack was improved to $2^{10.41}$. We refer the readers to [12] for a detailed analysis of the attack. Our key recovery attack goes on the same line as Muller's attack. We cannot describe the attack in full detail because of space constraints, however, we pick out a portion which is critical to our CP attack. The crux of the whole attack is solving the equation $(x + y) \oplus (x + (y \oplus \beta)) = \gamma$ with $\beta$'s and the corresponding $\gamma$'s to recover the secret information $(x, y)$, in the framework described in Sect. 3.1, for 50 times ($n = 32$ for the Helix cipher). Every time $\beta$ corresponds to a CP. Algorithm 1 shows that the above equation can be solved with $2^{n-2}$ CP's ($2^{30}$ for $n = 32$). Therefore, the total data complexity of our CP attack is $50 \cdot 2^{30}$, i.e., $2^{35.64}$ plaintexts.

## 8 Conclusion and Further Research

We showed a lower bound on the number of *batch* queries to solve a DEA which is optimal up to a constant factor. For solving another DEA, our algorithm uses number of queries which is constant asymptotically. Our results are used directly to recover the key of the Helix cipher with chosen plaintexts rather than with adaptive chosen plaintexts which has so far been the best CP attack on this cipher. The paper also leaves many interesting questions open. One possible research direction may be to close the gap between the lower and upper bounds on the number of queries to solve DEA. Another way to extend the work is to analyze components which combine more complex transformations such as modular multiplication, $T$-functions with addition.

**Acknowledgments.** We thank Jongsung Kim of KUleuven and the anonymous reviewers of Indocrypt 2005 for useful suggestions.

## References

1. I. A. Ajwa, Z. Liu, P. S. Wang, “Gröbner Bases Algorithm,” *ICM Technical Report*, February 1995, Available Online at `http://icm.mcs.kent.edu/reports/1995/gb.pdf`.
2. T. A. Berson, “Differential Cryptanalysis Mod $2^{32}$ with Applications to MD5,” *Eurocrypt 1992* (R. A. Rueppel, ed.), vol. 658 of *LNCS*, pp. 71-80, Springer-Verlag, 1993.

3. C. Burwick, D. Coppersmith, E. D'Avignon, Y. Gennaro, S. Halevi, C. Jutla, S. M. Matyas Jr., L. O'Connor, M. Peyravian, D. Safford and N. Zunic, "MARS – A Candidate Cipher for AES," Available Online at `http://www.research.ibm.com/security/mars.html`, June 1998.
4. E. Biham, A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," *Crypto '90* (A. Menezes, S. A. Vanstone, eds.), vol. 537 of *LNCS*, pp. 2-21, Springer-Verlag, 1991.
5. A. Biryukov, D. Wagner, "Slide Attacks,"*Fast Software Encryption 1999*, (Lars R. Knudsen, ed.), vol. 1636 of *LNCS*, pp. 245-259, Springer-Verlag, 1999.
6. N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, S. Lucks, T. Kohno, "Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive," *Fast Software Encryption 2003* (T. Johansson, ed.), vol. 2887 of *LNCS*, pp. 330-346, Springer-Verlag, 2003.
7. M. E. Hellman, "A Cryptanalytic Time-Memory Trade-off," IEEE Transaction on Information Theory, vol. IT-26, No. 4, July, 1980.
8. A. Klimov, A. Shamir, "New Cryptographic Primitives Based on Multiword T-Functions," *Fast Software Encryption 2004* (B. Roy, W. Meier, eds.), vol. 3017 of *LNCS*, pp. 1-15, Springer-Verlag, 2004.
9. X. Lai, J. L. Massey, S. Murphy, "Markov Ciphers and Differential Cryptanalysis," *Eurocrypt '91* (W. Davis, ed.), vol. 547 of *LNCS*, pp. 17-38, Springer-Verlag, 1991.
10. H. Lipmaa, S. Moriai, "Efficient Algorithms for Computing Differential Properties of Addition," *FSE 2001* (M. Matsui, ed.), vol. 2355 of *LNCS*, pp. 336-350, Springer-Verlag, 2002.
11. L. Lipmaa, J. Wallén, P. Dumas, "On the Additive Differential Probability of Exclusive-Or," *Fast Software Encryption 2004* (B. Roy, W. Meier, eds.), vol. 3017 of *LNCS*, pp. 317-331, Springer-Verlag, 2004.
12. F. Muller, "Differential Attacks against the Helix Stream Cipher," *Fast Software Encryption 2004* (B. Roy, W. Meier, eds.), vol. 3017 of *LNCS*, pp. 94-108, Springer-Verlag, 2004.
13. S. Paul and B. Preneel, "Solving Systems of Differential Equations of Addition (Extended Abstract)," *10th Australasian Conference on Information Security and Privacy, ACISP 2005* (Colin Boyd and Juan Gonzalez, eds.), vol. 3574 of *LNCS*, pp. 75-88, Springer-Verlag, 2005, Extended Version available online on IACR ePrint Archive as Report 2004/294 at `http://eprint.iacr.org/2004/294`, April 2005.
14. R. L. Rivest, M. Robshaw, R. Sidney, Y. L. Yin, "The RC6 Block Cipher," Available Online at `http://theory.lcs.mit.edu/ rivest/rc6.ps`, June 1998.
15. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C.Hall, N. Ferguson, "The Twofish Encryption Algorithm: A 128-Bit Block Cipher," John Wiley & Sons, April 1999, ISBN: 0471353817.
16. O. Staffelbach, W. Meier, "Cryptographic Significance of the Carry for Ciphers Based on Integer Addition," *Crypto '90* (A. Menezes, S. A. Vanstone, eds.), vol. 537 of *LNCS*, pp. 601-614, Springer-Verlag, 1991.
17. D. Wagner, "The Boomerang Attack," *Fast Software Encryption 1999*, (Lars R. Knudsen, ed.), vol. 1636 of *LNCS*, pp. 156-170, Springer-Verlag, 1999.
18. J. Wallén, "Linear Approximations of Addition Modulo $2^n$," *Fast Software Encryption 2003* (T. Johansson, ed.), vol. 2887 of *LNCS*, pp. 261-273, Springer-Verlag, 2003.

# Design Principles for Combiners with Memory

Frederik Armknecht, Matthias Krause, and Dirk Stegemann

Theoretical Computer Science,
University of Mannheim, Germany
{armknecht, krause, stegemann}@th.informatik.uni-mannheim.de

**Abstract.** Stream ciphers are widely used for online-encryption of arbitrarily long data, for example when transmitting speech-data between a mobile phone and a base station. An important class of stream ciphers are combiners with memory, with the $E_0$ generator from the Bluetooth standard for wireless communication being their most prominent example. In this paper, we develop design principles for increasing the resistance of combiners with memory against the most dangerous types of cryptanalytic attacks, namely correlation attacks and algebraic attacks. In the case of algebraic attacks, we introduce the first method to guarantee lower bounds on the attack complexity. Starting from the design of the $E_0$ generator, we combine our results in order to construct ciphers that are simultaneously strengthened against both kinds of attacks. Our analysis shows that small changes in the design of $E_0$ already suffice to improve its security enormously.

**Keywords:** Stream cipher, combiners with memory, algebraic attacks, correlation attacks, Bluetooth $E_0$.

## 1 Introduction

Today, electronic communication has gained more and more importance, invoking an increasing demand for confidential data transmission. Widely used are keystream generators which produce bitstreams $z := z_1, z_2, \ldots$ of arbitrary length in dependence on a secret initial value $K \in \{0,1\}^n$. The sender encrypts a stream of plaintext bits $p := p_1, p_2, \ldots$ to a stream of ciphertext bits $c := c_1, c_2, \ldots$ by XOR-ing $p$ and $z$ componentwise, i.e., $c_t := p_t \oplus z_t$. A receiver who shares the secret key $K$ can produce $z$ in the same way as the sender and decrypt $c_t$ via $p_t = c_t \oplus z_t$. Following Kerckhoff's principle, it is assumed that an adversary knows the specification of the keystream generator and some of the keystream bits $z_t$, whereas $K$ is secret to him. Consequently, an attack consists of recovering the secret key $K$.

An important class of keystream generators are combiners with memory. Since their introduction in [17] to overcome the trade-off between linear complexity and correaltion immunity, they have been widely examined in cryptography and have found their way into practical applications. The perhaps best known example used in practice is the $E_0$ keystream generator which is part of

© Springer-Verlag Berlin Heidelberg 2005

the Bluetooth standard, a widely applied standard for short- to mid-distance wireless communication between mobile devices.

The best attacks against combiners with memory that are currently known are correlation attacks [10, 11, 18, 14, 15] and algebraic attacks [1, 6]. A correlation attack consists of finding and exploiting linear functions

$$L(X_t, \ldots, X_{t+r-1}, z_t, \ldots, z_{t+r-1})$$

which are biased, i.e., equal to zero with some probability $\neq 1/2$. Algebraic attacks mark somehow the opposite. Here, valid non-linear equations of preferably low degree are used to describe $K$ by a system of equations.

Although much effort has been put into the refinement of these attacks, only little is known about how to resist them. Our results are design principles for combiners with memory to improve the security in respect of both kinds of attacks.

In general, finding highly biased linear functions $L$ for correlation attacks are only feasible for small values of $r$. However, in the case of $E_0$, the best currently known attack [14] uses a special class of biased linear functions which allow for an exhaustive search for the best correlations even for relatively large values of $r$, more precisely for $r$ up to 25. We show how to avert this approach completely.

Further on, we introduce a design principle which guarantees that all valid equations in $X_t, \ldots, X_{t+r-1}, z_t, \ldots, z_{t+r-1}$ have a degree greater than or equal to a certain lower bound. This marks the first lower bound on the complexity of algebraic attacks derived so far.

Our proposals can be easily combined to construct combiners with memory which are strengthend against both correlation and algebraic attacks.

The paper is structured as follows. Section 2 defines combiners with memory and explains correlation and algebraic attacks against them. In Sects. 3 and 4, we put correlation attacks and algebraic attacks into theoretical frameworks and derive according countermeasures, which we use in Sect. 5 to introduce and examine modified versions of $E_0$. Section 6 concludes the paper.

## 2 Combiners with Memory

A combiner with memory, or shortly a $(k, \ell)$-combiner, consists of $k$ driving devices, a finite state machine (FSM) $\mathcal{C}$ with an $\ell$ bit state and two mappings $f : \{0,1\}^\ell \times \{0,1\}^k \to \{0,1\}$ and $\delta : \{0,1\}^\ell \times \{0,1\}^k \to \{0,1\}^\ell$. Let $X_t \in \{0,1\}^k$ denote the output of the driving devices and $C_t \in \{0,1\}^\ell$ the state of the FSM at clock $t \geq 1$.

Combiners with memory are regulary clocked. At each clock, one keystream bit is produced as $z_t = f(C_t, X_t)$, and the state of the FSM is updated to $C_{t+1} := \delta(C_t, X_t)$. Combiners with memory have the advantage that they combine high algebraic degree with high correlation immunity (cf. [19]). Correlation immunity means that the output $z_t$ is not or only weakly correlated to the sum of a subset of the input bits.

For the rest of the paper, we focus on the well-studied subclass of combiners with memory that use Linear Feedback Shift Registers (LFSRs) as driving devices. The output $x_t$ of an LFSR is computed as $x_t = L_t(K)$, where $L_t$ denotes a known linear Boolean function and $K \in \{0,1\}^n$ the LFSR's secret initial state.

A famous practical LFSR-based combiner with memory is the $(4,4)$-combiner $E_0$, which uses four LFSRs of lengths 25, 31, 33 and 39, respectively. Based on an internal key $K$ that is re-initialized frequently, the Bluetooth stream cipher utilizes $E_0$ to produce consecutive, fixed-length frames of keystream. In [3], it was shown that an efficient attack on $E_0$ implies an efficient attack on the whole cipher. Consequently, improving the security of $E_0$ is a natural demand.

Amongst all publicly known attacks on combiners with memory, the fastest are correlation attacks and algebraic attacks. In the following, we give a brief description of these attacks.

Correlation attacks exploit linear equations

$$L(X_t, \ldots, X_{t+r-1}, z_t, \ldots, z_{t+r-1}) = 0$$

which are true with probability $1/2 + \lambda$ with $\lambda \neq 0$. $\lambda$ is called the bias. General methods to systematically compute the equations with the highest value of $|\lambda|$ exist (e.g., cf. [10]), but since their complexity is exponential in $k$, $\ell$ and $r$, these methods are only feasible for small values.

However, if the output function $f(C, X)$ can be written as the sum of two functions $\alpha(X)$ and $\beta(C)$, i.e., $z_t = \alpha(X_t) \oplus \beta(C_t)$, one can try to use biased linear combinations in the expressions $\beta(C_t)$. More precisely, an attacker looks for coefficients $\gamma = (\gamma_0, \ldots, \gamma_{r-1})$ such that

$$\lambda := \lambda(\gamma) := \left( Pr\left[ \bigoplus_{i=0}^{r-1} \gamma_i \cdot \beta_{t+i} = 0 \right] - Pr\left[ \bigoplus_{i=0}^{r-1} \gamma_i \cdot \beta_{t+i} = 1 \right] \right) \neq 0 \ . \qquad (1)$$

Amongst all non-trivial biases, the attacker is interested in finding and using the maximum bias, which is defined as $\lambda_{\max} := \max\{|\lambda(\gamma)|\}$.

The output function of $E_0$ is exactly of the desired type. In [14], it was proved that $\lambda_{\max} = 25/256$ for $r \leq 25$, where 25 is the length of the shortest LFSR. This observation and the exploit of a synchronization flaw led to the best currently known attack on the Bluetooth cipher [15]. The complexities of this attack are given in Table 1.

**Table 1.** The complexity of the fastest correlation attack on $E_0$ as presented in [15]

| $\lambda_{\max}$ | Frames | Data | Time | Space |
|---|---|---|---|---|
| $\lambda$ | $m = \max(\frac{1}{\lambda^{10}}, \frac{236.59}{\lambda^8})$ | $24m$ | $36m + 3 \cdot 2^{18} \cdot \min(m, 2^{18})$ | $m$ |
| $\frac{25}{256}$ | $2^{34.74}$ | $2^{39.32}$ | $2^{40.17}$ | $2^{34.74}$ |

Algebraic attacks are based on solving systems of equations. An attacker uses a Boolean function $F : \{0,1\}^{k\cdot r} \rightarrow \{0,1\}$ such that for all clocks $t$, it holds that

$$F(X_t, \ldots, X_{t+r-1}, z_t, \ldots, z_{t+r-1}) = 0 \ . \tag{2}$$

The existence of such equations has been proved in [1]. Using the equivalence $X_t = L_t(K)$, (2) allows to write a system of equations which describes the secret key $K$ in dependence of the observed keystream $(z_t)$. Accordingly, the secret key $K$ can be recovered by solving the system of equations.

Although computing the solution is NP-hard in general, it might become easier if the number of known keystream bits and therefore the number of equations increases. Let $R$ denote the number of accessible equations and $\mu$ the number of occurring monomials. If $R \ll \mu$, the best method known today is to compute Groebner bases. Unfortunately, until now it is impossible to predict the time effort, albeit simulations indicate that the effort drops with increasing number of equations (cf. [9]).

In the case of $R \approx \mu$, linearization [7] is the first choice. The idea of linearization is to substitute each occurring monomial by a new variable and to treat the whole system as a system of linear equations, making it easily solvable by Gaussian elimination.

For the case that the number of equations exceeds the number of monomials, one might reduce the degree of the equations in a precomputation step. This idea is known as "fast algebraic attacks", which have been introduced in [6] and further improved in [2,12]. However, the attack scenario is more restrictive as it requires the attacker to know many successive keystream bits and (2) to have a special structure.

All theses approaches have in common that their time effort strongly depends on the degree $d$ of the incorporated equations. The lower the degree, the faster the attacks. Hence, a natural countermeasure against such attacks is to avoid low degree equations. For this subject, the terminology "algebraic immunity" has been introduced in [16] and extended to combiners with memory in [4].

For the rest of the paper, we will concentrate on algebraic attacks where $R \approx \mu$ and $\mu$ is approximately $\binom{n}{d}$. If $\varphi$ denotes the number of functions $F$ of degree $d$ fulfilling (2) and $n$ denotes the key size $|K|$, then the amount of data is $\approx \binom{n}{d}/\varphi$, and the memory and time efforts are in $O\left(\binom{n}{d}^2\right)$ and $O\left(\binom{n}{d}^3\right)$, respectively.

The currently best algebraic attack on $E_0$ in this scenario uses a function $F$ of degree 4 over 4 clocks. The corresponding attack complexities are given in Table 2.

**Table 2.** The efforts of an algebraic attack on $E_0$ with key size $n$ and an equation of degree $d$

| | $\#F$ | Data | Time | Space |
|---|---|---|---|---|
| General | $\varphi$ | $O\left(\binom{n}{d}/\varphi\right)$ | $O\left(\binom{n}{d}^3\right)$ | $O\left(\binom{n}{d}^2\right)$ |
| $E_0$: $n = 128$, $d = 4$ | 1 | $2^{23.35}$ | $2^{70.04}$ | $2^{46.69}$ |

Observe that the performance of algebraic attacks drops if the minimum possible degree $d$ increases. The same is true for the correlation attack if $\lambda_{\max}$ shrinks. In the following, we derive several countermeasures against these attacks for general $(k, l)$-combiners. In Sect. 5, we will then apply these results to construct variations of $E_0$ with improved security.

## 3 Design Principles Against Correlation Attacks

As stated in Sect. 2, the correlation attack exploits biased linear equations $L(X_t, \ldots, X_{t+r-1}, z_t, \ldots, z_{t+r-1}) = 0$. Finding functions with the highest absolute bias is difficult in general. In the case that the output function can be written as the sum of two functions $\alpha(X)$ and $\beta(C)$, i.e., $z_t = \alpha(X_t) \oplus \beta(C_t)$ for all keybits $z_t$, one can instead try to find biased linear combinations in the expressions $\beta(C_t)$. This allows to treat much higher values of $r$ than the general method. In the case of $E_0$, this approach yielded the fastest attack known so far (cf. [15]).

The theory in this section is motivated by this setting. Golić [11] already established a theoretical framework and a lot of interesting results on estimating correlations between XOR-sums of $X$-inputs and $Z$-outputs for functions $Z = F(X, Y)$, which matches exactly the situation of combiners with memory. What we do in the following is to derive a formula (Theorem 1) allowing to efficiently compute the special correlations that the correlation attack of Lu and Vaudenay [15] is based on. We then conclude some design criteria for avoiding this type of correlations and show how these correlations can be reduced by a local optimization of the $E_0$-design.

We consider the scenario that for each time unit $t \geq 1$, there is a separate Boolean function $\beta_t : \{0,1\}^\ell \times \{0,1\}^k \to \{0,1\}$, revealing information about $C_t$ and $X_t$.

Let $F^r : \{0,1\}^\ell \times \{0,1\}^{k \cdot r} \to \{0,1\}$ be defined by

$$F^r(C_1, X_1, \ldots, X_r) := \beta_1(C_1, X_1) \oplus \ldots \oplus \beta_r(C_r, X_r) \ ,$$

where $C_1 \in \{0,1\}^\ell$, $X_1, \ldots, X_r \in \{0,1\}^k$ and $C_{t+1} := \delta(C_t, X_t)$. We are interested in the value $\lambda := \lambda(F^r) := Pr[F^r = 0] - Pr[F^r = 1]$.

Observe that (1) is captured in this scenario by setting $\beta_t(C_t, X_t) := \beta(C_t)$ if $\gamma_t = 1$ and $\beta_t :\equiv 0$ otherwise.

We will express $\lambda$ in terms of the transition matrix $P$ and the bias matrices $B_t$ of the FSM $\mathcal{C}$, which are defined as follows.

**Definition 1.** *For all states $C, C' \in \{0,1\}^\ell$, we denote by $p(C, C')$ the probability that state $C$ will change into $C'$, i.e., $p(C, C') = 2^{-k} \left|\{X; \delta(C, X) = C'\}\right|$. Additionally, let $b_t(C, C')$ be the value*

$$2^{-k} \cdot (|\{X; \beta_t(C, X) = 0, \delta(C, X) = C'\}| - |\{X; \beta_t(C, X) = 1, \delta(C, X) = C'\}|) \ .$$

*We call the matrix $P = (p(C, C'))_{C,C' \in \{0,1\}^\ell}$ the transition matrix of $\mathcal{C}$ and the matrix $B_t = (b_t(C, C'))_{C,C' \in \{0,1\}^\ell}$ the bias matrix of $\mathcal{C}$ corresponding to time unit $t$.*

**Theorem 1.** *It holds for all $r \geq 1$ that*

$$\lambda = \lambda(F^r) = 2^{-\ell}\left(e^T\right) \circ B_1 \circ \cdots \circ B_r \circ e \ , \tag{3}$$

*where $e$ denotes the constant-1 vector of length $2^\ell$.*

As usual, we denote by $B^T$ the transpose of a given matrix $B$. The proof of Theorem 1 can be found in Appendix A.

Formula (3) allows to compute the biases which are relevant for correlation attacks against combiners with memory and to derive corresponding design criteria to immunize them against these types of attacks. In particular, (3) yields two different criteria for $\delta$ and $\beta_t$ in order to achieve that $\lambda(F^r) = 0$ for all $r \geq 1$. Note that these and more general criteria could also be derived from the more general and somewhat different calculations in [11].

The first one considers the situation that $\beta_t$ is independent of $X \in \{0,1\}^k$, i.e., $\beta(C,X) = \beta(C)$ for all $X$. This reflects the situation of $E_0$.

**Definition 2.** *We call $\beta_t$ to be balanced if $|\beta_t^{-1}(0)| = |\beta_t^{-1}(1)|$. Furthermore, we say that $\delta$ is balanced if $k = \ell$ and $p(C,C') = 2^{-k}$ for all $C, C'$.*

**Theorem 2.** *Let $\beta_t$ either be $\equiv 0$ or depend only on $C$ and be balanced, the latter being true for at least one time unit $t$. If $\delta$ is also balanced, then $\lambda = 0$.*

*Proof.* If $\beta_t \equiv 0$, then $B_t = P$. Because of $(e^T) \cdot P = e^T$, we can assume w.l.o.g. that $\beta_1 \not\equiv 0$. Observe that the property of $\beta_t$ being balanced implies that $\sum_C (-1)^{\beta_1(C)} = 0$. Let $X_{(C,C')} := \{X; \delta(C,X) = C'\}$. If $\beta_t$ depends only on $C$, then $b_t(C,C')$ can be rewritten to

$$b_t(C,C') = \left\{\begin{array}{cl} 0 & \text{if } X_{(C,C')} = \emptyset \\ |X_{(C,C')}|/2^k & \text{if } X_{(C,C')} \neq \emptyset, \beta(C) = 0 \\ -|X_{(C,C')}|/2^k & \text{if } X_{(C,C')} \neq \emptyset, \beta(C) = 1 \end{array}\right\} = (-1)^{\beta_t(C)} \cdot p(C,C')$$

Let $v^T := (e^T) \cdot B_1$. We show that $v$ is already the all-zero vector which concludes the proof. Let $(v^T)_C$ denote the $C$-th entry of $v^T$. Then,

$$(v^T)_C = \sum_C (-1)^{\beta_1(C)} p(C,C') = 2^{-k} \cdot \underbrace{\sum_C (-1)^{\beta_1(C)}}_{=0} = 0 \ . \qquad \square$$

In the case that the functions $\beta_t$ are not independent of $X$, it is also possible to entirely avoid correlations if we put some additional conditions on $\beta_t$.

**Definition 3.** *The function $\beta : \{0,1\}^\ell \times \{0,1\}^k \to \{0,1\}$ is called $C$-balanced if for all states $C \in \{0,1\}^\ell$ it holds that*

$$\left|\{X \in \{0,1\}^k,\ \beta(C,X) = 0\}\right| = \left|\{X \in \{0,1\}^k,\ \beta(C,X) = 1\}\right| \ .$$

**Lemma 1.** *Let $B$ denote the bias matrix with respect to the state transition function $\delta : \{0,1\}^\ell \times \{0,1\}^k \to \{0,1\}^\ell$ and a $C$-balanced function $\beta : \{0,1\}^\ell \times \{0,1\}^k \to \{0,1\}$. Then $B \circ e = \mathbf{0}$.*

*Proof.* It can be easily checked that for all $C \in \{0,1\}^\ell$ it is

$$(B \circ e)_C = \frac{\left|\{X \in \{0,1\}^k,\ \beta(C,X) = 0\}\right| - \left|\{X \in \{0,1\}^k,\ \beta(C,X) = 1\}\right|}{2^k} ,$$

which, by definition, equals 0 if $\beta$ is $C$-balanced. □

**Theorem 3.** *Let $r \geq 1$ and $\beta_t$ be either $C$-balanced or constant 0 for all $t$, $1 \leq t \leq r$. Then $\lambda(F^r) = 0$.*

*Proof.* Note that for $\beta_t \equiv 0$, the bias matrix $B_t$ equals the state transition function $P$ of the FSM $\mathcal{C}$. As each row of $P$ corresponds to a probability distribution over $\{0,1\}^\ell$, we obtain $P \circ e = e$. The rest follows straightforwardly from (3) and Lemma 1. □

We want to point out that the previous statements are only true as long as the corresponding input words $X_t$ are independent values in $\{0,1\}^k$. In the case that LFSRs are used as driving devices, this is only the case as long as $r$ is at most the length of the shortest LFSR. For example, in the case of $E_0$, this would mean that $r \leq 25$. This imposes no serious drawback, because so far, no feasible methods are known to compute the bias while considering the LFSR-structure.

The previous results immediately imply two different design criteria to avoid any biased linear combinations in the expressions $\beta(C_t)$. Actually, they have even wider applications. For example, the output function

$$f\left(\left(c^1,c^2,c^3,c^4\right),\left(x^1,x^2,x^3,x^4\right)\right) = c^2 \oplus x^1 \oplus x^2 \oplus x^3 \oplus x^4$$

used in $E_0$ is $C$-balanced. This guarantees that no biased linear combinations of the keystream bits $z_t$ exist for $r \leq 25$, the length of the shortest LFSR.

Golić [11] showed that there is a lower bound $t_0$ such that for all $t \geq t_0$ there are nontrivial correlations between certain XOR-sums of the inputs of $t$ consecutive time units and the corresponding outputs. It would be interesting to investigate if the correlation attack of Lu and Vaudenay [15] can be generalized towards exhibiting this type of correlations.

## 4 Design Principles Against Algebraic Attacks

In this section, we examine the existence of low-degree equations and introduce the first method to guarantee a lower bound for the effort of algebraic attacks. Let $Z \in \{0,1\}^r$ for $r \geq 1$ be fixed. We say that a function $F := \{0,1\}^{k \cdot r} \rightarrow \{0,1\}$ is a $Z$-function if $F \not\equiv 0$ and for each clock $t$, it holds that

$$(z_t,\ldots,z_{t+r-1}) = Z \ \Rightarrow F(X_t,\ldots,X_{t+r-1}) = 0 \ . \quad (4)$$

In [1], $Z$-functions were introduced and their existence was proved. An algebraic attack consists of computing for each $Z \in \{0,1\}^r$ a $Z$-function $F_Z$ of the lowest possible degree and to set up the system of equations

$$F_{(z_t,\ldots,z_{t+r-1})}(X_t,\ldots,X_{t+r-1}) = 0, \quad t = 1,2,\ldots$$

In [1], $Z$-functions of degree 4 for all $Z \in \{0,1\}^4$ were developed for $E_0$. In [6], a method was proposed to obtain equations of degree 3, however with the enormous value $r \approx 8.822.188$. It is still an open question whether $Z$-functions exist of degree $< 4$ and $r \ll 8.822.188$. In this section, we provide a general construction which guarantees a lower bound on the degrees of all possible $Z$-functions if $r$ is at most the length of the shortest LFSR. This is the first general countermeasure against algebraic attacks on combiners with memory proposed so far.

**Definition 4.** *Let $r \geq 1$ be fixed and $Z = (z_1, \ldots, z_r) \in \{0,1\}^r$. We set $X_Z$ to be the set of inputs $(X_1, \ldots, X_r) \in \{0,1\}^{k \cdot r}$ which possibly can produce the output $Z$, more precisely, for which there exist $C_1, \ldots, C_r \in \{0,1\}^\ell$ such that $z_i = f(C_i, X_i)$ for $1 \leq i \leq r$ and $C_{i+1} = \delta(C_i, X_i)\}$ for $1 \leq i \leq r-1$.*

*We will later need the additional property that $C_1$ equals some fixed value $C$ and denote the corresponding subset of $X_Z$ by $X_{C,Z}$.*

With (4), one can easily see that $F$ is a $Z$-function if and only if $F(X) = 0$ for all $X \in X_Z$. This leads directly to the notion of annihilators.

**Definition 5.** *We say that a Boolean function $p \not\equiv 0$ is an annihilator of a subset $A \subseteq \{0,1\}^n$ if $p(x) = 0$ for all $x \in A$. We denote the set of annihilators of $A$ by $Ann(A)$. Furthermore, we define for $A \subset \{0,1\}^n$ $\mathrm{mindeg}(A) := \min\{\deg(f); f \in Ann(A)\}$. If $A = \{0,1\}^n$, we set $\mathrm{mindeg}(A) := \infty$.*

The idea is that if we can prove a lower bound for $\mathrm{mindeg}(X_Z)$ for all $Z$, this gives a lower bound for the effort of algebraic attacks. In the following, we will propose a construction which makes it possible to derive such a lower bound.

We first show that under certain conditions, each "local" lower bound for $\mathrm{mindeg}(X_{z_r,C})$ is also a "global" lower bound for $\mathrm{mindeg}(X_{(z_1,\ldots,z_r)})$.

**Theorem 4.** *For a Boolean function $\alpha : \{0,1\}^k \to \{0,1\}$, let $\mathrm{mindeg}\left(\alpha^{-1}(0)\right)$ and $\mathrm{mindeg}\left(\alpha^{-1}(1)\right)$ both be equal to a value $d$ and let $\beta : \{0,1\}^\ell \to \{0,1\}$. If the output function $f$ can be expressed as*

$$f(C, X) := \alpha(X) \oplus \beta(C) = z \ , \tag{5}$$

*then it holds for all $r \geq 1$, $Z = (z_1, \ldots, z_r) \in \{0,1\}^r$, and $C \in \{0,1\}^\ell$ that*

$$\mathrm{mindeg}(X_Z) \geq \mathrm{mindeg}(X_{C,Z}) = d \ .$$

*Proof.* Because of $X_{Z,C} \subseteq X_Z$, each annihilator of $X_Z$ is an annihilator of $X_{Z,C}$ as well. This shows the first inequality.

Moreover, it holds for all choices $z \in \{0,1\}$ and $C \in \{0,1\}^\ell$ that $X_{C,z} = \alpha^{-1}(\beta(C) \oplus z)$ and therefore $\mathrm{mindeg}(X_{C,z}) = d$.

Let $r \geq 1$, $Z = (z_1, \ldots, z_r) \in \{0,1\}^r$, $C_1 \in \{0,1\}^\ell$ and $f(Y_1) \in \mathbb{F}_2[Y_1]$ be an annihilator of $X_{C_1,z_1}$. Then, $f$ can be seen as an element in $\mathbb{F}_2[Y_1, \ldots, Y_r]$ which annihilates $X_{C_1,Z}$, too. This shows that $\mathrm{mindeg}\left(X_{C_1,Z}\right) \leq \mathrm{mindeg}\left(X_{C_1,z_1}\right) = d$.

We prove now by induction over $r$ that $\mathrm{mindeg}(X_{C_1,Z}) \geq d$ for all choices of $C_1$ and $Z$. For $r = 1$, the claim is certainly true. Now let $r > 1$ and the claim be true for all $r' < r$. Fix $Z = (z_1, \ldots, z_r)$ and $C_1$ and $f(Y_1, \ldots, Y_r) \in Ann(X_{C,Z})$ having the minimal degree $\text{mindeg}(X_{C,Z})$. Choose an arbitrary value $(X_1, \ldots, X_r) \in \{0,1\}^{k \cdot r}$ and set $C_2 := \delta(C_1, X_1)$. Then

$$f^*(Y_2 \ldots, Y_r) := f(X_1, Y_2, \ldots, Y_r)$$

annihilates $X_{C_2,(z_2,\ldots,z_r)}$. Hence,

$$\text{mindeg}\,(X_{C_1,Z}) = \deg(f) \geq \deg(f^*) \geq \text{mindeg}\left(X_{C_2,(z_2,\ldots,z_r)}\right) \geq d \ ,$$

where the last inequality is true by assumption. □

The theorem implies the following strategy. Choose an output function $\alpha$ such that $\text{mindeg}\left(\alpha^{-1}(0)\right)$ and $\text{mindeg}\left(\alpha^{-1}(1)\right)$ is the maximum possible value. This will guarantee the same lower bound for all $Z$-functions, as long the values $X_1, \ldots, X_r$ are independent elements in $\{0,1\}^k$. In the case that they are the outputs of LFSRs, this condition holds for $r$ less than or equal to the length of the shortest LFSR (e.g., 25 in the case of $E_0$). This restriction is not critical, since all currently known methods to derive $Z$-functions become practically infeasible as soon as $r \geq 7$. We emphasize that Theorem 4 yields the only lower bound for algebraic attacks proposed so far. Further on, this approach can be combined with the methods presented in the previous section to achieve maximum resistance against certain correlation attacks.

The value $d$ is equivalently known under the term "algebraic immunity", which was introduced in [16] in the context of memoryless combiners and examined in several papers since then. This means that any proposal for a function with optimum algebraic immunity can be incorporated in our design.

So far, only few proposals for algebraic immune functions have been made (e.g., [8]). A rather straightforward candidate is the majority-function:

**Corollary 1.** *Let $k \geq 1$. For the majority function $maj : \{0,1\}^k \rightarrow \{0,1\}$ defined by*

$$maj(x) = \begin{cases} 0 & \textit{if } weight(x) < k/2 \text{ or } weight(x) = k/2 \text{ and } x_1 = 0 \\ 1 & \textit{otherwise} \end{cases} ,$$

*it holds that* $\text{mindeg}(maj^{-1}(0)) = \text{mindeg}(maj^{-1}(1)) = k/2$.

A proof can be found in [5]. The authors pointed out that *maj* has a very low nonlinearity, making it a bad choice for memoryless combiners. However, this is no problem in the context described above, as long as high biases $\lambda$ are avoided (e.g., using the principles described in Sect. 3).

Using our design principle and a Boolean function with optimum algebraic immunity, it is possible to exclude the existence of $Z$-functions having a degree less than $\lceil k/2 \rceil$. In fact, experiments have shown that the actual values of mindeg are higher, showing that $\lceil k/2 \rceil$ is a rather coarse estimation. Further on, one can easily increase this bound, even without increasing the number of LFSRs by using several different bits per LFSR and clock. For example, in the case of $E_0$, one could use the modified output and update functions $z_t := maj(X_{2t-1}, X_{2t})$ and $C_{t+1} := \delta(\delta(C_t, X_{2t-1}), X_{2t})$. The bitrate is halfed, but the existence of $Z$-functions of degree less than 4 can be excluded.

## 5 Application to $E_0$

In this section, we apply the results from the previous sections to improve the security of the $E_0$ keystream generator. Consequently, we assume that $k = \ell = 4$ and that the keystream bit $z_t$ is computed by $z_t = f(C_t, X_t) = \alpha(X_t) \oplus \beta(C_t)$.

In the case of $E_0$, we have $\alpha(X_t) = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4$ and $\beta(C_t) = c_t^2$. The state transition function of $E_0$ is defined as

$$\delta_0(C_t, X_t) = \left(\mathcal{S}_{t+1}^1 \oplus c_t^1 \oplus c_t^4, \mathcal{S}_{t+1}^0 \oplus c_t^2 \oplus c_t^3 \oplus c_t^4, c_t^1, c_t^2\right) \quad ,$$

where $\mathcal{S}_{t+1} = (\mathcal{S}_{t+1}^1, \mathcal{S}_{t+1}^0) = \left\lfloor \frac{x_t^1 + x_t^2 + x_t^3 + x_t^4 + 2 \cdot c_t^1 + c_t^2}{2} \right\rfloor$ .

Using Theorem 1, we computed the maximum absolute biases over 25 clock cycles (the length of $E_0$'s shortest LFSR) for all 16 $E_0$-variants where the function $\beta$ is of the form $\beta_{(a_1,a_2,a_3,a_4)}(C_t) = a_1 \cdot c_t^1 \oplus a_2 \cdot c_t^2 \oplus a_3 \cdot c_t^3 \oplus a_4 \cdot c_t^4$ for $a = (a_1, a_2, a_3, a_4) \in \{0,1\}^4$. Note that the original $\beta$ function of $E_0$ corresponds to $\beta_{(0,1,0,0)}$. As shown in Table 3, the minimum absolute bias $\lambda = 0.024414$ is obtained for $a = (0,1,1,1)$. We denote the corresponding generator by $E_0^1$. Unfortunately, there exist $Z$-functions of degree 3 for $E_0^1$, which makes it weaker against algebraic attacks than the original $E_0$. However, choosing the $a$-value with the second best minimum absolute bias (we call the corresponding generator $E_0^2$) yields mindeg $= 6$.

In the next step, we exploit our theory to completely avoid biases. Starting from the definition of $E_0$, we obtain the generator $E_0^3$ by replacing the state transition function by $\delta_1$, which is defined as the integer addition modulo $2^4$, i.e.,

$$\delta_1\left(c_t^1, \ldots, c_t^4, x_t^1, \ldots, x_t^4\right) = \left(\sum_{i=0}^{3} c_t^{4-i} 2^i + \sum_{j=0}^{3} x_t^{4-j} 2^j\right) \mod 16 \ .$$

Since $\delta_1$ is balanced, Theorem 2 implies $\lambda = 0$, i.e., $E_0^3$ is immune against the approach used in [14]. However, we computed $Z$-functions of degree 3 for $E_0^3$. We therefore replace the function $\alpha$ of $E_0^3$ by the majority function described in Corollary 1. For the resulting generator $E_0^4$, we obtain mindeg $= 5$. If we additionally replace the function $\beta$ by the majority function, mindeg even increases to 6. Note that the $\lambda = 0$ property is still preserved by these modifications. Thus, we obtain a keystream generator $E_0^5$ with $\lambda_{\max} = 0$ , and whose resistance against algebraic attacks is significantly increased compared to $E_0$. Notice that in all cases, the values of mindeg were actually higher than the theoretical lower bound of $\lceil k/2 \rceil = 2$. The constructions of the considered generators and the respective performances of algebraic and correlation attacks are summarized in Table 4. The average numbers of $Z$-functions for the generators are given in

**Table 3.** Maximum absolute biases and efficiency of correlation attacks for $\beta_a$-generators

| $a$ | max $(\lambda)$ | Frames | Data | Time | Space |
|---|---|---|---|---|---|
| $(0,0,0,1)$ | 0.097656 | $2^{34.74}$ | $2^{39.32}$ | $2^{40.17}$ | $2^{34.74}$ |
| $(0,0,1,0)$ | 0.244141 | $2^{24.16}$ | $2^{28.74}$ | $2^{37.59}$ | $2^{24.16}$ |
| $(0,0,1,1)$ | 0.156250 | $2^{29.31}$ | $2^{33.90}$ | $2^{37.74}$ | $2^{29.31}$ |
| $(0,1,0,0)$ | **0.097656** | $\mathbf{2^{34.74}}$ | $\mathbf{2^{39.32}}$ | $\mathbf{2^{40.17}}$ | $\mathbf{2^{34.74}}$ |
| $(0,1,0,1)$ | 0.097656 | $2^{34.74}$ | $2^{39.32}$ | $2^{40.17}$ | $2^{34.74}$ |
| $(0,1,1,0)$ | 0.156250 | $2^{29.31}$ | $2^{33.90}$ | $2^{37.74}$ | $2^{29.31}$ |
| $(0,1,1,1)$ | **0.024414** | $\mathbf{2^{53.56}}$ | $\mathbf{2^{58.15}}$ | $\mathbf{2^{58.73}}$ | $\mathbf{2^{53.56}}$ |
| $(1,0,0,0)$ | 0.244141 | $2^{24.16}$ | $2^{28.74}$ | $2^{37.59}$ | $2^{24.16}$ |
| $(1,0,0,1)$ | 0.250000 | $2^{23.89}$ | $2^{28.47}$ | $2^{37.59}$ | $2^{23.89}$ |
| $(1,0,1,0)$ | 0.097656 | $2^{34.74}$ | $2^{39.32}$ | $2^{40.17}$ | $2^{34.74}$ |
| $(1,0,1,1)$ | **0.038528** | $\mathbf{2^{46.98}}$ | $\mathbf{2^{51.56}}$ | $\mathbf{2^{52.15}}$ | $\mathbf{2^{46.98}}$ |
| $(1,1,0,0)$ | 0.156250 | $2^{29.31}$ | $2^{33.90}$ | $2^{37.74}$ | $2^{29.31}$ |
| $(1,1,0,1)$ | 0.156250 | $2^{29.31}$ | $2^{33.90}$ | $2^{37.74}$ | $2^{29.31}$ |
| $(1,1,1,0)$ | 0.152588 | $2^{29.58}$ | $2^{34.17}$ | $2^{37.77}$ | $2^{29.58}$ |
| $(1,1,1,1)$ | 0.097656 | $2^{34.74}$ | $2^{39.32}$ | $2^{40.17}$ | $2^{34.74}$ |

**Table 4.** Definitions of the candidate generators and efficiencies of algebraic and correlation attacks

| | | | | Algebraic Attack | | Correlation Attack | |
|---|---|---|---|---|---|---|---|
| | $\delta$ | $\alpha\left(x_t^1,x_t^2,x_t^3,x_t^4\right)$ | $\beta\left(c_t^1,c_t^2,c_t^3,c_t^4\right)$ | mindeg | Time | $\lambda$ | Time |
| $E_0$ | $\delta_0$ | $x_t^1\oplus x_t^2\oplus x_t^3\oplus x_t^4$ | $c_t^2$ | 4 | $2^{70.18}$ | 0.097656 | $2^{40.17}$ |
| $E_0^1$ | $\delta_0$ | $x_t^1\oplus x_t^2\oplus x_t^3\oplus x_t^4$ | $c_t^2\oplus c_t^3\oplus c_t^4$ | 3 | $2^{55.25}$ | 0.024414 | $2^{58.73}$ |
| $E_0^2$ | $\delta_0$ | $x_t^1\oplus x_t^2\oplus x_t^3\oplus x_t^4$ | $c_t^1\oplus c_t^3\oplus c_t^4$ | 6 | $2^{97.22}$ | 0.038528 | $2^{52.15}$ |
| $E_0^3$ | $\delta_1$ | $x_t^1\oplus x_t^2\oplus x_t^3\oplus x_t^4$ | $c_t^2$ | 3 | $2^{55.25}$ | 0 | n/a |
| $E_0^4$ | $\delta_1$ | $maj(x_t^1,x_t^2,x_t^3,x_t^4)$ | $c_t^2$ | 5 | $2^{84.11}$ | 0 | n/a |
| $E_0^5$ | $\delta_1$ | $maj(x_t^1,x_t^2,x_t^3,x_t^4)$ | $maj(c_t^1,c_t^2,c_t^3,c_t^4)$ | 6 | $2^{97.22}$ | 0 | n/a |

Appendix B. We note that the generator $E_0^2$, which is just a slight modification of $E_0$ (we only made $\beta$ depend on two more state bits), already yields a similar resistance against algebraic attacks as $E_0^5$ and significantly decreases the vulnerability against correlation attacks. This desirable property – small changes to the cipher leading to great security improvements – particularly recommends this construction for the 3-year roadmap for enhancing the Bluetooth system [20] that was recently announced by the Bluetooth SIG and so far only includes moderate security enhancements.

## 6 Conclusion

In this paper, we developed design principles to counter the most effective attacks today, namely correlation and algebraic attacks. Applied to $E_0$, we can show that small modifications already suffice to improve the security significantly.

## References

1. Armknecht, Krause: *Algebraic Attacks on Combiners with Memory*, Crypto 2003, LNCS 2729, pp. 162-176, Springer, 2003
2. Armknecht: *Improving Fast Algebraic Attacks*, Fast Software Encryption 2004, LNCS 3017, pp. 65 - 82, Springer, 2004
3. Armknecht, Lano, Preneel: *Extending the Resynchronization attack*, SAC 2004, LNCS 3357, pp. 19-38, Springer, 2004
4. Armknecht: *On the Existence of Low-Degree Equations for Algebraic Attacks*, Cryptology ePrint Archive: Report 2004/185
5. An Braeken, Joe Lano: *On the (im)possibiliy of practical and secure nonlinear filters and combiners*, SAC 2005 (to appear)
6. Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback*, Crypto 2003, LNCS 2729, pp. 177-194, Springer, 2003
7. Courtois, Klimov, Patarin, Shamir: *Efficient Algoprithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt 2000, LNCS 1807, pp. 392-407, Springer 2000
8. Dalai, Gupta, Maitra: *Cryptographically Significant Boolean Functions: Construction and Analysis in Terms of Algebraic Immunity*, FSE 2005
9. Jean-Charles Faugère, Gwenole Ars: *An Algebraic Cryptanalysis of Nonlinear Filter Generators using Gröbner Bases*, 2003. Available at http://www.inria.fr/rrrt/rr-4739.html
10. Golić: *Correlation via Linear Sequential Circuit Approximation of Combiners with Memory*, Eurocrypt 1993, LNCS 658, pp. 113-123, Springer, 1993
11. Golić: *Correlation Properties of General Binary Combiners with Memory*. Journal of Cryptology 9(2), pp. 111-126, 1996
12. Hawkes, Rose: *Rewriting Variables: the Complexity of Fast Algebraic Attacks on Stream Ciphers*, Crypto 2004, LNCS 3152, pp. 390 - 406, Springer, 2004. Available at http://eprint.iacr.org/2004/081/
13. Key, McDonough, Mavron: *Information sets and partial permutation decoding for codes from finite geometries*, Preprint Clemson Univ., to appear in Finite Field Applic., 2005
14. Lu, Vaudenay: *Faster Correlation Attack on the Bluetooth Keystream Generator*, Crypto 2004, LNCS 3152, pp. 407-425, Springer, 2004
15. Lu, Vaudenay: *Cryptanalysis of Bluetooth Keystream Generator Two-Level E0*, Asiacrypt 2004, LNCS 3329, pp. 483-499, Springer, 2004
16. Meier, Pasalic, Carlet: *Algebraic Attacks and Decomposition of Boolean Functions*, Proceedings of Eurocrypt 2004, LNCS 3027, pp. 474-491, Springer, 2004
17. Rueppel: *Analysis and Design of Stream Ciphers*. Springer, 1986
18. Salmasizadeh, Golić, Dawson, Simpson: *A Systematic Procedure for Applying Fast Correlation Attacks to Combiners with Memory*, SAC 1997
19. Siegenthaler: *Correlation Immunity of Nonlinear Combining Functions for Cryptographic Applications*. IEEE Inform. Theory, IT-30, pp. 776-780, 1984
20. The Bluetooth SIG: Bluetooth SIG Lays Out Roadmap for Bluetooth Wireless Technology, SIG press release, November 8, 2004

## A Proof of Theorem 1

We have to prove that for all $r \geq 1$ it holds that $\lambda(F^r) = 2^{-\ell}(e^T) \circ B_1 \circ \cdots \circ B_r \circ e$, where $F^r : \{0,1\}^\ell \times \left(\{0,1\}^k\right)^r \to \{0,1\}$ is for all $C_1 \in \{0,1\}^\ell$ and $X_1, \cdots, X_r \in$

$\{0,1\}^k$ defined by $F^r(C_1, X_1, \cdots, X_r) = \beta_1(C_1, X_1) \oplus \cdots \oplus \beta_r(C_r, X_r)$, with $C_{t+1} = \delta(C_t, X_t)$.

We start with some basics on computing biases. For given finite set $S$ and real functions $f, g : S \to \mathbb{R}$ we denote by $(f, g) = \frac{1}{|S|}\sum_{s\in S} f(s)g(s)$ a positive definite scalar product on $\mathbb{R}^S$. Note that for each Boolean function $f : \{0,1\}^n \to \{0,1\}$ it holds that $\lambda(f) = \big((-1)^f, 1\big)$.

We call a Boolean function $f : \{0,1\}^n \to \{0,1\}$ unbiased, if $\lambda(f) = 0$, i.e., if $\big|f^{-1}(1)\big| = \big|f^{-1}(0)\big|$.

Consider two disjoint finite sets $S$ and $S'$, functions $f : S \to \mathbb{R}$ and $g : S' \to \mathbb{R}$, and let $h : S \times S' \to \mathbb{R}$ be defined by $h(s, s') = f(s)g(s')$. Note that

$$(h, 1) = \frac{1}{|S||S'|}\sum_{s\in S, s'\in S'} f(s)g(s') = \frac{1}{|S|}\sum_{s\in S} f(s)\frac{1}{|S'|}\sum_{s'\in S'} g(s') = (f, 1)(g, 1) \ .$$

This implies that for Boolean functions $f : \{0,1\}^n \to \{0,1\}$, $g : \{0,1\}^m \to \{0,1\}$, and $h : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$, defined by $h(s, s') = f(s) \oplus g(s')$, it holds that $\lambda(h) = \lambda(f) \cdot \lambda(g)$.

Now let us denote by $f^r : \{0,1\}^\ell \times \big(\{0,1\}^k\big)^r \to \mathbb{R}$ the function $(-1)^{F^r}$.

For all $r \geq 1$ and $C \in \{0,1\}^\ell$, we define an additional function $f^r_C : \{0,1\}^\ell \times \big(\{0,1\}^k\big)^r \to \mathbb{R}$ as follows. For all $C_1, X_1, \cdots, X_r \in \{0,1\}^\ell$ let

$$f^r_C(C_1, X_1, \cdots, X_r) = \begin{cases} f^r(C_1, X_1, \cdots, X_r) & \text{if } \delta(C_r, X_r) = C \\ 0 & \text{otherwise} \end{cases} \ .$$

The intermediate states $C_2, \cdots, C_r$ are defined via the rule $C_{i+1} = \delta(C_t, X_t)$. Note that $f^r_C$ maps into $\{-1, 0, 1\}$. Let $\Gamma^r_C = (f^r_C, 1)$ and $\Gamma^r = (\Gamma^r_C)_{C\in\{0,1\}^\ell}$. Observe that $f^r = \sum_{C\in\{0,1\}^\ell} f^r_C$ and $\lambda(F^r) = \sum_{C\in\{0,1\}^\ell} \Gamma^r_C$. Theorem 1 is a straightforward consequence of the following lemma.

**Lemma 2.** *For $C \in \{0,1\}^\ell$ and $r \geq 1$ it holds that $(\Gamma^r)^T = 2^{-\ell}(e^T)\circ B_1 \circ \cdots \circ B_r$, where $B_t$ denote the bias matrices as defined in Definition 1.*

*Proof.* For all $C, C' \in \{0,1\}^\ell$ we denote by $g_{C,C'} : \{0,1\}^k \to \{0,1\}$ the Boolean function defined by $g_{C,C'}(X) = 1$ iff $\delta(C, X) = C'$. Observe that for each $t \geq 1$ it holds that

$$((-1)^{\beta_t(C,\cdot)} g_{C,C'}, 1) = b_t(C, C') \ . \qquad (6)$$

We show the lemma by induction on $r$. Note that, due to (6), for all $C \in \{0,1\}^\ell$

$$\Gamma^1_C = 2^{-(k+\ell)}\sum_{C_1, X_1} (-1)^{\beta_1(C_1,X_1)} g_{C_1,C}(X_1) = 2^{-\ell}\sum_{C_1} b_t(C_1, C) \ .$$

Consequently, $\Gamma^1 = 2^{-\ell}(e^T) \circ B_1$. For $r > 1$, the function $f^r_C$ can be written as

$$f^r_C(C_1, X_1, \cdots, X_r) = \sum_{C'\in\{0,1\}^\ell} f^{r-1}_{C'}(C_1, X_1, \cdots, X_{r-1})(-1)^{\beta_r(C',X_r)} g_{C',C}(X_r) \ .$$

Hence, by the remarks in the last subsection and (6), we obtain

$$\Gamma^r_C = \sum_{C'\in\{0,1\}^\ell} \Gamma^{r-1}_{C'} b_r(C', C) \quad \textit{and} \quad (\Gamma^r)^T = (\Gamma^{r-1})^T \circ B_r \ . \qquad \square$$

## B *Z*-Functions for the Considered Generators

For all variants of $E_0$ that were considered in Sect. 5, Table 5 lists the minimum degree and the respective number of $Z$-functions over $r$ clock cycles. For Example, for $E_0^4$, the minimum degree of $Z$-functions over up to 5 clock cycles is 5, and there are 40, 264, 896, and 2528 $Z$-functions over $2, 3, 4$ and 5 clock cycles, respectively.

The computation of the number of $Z$-functions over 6 clocks for $E_0^5$ was still in progress when we finalized this paper. Since in all our experiments, the minimum degree of the $Z$-functions never decreased with increasing $r$, we suspect that mindeg = 6 will also hold for $E_0^5$ and $r = 6$.

**Table 5.** mindeg and number of $Z$-functions for the candidate generators

| Cipher | $E_0$ | $E_0^1$ | $E_0^2$ | $E_0^3$ | $E_0^4$ | $E_0^5$ |
|---|---|---|---|---|---|---|
| mindeg | 4 | 3 | 6 | 3 | 5 | 6 |
| Clocks | Number of equations | | | | | |
| $r = 2$ | 0 | 12 | 0 | 4 | 40 | 12 |
| $r = 3$ | 0 | 48 | 24 | 40 | 264 | 318 |
| $r = 4$ | 16 | 144 | 160 | 144 | 896 | 1416 |
| $r = 5$ | 64 | 384 | 544 | 416 | 2528 | $> 0$ |
| $r = 6$ | 192 | ? | $> 0$ | ? | ? | ? |

# Cryptanalysis of the Quadratic Generator

Domingo Gomez, Jaime Gutierrez, and Alvar Ibeas

Faculty of Sciences,
University of Cantabria,
Santander E–39071, Spain
jaime.gutierrez@unican.es

**Abstract.** Let $p$ be a prime and let $a$ and $c$ be integers modulo $p$. The quadratic congruential generator (QCG) is a sequence $(v_n)$ of pseudorandom numbers defined by the relation $v_{n+1} \equiv av_n^2 + c \bmod p$. We show that if sufficiently many of the most significant bits of several consecutive values $v_n$ of the QCG are given, one can recover in polynomial time the initial value $v_0$ (even in the case where the coefficient $c$ is unknown), provided that the initial value $v_0$ does not lie in a certain small subset of exceptional values.

## 1 Introduction

For a prime $p$, denote by $\mathbb{F}_p$ the field of $p$ elements and always assume that it is represented by the set $\{0, 1, \ldots, p-1\}$. Accordingly, sometimes, where obvious, we treat elements of $\mathbb{F}_p$ as integer numbers in the above range.

For fixed $a \in \mathbb{F}_p^*$, $c \in \mathbb{F}_p$, the *quadratic generator* $(v_n)$ of elements of $\mathbb{F}_p$ is given by the recurrence relation

$$v_{n+1} \equiv av_n^2 + c \bmod p \qquad n = 0, 1, \ldots, \tag{1}$$

where $v_0$ is the *initial value*.

We refer to the coefficients $a$ and $c$ as the *multiplier* and *shift*, respectively. This generator has many interesting applications in cryptography, see [4, 14, 15, 16, 17, 9].

In the cryptographic setting, the initial value $v_0$ and the constants $a$ and $c$ are assumed to be the secret key, and we want to use the output of the generator as a stream cipher. Of course, if several consecutive values $v_n$ are revealed, it is easy to find $v_0$, $a$ and $c$. So in this setting, we output only the most significant bits of each $v_n$ in the hope that this makes the resulting output sequence difficult to predict. The paper [3], shows that not too many bits can be output at each stage: the quadratic generator is unfortunately polynomial time predictable if sufficiently many bits of its consecutive elements are revealed, so long as a small number of secret keys are excluded. However, some of the results in that paper only hold after excluding a small set of $a$, see [3–Theorem 3]. If this small set is not excluded, the algorithm for finding the secret information may fail. An optimist might hope that by deliberately choosing $a$ to lie in this excluded set, one

© Springer-Verlag Berlin Heidelberg 2005

can generate cryptographically stronger sequences. This paper aims to show that this strategy is unlikely to succeed. Namely we introduce some modifications and additions to the method of [3] which allow us to attack the generators no matter how the value of $a$ is chosen. In fact, our idea is similar to the approach in paper [2]. We demonstrate our approach in the special cases when $a$ and $c$ are public and when $c$ is secret and $a$ is public. This last case was not considered in the mentioned paper. But we believe that the extra strength of the result we obtain makes this situation of interest in its own right. We also believe this approach can be extended to the case when both $a$ and $c$ are secret [3–Theorem 5].

Assume that the sequence $(v_n)$ is not known but, for some $n$, some approximations $w_j$ are given. We show that if $a$ and $c$ are public or if $a$ is public and $c$ secret the values $v_{n+j}$ and $a$ can be recovered from this information in polynomial time if the approximations $w_j$ are sufficiently good and if a certain small set of initial values $v_0$ are excluded. (The results in [3] exclude a small set of $a$ in addition to values of $v_0$, and so in this sense our result here is stronger.)

The remainder of the paper is structured as follows.

We start with a short outline of some basic facts about lattices in Section 2.1 and polynomial in congruences Section 2.2. In Section 3 we consider the cases of quadratic generator with known multiplier and shift in Subsection 3.1 and with known multiplier and unknown shift in Subsection 3.2. Finally, Section 4 makes some final comments and poses several open questions.

## 2 Lattices and Polynomials

### 2.1 Background on Lattices

Here we collect several well-known facts about lattices which form the background to our algorithms.

We review several related results and definitions on lattices which can be found in [5]. For more details and more recent references, we also recommend consulting [1, 6, 7, 11, 12, 13].

Let $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$ be a set of linearly independent vectors in $\mathbb{R}^r$. The set

$$\mathcal{L} = \{\boldsymbol{z} \ : \ \boldsymbol{z} = c_1\boldsymbol{b}_1 + \ldots + c_s\boldsymbol{b}_s, \quad c_1, \ldots, c_s \in \mathbb{Z}\}$$

is called an *s-dimensional lattice* with *basis* $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$. If $s = r$, the lattice $L$ is of *full rank*.

To each lattice $\mathcal{L}$ one can naturally associate its *volume*

$$\mathrm{vol}\,(\mathcal{L}) = \left(\det\left(\langle \boldsymbol{b}_i, \boldsymbol{b}_j\rangle\right)_{i,j=1}^{s}\right)^{1/2},$$

where $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$ denotes the inner product. This definition does not depend on the choice of the basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_s\}$.

For a vector $\boldsymbol{u}$, let $\|\boldsymbol{u}\|$ denote its *Euclidean norm*. The famous Minkowski theorem, see Theorem 5.3.6 in Section 5.3 of [5], gives the upper bound

$$\min\{\|\boldsymbol{z}\| : \ \boldsymbol{z} \in \mathcal{L} \setminus \{\boldsymbol{0}\}\} \le s^{1/2}\,\mathrm{vol}\,(\mathcal{L})^{1/s} \tag{2}$$

on the shortest nonzero vector in any $s$-dimensional lattice $\mathcal{L}$ in terms of its volume. In fact, $s^{1/2}$ can be replaced by the *Hermite constant* $\gamma_s^{1/2}$, for which we have

$$\frac{1}{2\pi e}s + o(s) \le \gamma_s \le \frac{1.744}{2\pi e}s + o(s), \qquad s \to \infty.$$

The Minkowski bound (2) motivates a natural question: how to find the shortest vector in a lattice. The celebrated *LLL algorithm* of Lenstra, Lenstra and Lovász [10] provides a desirable solution in practice, and the problem is known to be solvable in deterministic polynomial time (polynomial in the bit-size of the basis of $\mathcal{L}$), provided that the dimension of $\mathcal{L}$ is fixed (see Kannan [8–Section 3], for example). The lattices in this paper are of fixed dimension. (Note that there are several indications that the shortest vector problem is **NP**-complete when the dimension grows.)

In fact, in this paper we consider only very special lattices. Namely, only lattices which are consisting of integer solutions $\boldsymbol{x} = (x_0, \ldots, x_{s-1}) \in \mathbb{Z}^s$ of the system of congruences

$$\sum_{i=0}^{s-1} a_{ij}x_i \equiv 0 \bmod q_j, \qquad j = 1, \ldots, m,$$

modulo some integers $q_1, \ldots, q_m$. Typically (although not always) the volume of such a lattice is the product $Q = q_1 \ldots q_m$. Moreover, all the aforementioned algorithms, when applied to such a lattice, become polynomial in $\log Q$.

## 2.2 Polynomial Congruences

Our second basic tool is essentially the theorem of Lagrange which asserts that a non-zero univariate polynomial of degree $N$ over any field has no more than $N$ zeros in this field.

The polynomials we consider belong to a certain family of functions parametrised by small vectors in a certain lattice, thus the size of the family can be kept under control.

Now we present a technical result for later use. The following lemma is an adaptation of an argument in the paper [2]:

**Lemma 1.** *Let $p > 5$ be a prime and let $\alpha$ be a nonzero integer modulo $p$. Then the bivariate congruence*

$$\alpha x \equiv y \bmod p,$$

*with $\gcd(x, y) = 1$, $|x| < p^{1/3}$ and $|y| < p^{1/3}$ has at most two integer solutions $(x, y)$.*

*Proof.* Suppose that $(x, y)$ and $(x', y')$ are two solutions. Then $xy' \equiv yx' \bmod p$ and since both $xy'$ and $yx'$ have absolute value at most $p^{2/3}$ we find that $xy' = yx'$. But since $\gcd(x, y) = \gcd(x', y') = 1$ we now obtain the thesis.

## 3 Predicting the Quadratic Generator

Throughout the paper the term polynomial time means polynomial in $\log p$. Our results involve another parameter $\Delta$ which measures how well the values $w_j$ approximate the terms $v_{n+j}$. This parameter is assumed to vary independently of $p$ subject to satisfying the inequality $\Delta < p$ (and is not involved in the complexity estimates of our algorithms.)

More precisely, we say that $w$ is a *$\Delta$-approximation* to $u$ if $|w-u| \le \Delta$. In all of our results, the case where $\Delta$ grows like a fixed power $p^\delta$ where $0 < \delta < 1$ corresponds to the situation where a positive proportion $\delta$ of the least significant bits of terms of the output sequence remain hidden.

To simplify the notation, we assume that $n = 0$ from now on.

### 3.1 Predicting the Quadratic Generator with Known Multiplier and Shift

We can formulate the main result in this subsection.

**Theorem 1.** *Let $p$ be a prime number and let $\Delta$ be an integer such that $p > \Delta \ge 1$. For any $a \in \mathbb{F}_p^*$ and $c \in \mathbb{F}_p$, there exists a set $\mathcal{U}(\Delta; a, c) \subseteq \mathbb{F}_p$ of cardinality $\#\mathcal{U}(\Delta; a, c) = O(\Delta^4)$ with the following property. There exists an algorithm which, when given $a$, $c$ and $\Delta$-approximations $w_0, w_1$ to two consecutive values $v_0, v_1$ produced by the quadratic generator* (1)*, where $v_0 \notin \mathcal{U}(\Delta; a, c)$, returns the value of $v_0$ in deterministic polynomial time.*

*Proof.* The theorem is trivial when $\Delta^4 \ge p$ and we assume that $\Delta^4 < p$. We fix $a, c \in \mathbb{F}_p$ and we assume that $v_0 \in \mathbb{F}_p$ is chosen so as not to lie in a certain subset $\mathcal{U}(\Delta; a, c)$ of $\mathbb{F}_p^*$ of cardinality $O(\Delta^4)$. As its definition is fairly complicated we define it gradually.

Let $\varepsilon_j := v_j - w_j$, $j = 0, 1$. From $v_1 \equiv a v_0^2 + c \bmod p$, we obtain

$$w_1 + \varepsilon_1 - a(w_0 + \varepsilon_0)^2 - c \equiv 0 \bmod p.$$

Writing

$$A \equiv (w_1 - a w_0^2 - c) \bmod p, \qquad B_1 \equiv -2 a w_0 \Delta \bmod p,$$
$$B_2 \equiv \Delta \bmod p, \qquad C \equiv -a\Delta^2 \bmod p,$$

we obtain

$$A\Delta^2 + B_1 \Delta \varepsilon_0 + B_2 \Delta \varepsilon_1 + C \varepsilon_0^2 \equiv 0 \bmod p. \tag{3}$$

Therefore the lattice $\mathcal{L}$ consisting of integer solutions

$$\boldsymbol{x} = (x_0, x_1, x_2, x_3) \in \mathbb{Z}^4$$

of the system of congruences

$$\begin{aligned} A x_0 + B_1 x_1 + B_2 x_2 + C x_3 &\equiv 0 \bmod p,\\ x_0 &\equiv 0 \bmod \Delta^2,\\ x_1 \equiv x_2 &\equiv 0 \bmod \Delta, \end{aligned}$$

contains a vector

$$\boldsymbol{e} = \left(\varDelta^2 e_0, \varDelta e_1, \varDelta e_2, e_3\right) = \left(\varDelta^2, \varDelta\varepsilon_0, \varDelta\varepsilon_1, \varepsilon_0^2\right).$$

We have

$$e_0 = 1, \qquad |e_1|, |e_2| \le \varDelta, \qquad |e_3| \le \varDelta^2,$$

thus

$$\|\boldsymbol{e}\| \le \left(4\varDelta^4\right)^{1/2} = 2\varDelta^2.$$

Let $\boldsymbol{f} = \left(\varDelta^2 f_0, \varDelta f_1, \varDelta f_2, f_3\right)$ be a shortest nonzero vector in $\mathcal{L}$. So, $\|\boldsymbol{f}\| \le \|\boldsymbol{e}\| \le 2\varDelta^2$. We have

$$|f_0| \le 2, \qquad |f_1|, |f_2| \le 2\varDelta, \qquad |f_3| \le 2\varDelta^2.$$

Note that we may compute $\boldsymbol{f}$ in polynomial time from the information we are given. The vector $\boldsymbol{d} = f_0\boldsymbol{e} - e_0\boldsymbol{f} = (0, \varDelta d_1, \varDelta d_2, d_3) \in \mathcal{L}$ and has first component 0. We might hope that $\boldsymbol{e}$ and $\boldsymbol{f}$ are always parallel. If not, we claim that $d_1 = 0$ unless $v_0$ belongs to the set $\mathcal{V}(\varDelta; a, c)$ which we define below.

Using the definition of $\mathcal{L}$, we find that

$$-2aw_0d_1 + d_2 - ad_3 \equiv 0 \bmod p, \tag{4}$$

where $d_i = e_if_0 - f_i$, and thus $|d_i| \le 2|e_i| + |f_i|$ for $i = 1, 2, 3$. Hence

$$|d_1|, |d_2| \le 4\varDelta, \qquad |d_3| \le 4\varDelta^2. \tag{5}$$

Substituting $w_i = v_i - \varepsilon_i$, $i = 0, 1$, into the congruence (4), we find the following congruence

$$-2ad_1v_0 \equiv E \bmod p,$$

where

$$E = a\left(-2d_1\varepsilon_0 + d_3\right) - d_2.$$

We define $\mathcal{U}(\varDelta; a, c)$ as the set a values $v_0$ that satisfy some congruence of the form (4) with $d_1 \not\equiv 0 \bmod p$. The bounds (5) imply that $d_1$ can take only $O(\varDelta)$ distinct values. Moreover, $E$ can take $O(\varDelta^3)$ distinct values (because $2d_1\varepsilon_0 - d_3 = O(\varDelta^2)$ and $d_2 = O(\varDelta)$). Since $d_1 \not\equiv 0 \bmod p$, this means that $\#\mathcal{U}(\varDelta; a, c) = O(\varDelta^4)$.

So, we can assume that $v_0 \notin \mathcal{U}(\varDelta; a, c)$. The bound (5) on $|d_1|$ and this assumption imply

$$d_1 = 0 \quad \text{and} \quad -d_2 + ad_3 \equiv 0 \bmod p.$$

We distinguish two cases: $f_0 \ne 0$ and $f_0 = 0$ and analyze them separately.

*Predicting the generator when $f_0 \ne 0$.* Since $d_1 = 0$ we have $f_0\varepsilon_0 - f_1 \equiv 0 \bmod p$. The bound on $|f_1|$ shows that $\varepsilon_0 = f_1/f_0$ and so we may compute the secret information $\varepsilon_0$.

*Predicting the generator when $f_0 = 0$.* In this case we have $\boldsymbol{d} = \boldsymbol{f} = (0, 0, \Delta f_2, f_3)$ verifying $f_2 \equiv af_3 \bmod p$. It is easy to see that $f_3 \not\equiv 0 \bmod p$. Otherwise would contradict the fact that $\boldsymbol{f}$ is a nonzero vector. Hence $f_3 a \equiv f_2 \bmod p$ and so we may write

$$sa \equiv r \bmod p, \text{ where } r = f_2/\gcd(f_2, f_3) \text{ and } s = f_3/\gcd(f_2, f_3).$$

Note that $r$ and $s$ are coprime,

$$|r| \le 2\Delta, \qquad |s| \le 2\Delta^2. \tag{6}$$

Moreover we know explicitly $r$ and $s$ since we have computed $\boldsymbol{f}$.

From the congruence (3) we derive

$$\underbrace{rw_0^2 - sw_1 + sc} + \underbrace{2rw_0}\,\varepsilon_0 - s\varepsilon_1 + r\varepsilon_0^2 \equiv 0 \bmod p$$

We now consider a new lattice: the lattice $\mathcal{L}'$ consisting of solutions $\boldsymbol{x} = (x_0, x_1, x_2) \in \mathbb{Z}^3$ of the congruences

$$\mathcal{L}' : \begin{cases} (rw_0^2 + sc - sw_1)\Delta^{-3}x_0 + 2rw_0\Delta^{-2}x_1 + x_2 \equiv 0 \bmod p \\ x_0 \equiv 0 \bmod \Delta^3 \\ x_1 \equiv 0 \bmod \Delta^2 \end{cases} \tag{7}$$

It is easy to check that the lattice (7) contains the vector

$$\boldsymbol{e'} = \left(\Delta^3, \Delta^2\varepsilon_0, r\varepsilon_0^2 - s\varepsilon_1)\right).$$

Thus the Euclidean norm $\|\boldsymbol{e'}\|$ of $\boldsymbol{e'}$ satisfies the inequality

$$\|\boldsymbol{e'}\| \le \sqrt{\Delta^6 + \Delta^6 + 16\Delta^6} = 3\sqrt{2}\Delta^3.$$

Again, we now show that all short vectors in $\mathcal{L}'$ are parallel to $\boldsymbol{e'}$ unless $v_0$ belongs to the set $\mathcal{V}'(\Delta; a, b)$ which we define below.

Assume, for a contradiction, that there is another vector

$$\boldsymbol{f'} = (\Delta^3 f_0', \Delta^2 f_1', f_2') \in \mathcal{L}'$$

with $\|\boldsymbol{f'}\| \le \|\boldsymbol{e'}\| \le 3\sqrt{2}\Delta^3$ which is not parallel to $\boldsymbol{e'}$. The vector $\boldsymbol{d'} \in \mathcal{L}'$ defined by

$$\boldsymbol{d}' = \boldsymbol{f}' - f_0'\boldsymbol{e}' = (0, \Delta^2 d_1', d_2').$$

verifies:

$$|d_1'| \le 9\Delta, \qquad |d_2'| \le 21\Delta^3. \tag{8}$$

Using the first congruence in (7), we find that

$$2rw_0 d_1' + d_2' \equiv 0 \bmod p. \tag{9}$$

If $d_1' \equiv 0 \bmod p$, then using bounds (8) we obtain $d_1' = d_2' = 0$. This implies that vectors $\boldsymbol{e}'$ and $\boldsymbol{f}'$ are parallel which it is a contradiction.

Substituting $w_0 = v_0 - \varepsilon_0$ in the congruence (9)

$$2rv_0d'_1 \equiv E' \bmod p, \tag{10}$$

where $E' \equiv 2r\varepsilon_0 d'_1 - d'_2 \bmod p$. We define $\mathcal{V}'(\Delta; a, c)$ the set of values $v_0$ that satisfy some congruence of the form (10) with $d'_1 \not\equiv 0 \bmod p$. Since $d'_1 \not\equiv 0 \bmod p$, the congruence (10) can be satisfied for at most 1 values of $v_0$ once $r$, $d'_1$ and $E'$ have been chosen. By bounds (6) we can apply Lemma 1 then there are at most 2 choices for $r$. There are $O(\Delta^3)$ choices for $E'$ since $|E'| \le 42\Delta^3$. Hence there are only $O(\Delta^4)$ values of $v_0$ that satisfy some congruence of the form (10) where the $d'_i$ and $E'$ satisfy the appropriate bounds. This means that $\#\mathcal{V}'(\Delta; a, c) = O(\Delta^4)$. So all short vectors in $\mathcal{L}'$ are parallel to $\boldsymbol{e'}$ whenever $v_0 \notin \mathcal{V}'(\Delta; a, b)$.

Finally, we apply a deterministic polynomial time algorithm for the shortest vector problem in a finite dimensional lattice to find a shortest nonzero vector $\boldsymbol{f'}$ in $\mathcal{L}'$, and this vector must be parallel to $\boldsymbol{e'}$. We recover $\boldsymbol{e'}$ by using the fact that $\boldsymbol{e'} = \boldsymbol{f'}/f'_0$. This gives us $\varepsilon_0$ which is used to calculate $v_0$.

Defining

$$\mathcal{U}(\Delta; a, b) = \mathcal{V}(\Delta; a, b) \cup \mathcal{V}'(\Delta; a, b)$$

which concludes the proof.

As we said in the introduction section [3–Theorem 3] excludes a small set of values of $a$.

## 3.2 Predicting the Quadratic Generator with Known Multiplier and Unknown Shift

In this subsection we consider the problem of breaking the quadratic generator given $a$ and approximations to three consecutive values. We prove the following result:

**Theorem 2.** *Let $p$ be a prime number and let $\Delta$ be an integer such that $p > \Delta \ge 1$. For any $a \in \mathbb{F}_p^*$ and $c \in \mathbb{F}_p$, there exists a set $\mathcal{U}(\Delta; a, c) \subseteq \mathbb{F}_p$ of cardinality $\#\mathcal{U}(\Delta; a, c) = O(\Delta^5)$ with the following property: there exists an algorithm which, when given $a$ and $\Delta$-approximations $w_i$, $i = 0, 1, 2$ to three consecutive values $v_0, v_1, v_2$ produced by the quadratic generator* (1)*, where $v_0 \notin \mathcal{U}(\Delta; a, c)$, recovers $v_0$ and $c$ in deterministic polynomial time.*

*Proof.* We can assume that $\Delta^5 < p$ and that $v_0 \in \mathbb{F}_p$ is chosen so as not to lie in a certain subset $\mathcal{U}(\Delta; a, c)$ of $\mathbb{F}_p^*$ of cardinality $O(\Delta^5)$. As its definition is fairly complicated we define it gradually. By hypothesis, we have:

$$v_i = w_i + \varepsilon_i, \ |\varepsilon_i| \le \Delta, \ i = 0, 1, 2\ , \quad \text{and} \quad av_i^2 + c \equiv v_{i+1} \bmod p, \ i = 0, 1.$$

So, we obtain the following equation that involves the known parameters $a$, $w_i$ and with the desired information $\varepsilon_i$:

$$(aw_0^2 - w_1 - aw_1^2 + w_2) + 2aw_0 \underbrace{\varepsilon_0} - (1 + 2aw_1) \underbrace{\varepsilon_1} + \underbrace{\varepsilon_2} + a \underbrace{(\varepsilon_0^2 - \varepsilon_1^2)} \equiv 0 \bmod p \tag{11}$$

Then, the vector $(1, \varepsilon_0, \varepsilon_1, \varepsilon_2, \varepsilon_0^2 - \varepsilon_1^2)$ satisfies the congruence (11) with known coefficients. In order to handle a vector with norm-balanced components, we write:

$$(aw_0^2 - w_1 - aw_1^2 + w_2)\underbrace{\Delta^2} + 2aw_0\Delta\underbrace{\Delta\varepsilon_0} - (1 + 2aw_1)\Delta\underbrace{\Delta\varepsilon_1} + \\ + \Delta\underbrace{\Delta\varepsilon_2} + a\Delta^2\underbrace{(\varepsilon_0^2 - \varepsilon_1^2)} \equiv 0 \bmod p.$$

So, the $\boldsymbol{e} := (\Delta^2, \Delta\varepsilon_0, \Delta\varepsilon_1, \Delta\varepsilon_2, \varepsilon_0^2 - \varepsilon_1^2)$ lies in the lattice $\mathcal{L}$ consisting of $(x_0, x_1, x_2, x_3, x_4) \in \mathbb{Z}^5$ verifying:

$$\mathcal{L}: \begin{cases} (aw_0^2 - w_1 - aw_1^2 + w_2)x_0 + 2aw_0\Delta x_1 - (1 + 2aw_1)\Delta x_2 + \\ \qquad + \Delta x_3 + a\Delta^2 x_4 \equiv 0 \bmod p, \\ x_0 \equiv 0 \bmod \Delta^2, \\ x_1, x_2, x_3 \equiv 0 \bmod \Delta. \end{cases}$$

Also, we have $\|\boldsymbol{e}\| < 3\Delta^2$. We can compute on polynomial time a shortest vector $\boldsymbol{f}$ in the lattice $\mathcal{L}$:

$$\boldsymbol{f} =: (\Delta^2 f_0, \Delta f_1, \Delta f_2, \Delta f_3, f_4), \quad \|\boldsymbol{f}\| < 3\Delta^2, \\ |f_0| < 3, \ |f_1|, |f_2|, |f_3| < 3\Delta, \ |f_4| < 3\Delta^2. \tag{12}$$

We hope that $\boldsymbol{e}$ and $\boldsymbol{f}$ are always parallel, that this

$$\boldsymbol{d} := f_0\boldsymbol{e} - \boldsymbol{f} = (0, \Delta d_1, \Delta d_2, \Delta d_3, d_4) \in \mathcal{L}$$

is the null vector. If not, we claim that $d_1 = d_2 = 0$ unless $v_0$ belongs to the set $\mathcal{V}(\Delta; a, c)$ which we define below. Substituting in (11) we derive

$$2aw_0 d_1 - (1 + 2aw_1)d_2 + d_3 + ad_4 \equiv 0 \bmod p,$$

and using the bounds in (12)

$$|d_1|, |d_2|, |d_3| < 6\Delta, |d_4| < 9\Delta^2 \tag{13}$$

Now, plugin $w_i = v_i - \varepsilon_i, i = 0, 1$ in the above congruence, we get

$$2a(v_0 - \varepsilon_0)d_1 - (1 + 2a(v_1 - \varepsilon_1))d_2 + d_3 + ad_4 \equiv 0 \bmod p.$$

Substituting $v_1 \equiv av_0^2 + c \bmod p$ in the above congruence we obtain:

$$2ad_1v_0 - 2ad_1\varepsilon_0 - d_2 - 2ad_2(av_0^2 + c) + 2ad_2\varepsilon_1 + d_3 + ad_4 \equiv 0 \bmod p.$$

Then, $P(v_0) \equiv 0 \bmod p$ with

$$P(T) = -2a^2d_2T^2 + 2ad_1T - 2ad_1\varepsilon_0 - d_2 - 2ad_2c + 2ad_2\varepsilon_1 + d_3 + ad_4.$$

Let's define the first piece of the exceptional set $\mathcal{V}(\Delta; a, c)$ as the set of elements $v_0 \in \mathbb{F}_p$ such that there exist integers $d_1, d_2, d_3, d_4, \varepsilon_0, \varepsilon_1$ satifying:

$$d_1d_2 \not\equiv 0 \bmod p \quad \text{and} \quad P(v_0) \equiv 0 \bmod p.$$

The bounds in (13) imply that the number of elements in $\mathcal{V}(\Delta; a, c)$ is $O(\Delta^5)$, because in the equation:

$$-2a^2d_2v_0^2 + 2ad_1v_0 + 2ad_2c + d_2 - d_3 \equiv a(2d_2\varepsilon_1 - 2d_1\varepsilon_0 + d_4) \bmod p,$$

there may exist less than $O(\Delta^2)$ possibilities for the right term. On the other hand, in the left term, there may appear $O(\Delta^3)$ different (always nonconstant) polynomials.

Whenever $v_0 \notin \mathcal{V}(\Delta; a, c)$, it must be $d_1 = d_2 = 0$, because of the bounds for these integers, see again (13).

Once under this assumption, we look at the first coefficient of the vector $\boldsymbol{f}$. If $f_0 \neq 0$, we can easily recover:

$$\varepsilon_0 = f_1(f_0)^{-1},\ \varepsilon_1 = f_2(f_0)^{-1},\ \text{(as identities in } \mathbb{Z}).$$

We concentrate the study when $f_0 = 0$. Then, $\boldsymbol{d} = (0, 0, \Delta d_2, \Delta d_3, d_4)$ with

$$d_3 + ad_4 = f_3 + af_4 \equiv 0 \bmod p$$

It is easy to see that $f_4 \not\equiv 0 \bmod p$, otherwise $\boldsymbol{f}$ is the null vector. We compute integers $r, s$, with $\gcd(r, s) = 1$, and $|r| < 3\Delta$, $|s| < 3\Delta$, such that:

$$a \equiv rs^{-1} \bmod p$$

By Lemma 1, the possibilities for these integers do not vary as we consider different aproximations, but remain fixed for the parameters $a, p, \Delta$. Now, we change equation (11):

$$(rw_0^2 - sw_1 - rw_1^2 + sw_2) + 2rw_0\varepsilon_0 - (s + 2rw_1)\varepsilon_1 + s\varepsilon_2 + r(\varepsilon_0^2 - \varepsilon_1^2) \equiv 0 \bmod p.$$

$$(rw_0^2 - sw_1 - rw_1^2 + sw_2) + 2w_0r\underbrace{\varepsilon_0} - 2w_1r\underbrace{\varepsilon_1} + \underbrace{s(\varepsilon_2 - \varepsilon_1) + r(\varepsilon_0^2 - \varepsilon_1^2)} \equiv 0 \bmod p. \tag{14}$$

Finally,

$$(rw_0^2 - sw_1 - rw_1^2 + sw_2)\underbrace{\Delta^3} + 2w_0r\Delta\underbrace{\Delta^2\varepsilon_0} - 2w_1r\Delta\underbrace{\Delta^2\varepsilon_1} + \\ + \Delta^3\underbrace{s(\varepsilon_2 - \varepsilon_1) + r(\varepsilon_0^2 - \varepsilon_1^2)} \equiv 0 \bmod p.$$

So, the vector $\boldsymbol{e}' := (\Delta^3, \Delta^2\varepsilon_0, \Delta^2\varepsilon_1, s(\varepsilon_2 - \varepsilon_1) + r(\varepsilon_0^2 - \varepsilon_1^2))$ lies in the lattice:

$$\mathcal{L}' : \begin{cases} (rw_0^2 - sw_1 - rw_1^2 + sw_2)x_0 + 2rw_0\Delta x_1 - 2rw_1\Delta x_2 + \Delta^3 x_3 \equiv 0 \bmod p \\ x_0 \equiv 0 \bmod \Delta^2 \\ x_1, x_2 \equiv 0 \bmod \Delta^2 \end{cases}$$

Again, we now show that all short vectors in $\mathcal{L}'$ are parallel to $\boldsymbol{e}'$ unless $v_0$ belongs to the set $\mathcal{V}'(\Delta; a, b)$ which we define below.

Assume, for a contradiction, that there is another vector. We compute on polynomial time a vector $\boldsymbol{f}'$ with minimum norm in $\mathcal{L}'$.

$$\boldsymbol{f}' = (\Delta^3 f_0', \Delta^2 f_1', \Delta^2 f_2', f_3'),\ \|\boldsymbol{f}'\| < 13\Delta^3$$

$$|f_0'| < 13,\ |f_1'|, |f_2'| < 13\Delta,\ |f_3'| < 13\Delta^3 \tag{15}$$

The vector $\boldsymbol{d}' := f_0'\boldsymbol{e}' - \boldsymbol{f}' =: (0, \Delta^2 d_1', \Delta^2 d_2', d_3')$ is also in the lattice $\mathcal{L}'$. We can bound its coefficients by (15):

$$|d_1'|, |d_2'| < 26\Delta,\ |d_3'| < 169\Delta^3. \tag{16}$$

Now, by (14), we find that

$$2w_0 r d_1' - 2w_1 r d_2' + d_3' \equiv 0 \bmod p \tag{17}$$

Substituting $w_0 = v_i - \varepsilon_i$ in the congruence (17) we derive

$$2(v_0 - \varepsilon_0) r d_1' - 2(a v_0^2 + c - \varepsilon_1) r d_2' + d_3' \equiv 0 \bmod p. \tag{18}$$

Then, $P'(v_0) \equiv 0 \bmod p$ with

$$P'(T) := -2ard_2'T^2 + 2rd_1'T - 2rd_1'\varepsilon_0 - 2rd_2'c + 2rd_2'\varepsilon_1 + d_3'.$$

We define $\mathcal{V}'(\Delta; a, c)$ the set of elements $v_0 \in \mathbb{F}_p$ such that there exist integers $d_1', d_2', d_3', \varepsilon_0, \varepsilon_1, r, s$ with the appropriate bounds verifying:

$$d_1' d_2' \not\equiv 0 \bmod p \quad \text{and} \quad P'(v_0) \equiv 0 \bmod p.$$

By the bounds in the integers $d_1', d_2', d_3', \varepsilon_0, \varepsilon_1, r, s$, we have that $\#\mathcal{V}'(\Delta; a, c) = O(\Delta^5)$, because in the equation:

$$-2ard_2'v_0^2 + 2rd_1'v_0 + 2rd_2'c \equiv 2rd_1'\varepsilon_0 - 2rd_2'\varepsilon_1 - d_3' \bmod p,$$

there are $O(\Delta^3)$ options for the right side, and $O(\Delta^2)$ different (and nonconstant) polynomials in $v_0$ for the left one.

Now, if $v_0 \notin \mathcal{V}'(\Delta; a, c)$, it must be $d_1' \equiv d_2' \equiv 0 \bmod p$, then using bounds (16) we obtain $d_1' = d_2' = d_3' = 0$. This implies that vectors $\boldsymbol{e}'$ and $\boldsymbol{f}'$ are parallel which it is a contradiction. So, $\boldsymbol{e}'$ and $\boldsymbol{f}'$ are parallel vectors.

Once again, if we have $f_0' \not\equiv 0 \bmod p$, we recover easily the approximation errors and the orginal values. Now, if $f_0' \equiv 0 \Rightarrow f_0' = 0$, we would have $\boldsymbol{f}' = (0, 0, 0, 0')$ which it is a contradiction.

We just must define $\mathcal{U}(\Delta; a, c) := \mathcal{V}(\Delta; a, c) \cup \mathcal{V}'(\Delta; a, c)$ to comple the proof.

## 4 Remarks and Open Questions

Obviously our Theorem 1 is nontrivial only for $\Delta = O(p^{1/4})$ and Theorem 2 only for $\Delta = O(p^{1/5})$. Thus increasing the size of the admissible values of $\Delta$ (even at the cost of considering more consecutive approximations) is interesting.

One can presumably obtain a very similar result in the dual case, where $c$ is given but the multiplier $a$ is unknown.

As we have mentioned several other results about predictability of nonlinear generators have recently been obtained in [3]. However, they are somewhat weaker than the present result because each of them excludes a certain small exceptional set of pairs of parameters $(a, c)$. In particular the Theorem 5 of [3] when both multiplier and shift are secret. We believe that the approach of this work may help to eliminate this drawback. Certainly this question deserves further study.

We do not know how to predict the quadratic (and other generators considered in [3]) in the case when the modulus $p$ is secret as well. We remark that in the case of the linear congruential generator a heuristic approach to this problem has been proposed in [6]. However it is not clear how to extend this (even just heuristically) to the case of nonlinear generators.

**Acknowledgment.** This work is partially supported by Spanish Ministry of Science grant MTM2004-07086.

## References

1. M. Ajtai, R. Kumar and D. Sivakumar, 'A sieve algorithm for the shortest lattice vector problem', *Proc. 33rd ACM Symp. on Theory of Comput. (STOC 2001)*, Association for Computing Machinery, 2001, 601–610.
2. S. R. Blackburn, D. Gomez-Perez, J. Gutierrez and I. E. Shparlinski, 'Predicting the inversive generator', *Proc. 9th IMA Intern. Conf on Cryptography and Coding*, Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, **2898** (2003), 264–275.
3. S. R. Blackburn, D. Gomez-Perez, J. Gutierrez and I. E. Shparlinski, 'Predicting nonlinear pseudorandom number generators', *Math. Computation*, **74** (2005), 1471-1494.
4. E. F. Brickell and A. M. Odlyzko, 'Cryptanalysis: A survey of recent results', *Contemp. Cryptology*, IEEE Press, NY, 1992, 501–540.
5. M. Grötschel, L. Lovász and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer-Verlag, Berlin, 1993.
6. A. Joux and J. Stern, 'Lattice reduction: A toolbox for the cryptanalyst', *J. Cryptology*, **11** (1998), 161–185.
7. R. Kannan, 'Algorithmic geometry of numbers', *Annual Review of Comp. Sci.*, **2** (1987), 231–267.
8. R. Kannan, 'Minkowski's convex body theorem and integer programming', *Math. Oper. Res.*, **12** (1987), 415–440.
9. J. C. Lagarias, 'Pseudorandom number generators in cryptography and number theory', *Proc. Symp. in Appl. Math.*, Amer. Math. Soc., Providence, RI, **42** (1990), 115–143.
10. A. K. Lenstra, H. W. Lenstra and L. Lovász, 'Factoring polynomials with rational coefficients', *Mathematische Annalen*, **261** (1982), 515–534.
11. D. Micciancio and S. Goldwasser, *Complexity of lattice problems*, Kluwer Acad. Publ., 2002.
12. P. Q. Nguyen and J. Stern, 'Lattice reduction in cryptology: An update', in: W. Bosma (Ed), *Proc. ANTS-IV, Lect. Notes in Comp. Sci. Vol. 1838*, Springer-Verlag, Berlin, 2000, 85–112.

13. P. Q. Nguyen and J. Stern, 'The two faces of lattices in cryptology', in: J.H. Silverman (Ed), *Cryptography and Lattices Lect. Notes in Comp. Sci. Vol. 2146* , Springer-Verlag, Berlin, 2001, 146–180.
14. H. Niederreiter, 'New developments in uniform pseudorandom number and vector generation', in: H. Niederreiter and P.J. Shiue (Eds), *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Lect. Notes in Statistics Vol. 106*, Springer-Verlag, Berlin, 1995, 87–120.
15. H. Niederreiter, 'Design and analysis of nonlinear pseudorandom number generators', in G.I. Schueller and P. D. Spanos (Eds) *Monte Carlo Simulation*, A.A. Balkema Publishers, Rotterdam, 2001, 3–9.
16. H. Niederreiter and I. E. Shparlinski, 'Recent advances in the theory of nonlinear pseudorandom number generators', in: K.-T. Fang, F.J. Hickernell and H. Niederreiter (Eds), *Proc. Conf. on Monte Carlo and Quasi-Monte Carlo Methods, 2000*, Springer-Verlag, Berlin, 2002, 86–102.
17. H. Niederreiter and I. E. Shparlinski, 'Dynamical systems generated by rational functions', in: Marc Fossorier, Tom Høholdt and Alain Poli (Eds), *Applied Algebra, Algebraic Algorithms and Error Correcting Codes – AAECC-15, Lect. Notes in Comp. Sci. Vol. 2643*, Springer-Verlag, Berlin, 2003, 6–17.

# Attack the Dragon[⋆]

Håkan Englund and Alexander Maximov

Dept. of Information Technology, Lund University,
P.O. Box 118, 221 00 Lund, Sweden

**Abstract.** Dragon is a word oriented stream cipher submitted to the ECRYPT project, it operates on key sizes of 128 and 256 bits. The original idea of the design is to use a nonlinear feedback shift register (NLFSR) and a linear part (counter), combined by a filter function to generate a new state of the NLFSR and produce the keystream. The internal state of the cipher is 1088 bits, i.e., any kinds of TMD attacks are not applicable. In this paper we present two statistical distinguishers that distinguish Dragon from a random source both requiring around $O(2^{155})$ words of the keystream. In the first scenario the time complexity is around $O(2^{155+32})$ with the memory complexity $O(2^{32})$, whereas the second scenario needs only $O(2^{155})$ of time, but $O(2^{96})$ of memory. The attack is based on a statistical weakness introduced into the keystream by the filter function $F$. This is the first paper presenting an attack on Dragon, and it shows that the cipher does not provide full security when the key of size 256 bits is used.

## 1 Introduction

A stream cipher is a cryptographic primitive used to ensure privacy over a communication channel. A common way to build a stream cipher is to use a keystream generator (KSG) and add the plaintext and the output from the keystream generator, resembling a one-time pad. A block cipher is another cryptographic primitive, which could be considered as a one-to-one function, mapping a block of the plaintext to a block of the ciphertext. Although block ciphers are well studied, stream ciphers can offer certain advantages compared to block ciphers. Stream ciphers can offer much higher speed, and can be constructed to be much smaller in hardware, and thus they are of great interest to the industry. To mention a few of the most recent proposals of such word-oriented KSGs are, e.g., VMPC [1], RC4A [2], RC4 [3], SEAL [4], SOBER [5], SNOW [6, 7], PANAMA [8], Scream [9], MUGI [10], Helix [11], Rabbit [12], Turing [13], etc.

The NESSIE project [14] was funded by the European Unions Fifth Framework Program and was launched in 2000. The main objective was to collect a

[⋆] The work described in this paper has been supported in part by Grant VR 621-2001-2149, and in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

© Springer-Verlag Berlin Heidelberg 2005

portfolio of strong cryptographic primitives from different fields of cryptography, one of those fields was stream ciphers. During those three years of NESSIE new techniques for cryptanalysis on stream ciphers were found, and many new proposals were broken. After a few rounds of the project evaluation, all of the stream cipher proposals were found to contain some weaknesses. At the end, no stream cipher was included in the final portfolio.

The situation clearly requires the cryptographic community devote greater attention to design and analysis of stream ciphers. Due to this reason, the European project ECRYPT announced a call for stream cipher primitives. 35 proposals were submitted to the project by April 2005, and most of them were presented at the workshop SKEW 2005 [15] in May.

Cryptanalysis techniques discovered during the NESSIE project have allowed to strengthen new designs greatly, and to break new algorithms has become more difficult. However, there are many good submissions to ECRYPT, and the stream cipher Dragon [16] is one of them.

Dragon, designed by a group of researches, Ed Dawson et. al., is a word oriented stream cipher based on a linear block (counter) and a *nonlinear feedback shift register* (NLFSR) with a very large internal state of 1088 bits, which is updated by a nonlinear function denoted by $F$. This function is also used as a filter function producing the keystream. The idea to use a NLFSR is quite modern, and there are not many cryptanalysis techniques on NLFSRs yet found.

*This is the first work which propose an attack on Dragon.* In a distinguishing attack one has to decide whether a given sequence (keystream) is the product of a cipher, or a truly random generator. In this paper we show how statistical weaknesses in the $F$ function can be used to create a distinguisher for Dragon. Our distinguishing attack requires around $O(2^{155})$ words of keystream from Dragon, it has time complexity $O(2^{155+32})$ and needs $O(2^{32})$ of memory, an alternative method is also presented that only requires time complexity $O(2^{155})$ but needs $O(2^{96})$ of memory. This is an academic attack which shows that Dragon does not provide full security when a key of size 256 bits is used, i.e., it can be broken faster than exhaustive search. This kind of analysis is, perhaps, the most powerful attack on stream ciphers, and, in some cases, it can be turned into a key recovery attack.

The outline of the paper is the following. In Section 2 a short description of the stream cipher Dragon is given. Afterward, in Section 3, we derive linear relations and build our distinguisher. In Section 4 we summarize different attack scenarios on Dragon, and finally, in Section 5 we present our results, make conclusions and discuss possible ways to overcome the attack.

### 1.1 Notations and Assumptions

For notation purposes we use $\oplus$ and $\boxplus$ to denote 32 bit parallel XOR and arithmetical addition modulo $2^{32}$, respectively. By $x \gg n$ we denote a binary shift of $x$ by $n$ bits to the right. We write $x^{(t)}$ to refer the value of a variable $x$ at the time instance $t$. By $P_{\texttt{Expr}}$ we denote a distribution of a random variable or an expression "`Expr`".

To build the distinguisher, we first make two reasonable assumptions common for linear cryptanalysis:

(a) Assume that at any time $t$ the internal state of NLFSR is from the uniform distribution, i.e., the words $B_i$ are considered independent and uniformly distributed;
(b) Assume that the keystream words are independent.

## 2 A Short Description of Dragon

Dragon is a stream cipher constructed using a large nonlinear feedback shift register, an update function denoted by $F$, and a memory denoted by $M$[1] . It is a word oriented cipher operating on 32 bit words, the NLFSR is 1024 bits long, i.e., 32 words long. The words in the internal state are denoted by $B_i,\ 0 \leq i \leq 31$ . The memory $M$ (counter) contains 64 bits, which is used as a linear part with the period of $2^{64}$. The cipher handles two key sizes, namely 128 bits keys and 256 bit keys, in our attack we disregard the initialization procedure and just assume that the initial state of the NLFSR is truly random.

Each round the $F$ function takes six words as input and produces six words of output, as shown in Figure 1. These six words, denoted by $a, b, c, d, e, f$, are formed from words of the NLFSR and the memory register $M$, as explained in (1), where $M = (M_L || M_R)$.

$$\begin{array}{lll} a = B_0 & b = B_9 & c = B_{16} \\ d = B_{19} & e = B_{30} \oplus M_L & f = B_{31} \oplus M_R \end{array} \tag{1}$$

The $F$ function uses six $\mathbb{Z}_{2^{32}} \to \mathbb{Z}_{2^{32}}$ $S$-boxes $G_1,\ G_2,\ G_3,\ H_1,\ H_2$ and $H_3$, the purpose of which is to provide high algebraic immunity and non-linearity. These $S$-boxes are constructed from two other fixed $\mathbb{Z}_{2^8} \to \mathbb{Z}_{2^{32}}$ $S$-boxes, $S_1$ and $S_2$, as shown below.

$$\begin{aligned} G_1(x) &= S_1(x_0) \oplus S_1(x_1) \oplus S_1(x_2) \oplus S_2(x_3), \\ G_2(x) &= S_1(x_0) \oplus S_1(x_1) \oplus S_2(x_2) \oplus S_1(x_3), \\ G_3(x) &= S_1(x_0) \oplus S_2(x_1) \oplus S_1(x_2) \oplus S_1(x_3), \\ H_1(x) &= S_2(x_0) \oplus S_2(x_1) \oplus S_2(x_2) \oplus S_1(x_3), \\ H_2(x) &= S_2(x_0) \oplus S_2(x_1) \oplus S_1(x_2) \oplus S_2(x_3), \\ H_3(x) &= S_2(x_0) \oplus S_1(x_1) \oplus S_2(x_2) \oplus S_2(x_3), \end{aligned}$$

where 32 bits of input, $x$, is represented by its four bytes as $x = x_0||x_1||x_2||x_3$.

The exact specification of the $S$-boxes can be found in [16]. The output of the function $F$ is denoted as $(a', b', c', d', e', f')$, from which the two words $a'$ and

[1] This is rather a new way to design stream ciphers, when two independent linear and nonlinear parts are combined by a filter function. A similar idea is used in other proposals to ECRYPT, e.g., stream cipher Grain and others.

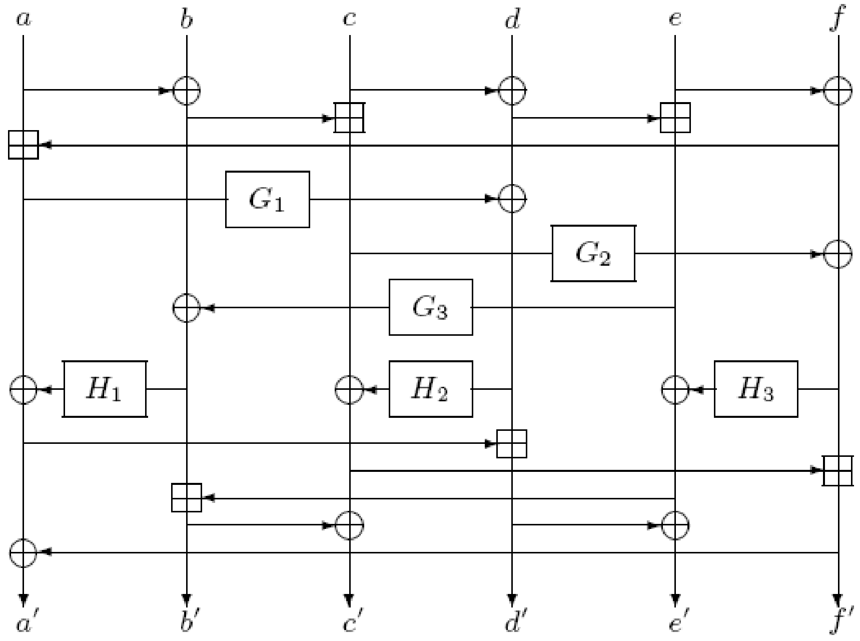


**Fig. 1.** $F$-function

$e'$ forms 64 bits of keystream as $k = a'||e'$. Two other output words from the filter function are used to update the NLFSR as follows $B_0 = b'$, $B_1 = c'$, the rest of the state is updated as $B_i = B_{i-2},\ 2 \leq i \leq 31$. A short description of the keystream generation function is summarized in Figure 2.

```
   Input = ${B_0||\ldots||B_{31}, M}$
1. $(M_L||M_R) = M$.
2. $a = B_0,\ b = B_9,\ c = B_{16},\ d = B_{19},\ e = B_{30} \oplus M_L,\ f = B_{31} \oplus M_R$.
3. $(a', b', c', d', e', f') = F(a, b, c, d, e, f)$.
4. $B_0 = b',\ B_1 = c'$
5. $B_i = B_{i-2},\ 2 \leq i \leq 31$.
6. $M{+}{+}$
7. $k = a'||e'$
   Output = ${k, B_0, \ldots, B_{31}, M}$
```

**Fig. 2.** Dragons's Keystream Generation Function

## 3 A Linear Distinguishing Attack on Dragon

### 3.1 Linear Approximation of the Function *F*

Recall, at time $t$ the input to the function $F$ is a vector of six words $(a, b, c, d, e, f) = (B_0, B_9,\ B_{16}, B_{19}, B_{30} \oplus M_L, B_{31} \oplus M_R)$. The output from the function is $(a', b', c', d', e', f')$. To simplify further expressions let us introduce new variables.

$$\begin{cases} b'' &= b \oplus a = B_9 \oplus B_0 \\ c'' &= c \boxplus (a \oplus b) = B_{16} \boxplus (B_9 \oplus B_0) \\ d'' &= d \oplus c = B_{19} \oplus B_{16} \\ f'' &= f \oplus e = B_{30} \oplus B_{31} \oplus M_L \oplus M_R \end{cases} \tag{2}$$

If the words denoted by $B$s are independent, then these new variables will also be independent (since $B_{19}$ is independent of $B_{16}$ and random, then $d''$ is independent and random as well; similarly, independence of $B_{16}$ lead to the independence of $c''$, etc.).

The output from $F$ can be expressed via $(a, b'', c'', d'', e, f'')$ as follows.

$$\begin{cases} a' &= (a \boxplus f'') \oplus H_1(b'' \oplus G_3(e \boxplus d''))\oplus \\ & \qquad \Big((f'' \oplus G_2(c'')) \boxplus \Big(c'' \oplus H_2(d'' \oplus G_1(a \boxplus f''))\Big)\Big) \\ e' &= (e \boxplus d'') \oplus H_3(f'' \oplus G_2(c''))\oplus \\ & \qquad \Big((d'' \oplus G_1(a \boxplus f'')) \boxplus \Big((a \boxplus f'') \oplus H_1(b'' \oplus G_3(e \boxplus d''))\Big)\Big) \end{cases} \tag{3}$$

Let us now analyze the expression for $a'$. The variable $b''$ appears only once (in the input of $H_1$), which means that this input is independent from other terms of the expression, i.e., the term $H_1(\ldots)$ can be substituted by $H_1(r_1)$, where $r_1$ is some independent and uniformly distributed random variable. Then, the same will happen with the input for $H_2$.

We would like to approximate the expression for $a'$ as

$$a' = a \oplus N_a, \tag{4}$$

where $N_a$ is some non uniformly distributed noise variable. If we XOR both sides with $a$ and then substitute $a'$ with the expression from (3), we derive

$$N_a = a \oplus (a \boxplus f'') \oplus H_1(r_1) \oplus \Big((f'' \oplus G_2(c'')) \boxplus (c'' \oplus H_2(r_2))\Big). \tag{5}$$

Despite the fact that $G$ and $H$ are $\mathbb{Z}_{2^{32}} \to \mathbb{Z}_{2^{32}}$ functions, they are not likely to be one-to-one mappings, consider the way the $S$-boxes are used as $\mathbb{Z}_{2^8} \to \mathbb{Z}_{2^{32}}$ functions [2] . It means that even if the input to a $G$ or a $H$ function is completely random, then the output will still be biased. Moreover, the output from the expressions ($x \oplus G_i(x)$ and similarly $x \oplus H_i(x)$) is also biased, since $x$ in these expressions plays a role of an approximation of the $G_i$ and the $H_i$ functions. These observations mean that the noise variable $N_a$, is also biased if the input variables are independent and uniformly distributed.

[2] The cipher Turing uses similar $\mathbb{Z}_{2^{32}} \to \mathbb{Z}_{2^{32}}$ functions based on $\mathbb{Z}_{2^8} \to \mathbb{Z}_{2^{32}}$ $S$-boxes, which can be regarded as a source of weakness. However, no attack was found on Turing so far.

By a similar observation, the expression for $e'$ can also be approximated as follows.

$$e' = e \oplus N_e, \tag{6}$$

where $N_e$ is the noise variable. The expression for $N_e$ can similarly be derived as

$$N_e = e \oplus (e \boxplus d'') \oplus H_3(r_3) \oplus \Big((d'' \oplus G_1(a'')) \boxplus (a'' \oplus H_1(r_4)\Big), \tag{7}$$

where $a'' = a \boxplus f''$ is a new random variable, which is also independent since it has $f''$ as its component, and $f''$ does not appear anywhere else in the expression (7). The two new variables $r_3$ and $r_4$ are also independent and uniformly distributed random variables by a similar reason.

## 3.2 Building the Distinguisher

The key observation for our distinguisher, is that one of the input words to the filter function $F$, at time $t$ is partially repeated as input to $F$ at time $t+15$, i.e.,

$$e^{(t+15)} = a^{(t)} \oplus M_L^{(t+15)}. \tag{8}$$

Let us consider the following sum of two words from the keystream.

$$\begin{aligned} s^{(t)} &= a'^{(t)} \oplus e'^{(t+15)} = (a^{(t)} \oplus N_a^{(t)}) \oplus (a^{(t)} \oplus M_L^{(t+15)} \oplus N_e^{(t+15)}) \\ &= \underbrace{N_a^{(t)} \oplus N_e^{(t+15)}}_{N_{tot}^{(t)}} \oplus M_L^{(t+15)} \end{aligned} \tag{9}$$

By this formula we show how to sample from a given keystream, so that the samples $s^{(t)}$ are from some nonuniform distribution $P_{\texttt{Dragon}}$ of the noise variable $N_{tot}^{(t)}$ (later also referred as $P_{N_{tot}^{(t)}}$). Collected samples $s^{(t)}$ form a so-called *type* $P_{\texttt{Type}}$, or an *empirical distribution*. Then, we have two hypothesis:

$$\begin{cases} H_1 : & P_{\texttt{Type}} \text{ is drawn according to } P_{\texttt{Dragon}} \\ H_2 : & P_{\texttt{Type}} \text{ is drawn according to } P_{\texttt{Random}} \end{cases}. \tag{10}$$

To distinguish between them with negligible probability of error (whether the samples are drawn from the noise distribution $P_{\texttt{Dragon}}$ or from the uniform distribution $P_{\texttt{Random}}$), the type should be constructed from the following number of samples

$$N \approx 1/\epsilon^2, \tag{11}$$

where $\epsilon$ is the bias, calculated as

$$\epsilon = |P_{\texttt{Dragon}} - P_{\texttt{Random}}| = \sum_{x=0}^{2^{32}-1} |P_{\texttt{Dragon}}(x) - P_{\texttt{Random}}(x)|. \tag{12}$$

When the type $P_{\mathtt{Type}}$ is constructed, a common tool in statistical analysis is the log-likelihood test. The ratio $I$ is calculated as

$$\begin{aligned} I &= D(P_{\mathtt{Type}}||P_{\mathtt{Random}}) - D(P_{\mathtt{Type}}||P_{\mathtt{Dragon}}) \\ &= \sum_{x=0}^{2^{32}-1} P_{\mathtt{Type}}(x) \log_2 \frac{P_{\mathtt{Dragon}}(x)}{P_{\mathtt{Random}}(x)} \qquad , \end{aligned} \tag{13}$$

where $D(\cdot)$ is the *relative entropy* defined for any two distributions $P_1$ and $P_2$ as

$$D(P_1||P_2) = \sum_{x \in \Omega} P_1(x) \log_2 \frac{P_1(x)}{P_2(x)}, \tag{14}$$

where $\Omega$ is the probability space.

Finally, the decision rule $\delta(P_{\mathtt{Type}})$ is the following

$$\delta(P_{\mathtt{Type}}) = \begin{cases} H_1, & \text{if } I \geq 0 \\ H_2, & \text{if } I < 0 \end{cases}. \tag{15}$$

For more on statistical analysis and hypothesis testing we refer to, e.g., [17, 18].

The remaining question is how to deal with the counter value $M_L$. Below we present a set of possible solutions that one could consider.

(1) One possible solution would be to guess the initial state of the counter $M^{(0)}$ (in total $2^{64}$ combinations), and then construct $2^{64}$ types from the given keystream, assuming the value $M_L^{(t)}$ in correspondence to the guessed initial value of $M^{(0)}$. However, it will increase the time complexity of the distinguisher by $2^{64}$ times;

(2) One more possibility is to guess the first 32 bits $M_R^{(0)}$ of the initial value of the counter $M^{(0)}$, i.e., $2^{32}$ values. If we do so, then we always know when the upper 32 bits $M_L$ are increased, i.e., at any time $t$ we can express $M_L^{(t)}$ as follows.

$$M_L^{(t)} = M_L^{(0)} \boxplus \Delta^{(t)}, \tag{16}$$

where $\Delta^{(t)}$ is known at each time, since $M_R^{(t)}$ is known. Recall from (9), the noise variable $N_{tot}^{(t)}$ is expressed as $s^{(t)} \oplus M_L^{(t+15)}$. However, this expression can also be approximated as

$$s^{(t)} \oplus (M_L^{(0)} \boxplus \Delta^{(t+15)}) \to s^{(t)} \oplus (M_L^{(0)} \oplus \Delta^{(t+15)}) \oplus N_2, \tag{17}$$

where $N_2$ is a new noise variable due to the approximation of the kind "$\boxplus \Rightarrow \oplus$". Since $M_L^{(0)}$ can be regarded as a constant for every sample $s^{(t)}$, it only "shifts" the distribution, but will not change the bias. Consider that a shift of the uniform distribution is again the uniform distribution, so, the distance between the noise and the uniform distributions will remain the same. This solution requires $O(2^{32})$ guesses, and also introduce a new noise variable $N_2$;

(3) Another possible solution could be to consider the sum of two consecutive samples $s^{(t)} \oplus s^{(t+1)}$. Since $M_L$ changes slowly, then with probability $(1 - 2^{-32})$ we have $M_L^{(t)} = M_L^{(t+1)}$, and this term will be eliminated from the expression for that new sample. Unfortunately, this method will decrease the bias significantly, and then the number of required samples $N$ will be much larger than in the previous cases.

In our attack we tried different solutions, and based on simulations we decided to choose solution (2) for our attack, as it has the lowest attack complexity.

### 3.3 Calculation of the Noise Distribution

Consider the expression for the noise variable $s^{(t)} \oplus M_L^{(t+15)} = N_a^{(t)} \oplus N_e^{(t+15)}$. For simplicity in the formula, we omit time instances for variables.

$$\begin{aligned} N_{tot}^{(t)} = N_a^{(t)} \oplus N_e^{(t+15)} = (a \boxplus f'') \oplus (a \boxplus d'') \oplus H_1(r_1) \oplus H_3(r_3) \oplus \\ \oplus \Big( (f'' \oplus G_2(c'')) \boxplus (c'' \oplus H_2(r_2)) \Big) \oplus \Big( (d'' \oplus G_1(a'')) \boxplus (a'' \oplus H_1(r_4)) \Big) \end{aligned} \tag{18}$$

We propose two ways to calculate the distribution of the total noise random variable $N_{tot}^{(t)}$. Lets truncate the word size by $n$ bits (when we consider the expression modulo $2^n$), then in the first case the computational complexity is $O(2^{4n})$ . This complexity is too high and, therefore, requires the noise variable to be truncated by some number of bits $n \ll 32$, much less than 32 bits. The second solution has a better complexity $O(n2^n)$, but introduces two additional approximations into the expression, which makes the calculated bias smaller than the real value, i.e., by this solution we can verify the lower bound for the bias of the noise variable. Below we describe two methods and give our simulation results on the bias of the noise variable $N_{tot}^{(t)}$.

(I) Consider the expression (18) taken by modulo $2^n$, for some $n = 1 \ldots 32$. Then the distribution of the noise variable can be calculated by the following steps.
   a) Construct three distributions, two of them are conditioned

   $$P_{(G_2(c'') \mod 2^n | c'')}, \quad P_{(G_1(a'') \mod 2^n | a'')}, \quad P_{(H_1(x) \mod 2^n)}{}^3.$$

   The algorithm requires one loop for $c''$ ($a''$ and $x$) of size $2^{32}$. The time required is $O(3 \cdot 2^{32})$;
   b) Afterwards, construct two more conditioned distributions

   $$P_{(d'' \oplus G_1(a'')) \boxplus (a'' \oplus H_1(r_4)) \mod 2^n | d'')}$$

   and

   $$P_{(f'' \oplus G_1(c'')) \boxplus (c'' \oplus H_2(r_2)) \mod 2^n | f'')}.$$

[3] If the inputs to the $H_i$ functions is random, their distributions are the same, i.e., $P_{H_1} = P_{H_2} = P_{H_3}$.

This requires four loops for $d'', a'', x(= G_1(a'') \mod 2^n)$, and $y(= H_1(r_4) \mod 2^n)$, which takes time $O(2^{4n})$ (and similar for the second distribution);

c) Then, calculate another two conditioned distributions

$$P_{(\texttt{Expr}_1|a)} = P_{((a \boxplus f'') \oplus (f'' \oplus G_1(c'')) \boxplus (c'' \oplus H_2(r_2)) \mod 2^n|a)},$$

$$P_{(\texttt{Expr}_2|a)} = P_{((a \boxplus d'') \oplus (d'' \oplus G_1(a'')) \boxplus (a'' \oplus H_1(r_4)) \mod 2^n|a)}.$$

Each takes time $O(2^{3n})$;

d) Finally, combine the results, partially using FHT, and then calculate the bias of the noise:

$$P_{N_{tot}} = P_{(\texttt{Expr}_1|a)} \oplus P_{(\texttt{Expr}_2|a)} \oplus P_{H_1} \oplus P_{H_3}.$$

This will take time $O(2^{3n} + 3n \cdot 2^n)$.

This algorithm calculates the exact distribution of the noise variable taken modulo $2^n$, and has the complexity $O(2^{4n})$. Due to such a high computational complexity we could only manage to calculate the bias of the noise when $n = 8$ and $n = 10$:

$$\begin{aligned} \epsilon_I\big|_{n=8} &= 2^{-80.59} \\ \epsilon_I\big|_{n=10} &= 2^{-80.57}. \end{aligned} \tag{19}$$

(II) Consider two additional approximations of the second $\boxplus$ to $\oplus$ in (18). Then, the total noise can be expressed as

$$\begin{aligned} N_{tot}^{(t)} =& H_1(r_1) \oplus H_2(r_2) \oplus H_3(r_3) \oplus H_1(r_4) \oplus \big(G_2(c'') \oplus c''\big) \\ & \oplus \big(G_1(a'') \oplus a''\big) \oplus N_3 \oplus N_{2,a} \oplus N_{2,e}, \end{aligned} \tag{20}$$

where

$$N_3 = (a \boxplus f'') \oplus (a \boxplus d'') \oplus f'' \oplus d'',$$

and $N_{2,a}$ and $N_{2,e}$ are two new noise variables due to the approximation $\boxplus \Rightarrow \oplus$, i.e., $N_{2,a} = (x \boxplus y) \oplus (x \oplus y)$, for some random inputs $x$ and $y$, and similar for $N_{2,e}$. Introduction of two new noise variables will statistically make the bias of the total noise variable smaller, but it can give us a lower bound of the bias, and also allow us to operate with distributions of size $2^{32}$.

First, calculate the distributions $P_{(H_i)}$, $P_{(G_1(a'') \oplus a'')}$ and $P_{(G_1(c'') \oplus c'')}$, each taking time $O(2^{32})$. Afterward, note that the expressions for $N_{2,a}, N_{2,e}$ and $N_3$ belong to the class of *pseudo-linear functions modulo* $2^n$ (PLFM), which were introduced in [19]. In the same paper, algorithms for construction of their distributions were also provided, which take time around $O(\delta \cdot 2^n)$, for some small $\delta$. The last step is to perform the convolution of precomputed distribution tables via FHT in time $O(n2^n)$. Algorithms (PLFM distribution construction and computation of convolutions) and data structures for operating on large distributions are given in [19]. If we consider $n = 32$, then the total time complexity to calculate the distribution table for $N_{tot}$ will be around $O(2^{38})$ operations, which is feasible for a common PC. It took us a few days to accomplish such calculations on a usual PC with memory 2Gb and 2×200Gb of HDD, and the received bias of $N_{tot}$ was

$$\epsilon_{II}|_{n=32} = 2^{-74.515}. \tag{21}$$

If we also approximate $(M_L^{(0)} \boxplus \Delta^{(t)}) \to (M_L^{(0)} \oplus \Delta^{(t)}) \oplus N_2$, and add the noise $N_2$ to $N_{tot}$, we receive the bias

$$\epsilon_{II}^{\Delta}|_{n=32} = 2^{-77.5}, \tag{22}$$

which is the lower bound meaning that our distinguisher requires approximately $O(2^{155})$ words of the keystream, according to (12).

## 4 Attack Scenarios

In the previous section we have shown how to sample from the given keystream, where 32 bit samples are drawn from the noise distribution with the bias $\epsilon_{II}^{\Delta}|_{n=32} = 2^{-77.5}$. I.e., our distinguisher needs around $O(2^{155})$ words of the keystream to successfully distinguish the cipher from random. Unfortunately, to construct the type correctly we have to guess the initial value of the linear part of the cipher, the lower 32 bits $M_R^{(0)}$ of the counter $M$. This guess increases the time complexity of our attack to $O(2^{187})$, and requires memory $O(2^{32})$. The algorithm of our distinguisher for Dragon is given in Table 1.

**Table 1.** The distinguisher for Dragon (Scenario I)

```
**for** $0 \le M_R^{(0)} < 2^{32}$
    $P_{\texttt{Type}}(x) = 0, \quad \forall x \in \mathbb{Z}_{2^{32}}$
    $\Delta = 0$  (or $= -1$, if $M_R^{(0)} = 0$)
    **for** $t = 0, 1, \ldots, 2^{155}$
        **if** $(M_R^{(0)} \boxplus t) = 0$ **then** $\Delta = \Delta \boxplus 1$
        $s^{(t)} = a'^{(t)} \oplus e'^{(t+15)} \oplus \Delta$
        $P_{\texttt{Type}}(s^{(t)}) = P_{\texttt{Type}}(s^{(t)}) + 1$
    $I = \sum_{x \in \mathbb{Z}_{2^{32}}} P_{\texttt{Type}}(x) \cdot \log_2(P_{\texttt{Dragon}}(x)/2^{-32})$
    **If** $I \ge 0$ **break** and **output** : Dragon
**output** : Random source
```

**Table 2.** Distinguisher for Dragon with lower time complexity (Scenario II)

```
**for** $0 \le t < 2^{155}$
    $T[t \mod 2^{64}][a'^{(t)} \oplus e'^{(t+15)}] + +$
**for** $M_R^{(0)} = 0, \ldots, 2^{32} - 1$
    **for** $\Delta = 0, \ldots, 2^{64} - 1$
        **for** $x = 0, \ldots, 2^{32} - 1$
            $P_{\mathtt{Type}}\Big(x \oplus \big((\Delta \boxplus M_R^{(0)}) \gg 32\big)\Big) + = T[\Delta][x]$
    $I = \sum_{x \in \mathbb{Z}_{2^{32}}} P_{\mathtt{Type}}(x) \cdot \log_2(P_{\mathtt{Dragon}}(x)/2^{-32})$
    **If** $I \ge 0$ **break** and **output** : Dragon
**output** : Random source
```

We, however, can also show that time complexity can easily be reduced downto $O(2^{155})$, if memory of size $O(2^{96})$ is available. Assume we first construct a special table $T[\Delta][s] = \#\{t \equiv \Delta \mod 2^{64}, s^{(t)} = s\}$, where the samples are taken as $s^{(t)} = a'^{(t)} \oplus e'^{(t+15)}$. Afterwards, for each guess of $M_L^{(0)}$ the type $P_{\mathtt{Type}}(\cdot)$ is then constructed from the table $T$ in time $O(2^{96})$. Hence, the total time complexity will be $O(2^{155} + 2^{32} \cdot 2^{96}) \approx O(2^{155})$. This scenario is given in Table 2.

## 5 Results and Conclusions

Two versions of a distinguishing attack on Dragon were found. The first scenarios requires a computational complexity of $O(2^{187})$ and needs memory only $O(2^{32})$. However, the second scenario has a lower time complexity around $O(2^{155})$, but requires a larger amount of memory $O(2^{96})$. These attacks show that Dragon does not provide full security and can successfully be broken much faster than the exhaustive search, when a key of 256 bits is used.

From the specification of Dragon we also note that the amount of the keystream for an unique pair of the IV and the key is limited to $2^{64}$. However, our attack works when the same key and IV are used to produce $2^{155}$ words of the keystream. This is an academic attack which shows a statistical weakness of the keystream sequence, and reveals the leakage in the design.

Actually, our distinguisher consists of $2^{32}$ subdistinguishers. If one of them says “this is Dragon”, then it is taken as the result of the final distinguisher. If all subdistinguishers output “Random source”, then the overall result is “Random” as well [4].

[4] The idea to use many subdistinguishers was first proposed in the attack on Scream [20].

Below we give a few suggestions how to prevent Dragon from this kind of attack:

1) The linear part $M$ changes predictably, when the initial state is known. It might be more difficult to mount the attack if the update of $M$ would depend on some state of the NLFSR;
2) Another leakage is that two words $a'||e'$ are accessible to the attacker. If we would have an access only to $a'$, or, may be, some other combination of the output from $F$ (like, the output $a'||d'$, instead), then it might also protect the cipher from this attack. However, both these suggestions have weaknesses for different reasons;
3) One more weakness are poor $G_i$ and $H_i$ $S$-boxes. May be they can be constructed in a different way, closer to some one-to-one mapping function.

Several new stream cipher proposals are based on NLFSRs and this topic has been poorly investigated so far. We believe that it is important to study such primitives, since it could be an interesting replacement for widely used LFSR based stream ciphers.

## References

1. B. Zoltak. VMPC one-way function and stream cipher. In B. Roy and W. Meier, editors, *Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 210–225. Springer-Verlag, 2004.
2. S. Paul and B. Preneel. A new weakness in the rc4 keystream generator and an approach to improve the security of the cipher. In B. Roy and W. Meier, editors, *Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 245–259. Springer-Verlag, 2004.
3. P. Rogaway and D. Coppersmith. A software-optimized encryption algorithm. *Journal of Cryptology*, 11(4):273–287, 1998.
4. P. Rogaway and D. Coppersmith. A software-optimised encryption algorithm. In R.J. Anderson, editor, *Fast Software Encryption'93*, volume 809 of *Lecture Notes in Computer Science*, pages 56–63. Springer-Verlag, 1994.
5. P. Hawkes and G.G. Rose. Primitive specification and supporting documentation for SOBER-t16 submission to NESSIE. In *Proceedings of First Open NESSIE Workshop*, 2000. Available at *http://www.cryptonessie.org*, Accessed October 5, 2003.
6. P. Ekdahl and T. Johansson. SNOW - a new stream cipher. In *Proceedings of First Open NESSIE Workshop*, 2000.
7. P. Ekdahl and T. Johansson. A new version of the stream cipher SNOW. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography—SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 47–61. Springer-Verlag, 2002.
8. J. Daemen and C. Clapp. Fast hashing and stream encryption with PANAMA. In *Fast Software Encryption'98*, volume 1372 of *Lecture Notes in Computer Science*, pages 60–74. Springer-Verlag, 1998.
9. S. Halevi, D. Coppersmith, and C.S. Jutla. Scream: A software-efficient stream cipher. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 195–209. Springer-Verlag, 2002.

10. D. Watanabe, S. Furuya, H. Yoshida, K. Takaragi, and B. Preneel. A new keystream generator MUGI. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 179–194. Springer-Verlag, 2002.
11. N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, S. Lucks, and T. Kohno. Helix fast encryption and authentication in a single cryptographic primitive. In *Fast Software Encryption 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 330–346. Springer-Verlag, 2003.
12. M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansed, and O. Scavenius. Rabbit: A new high-performance stream cipher. In *Fast Software Encryption 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 307–329. Springer-Verlag, 2003.
13. G.G. Rose and P. Hawkes. Turing: A fast stream cipher. In T. Johansson, editor, *Fast Software Encryption 2003*, To be published in Lecture Notes in Computer Science. Springer-Verlag, 2003.
14. NESSIE. New European Schemes for Signatures, Integrity, and Encryption. Available at *http://www.cryptonessie.org*, Accessed August 18, 2003, 1999.
15. SKEW. Symmetric key encryption workshop. Available at *http://www2.mat.dtu.dk/people/Lars.R.Knudsen/stvl/*, Accessed August 6, 2005, 2005.
16. K. Chen, M. Henricksen, W. Millan, J. Fuller, L. Simpson, E. Dawson, H. Lee, and S. Moon. Dragon: A fast word based stream cipher. *ECRYPT Stream Cipher Project Report 2005/006*.
17. D. Coppersmith, S. Halevi, and C.S. Jutla. Cryptanalysis of stream ciphers with linear masking. In M. Yung, editor, *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, 2002.
18. J. Golić. Intrinsic statistical weakness of keystream generators. pages 91–103, 1994.
19. A. Maximov and T. Johansson. Fast computation of large distributions and its cryptographic applications. To appear at Asiacrypt 2005.
20. T. Johansson and A. Maximov. A Linear Distinguishing Attack on Scream. In *Information Symposium in Information Theory—ISIT 2003*, page 164. IEEE, 2003.

# Two Algebraic Attacks Against the F-FCSRs Using the IV Mode

Thierry P. Berger[1] and Marine Minier[2]

[1] LACO, Faculté des Sciences de Limoges,
23 avenue Albert Thomas, F-87060 Limoges Cedex, France
thierry.berger@unilim.fr
[2] INSA Lyon, CITI , 21 Avenue Jean Capelle,
F-69621 Villeurbanne, Cedex, France
marine.minier@insa-lyon.fr

**Abstract.** This article presents some new results concerning two algebraic attacks against the F-FCSR constructions proposed in [2]. We focus on the parameters of the stream ciphers proposed that permit to mount algebraic attacks when using the IV mode. The complexity obtained for the first attack described here is $2^{45}$ binary instructions using $2^{15}$ known IV values for the construction F-FCSR-SF1. All the proposed attacks are full key recovery attacks. We do not contest that the FCSRs are a good and new idea, we just say that the chosen parameters do not ensure the security level claimed.

**Keywords:** Stream cipher, cryptanalysis, algebraic attack.

## 1 Introduction

In [8] and [7], a new class of attacks called "algebraic attacks" was introduced. Those cryptanalyses use the fact that the relation between the initial state constructed in the general case from the key and the internal state at time $t$ is linear. Then an attacker could construct a huge system of equations from the observed output words using the previous remark and he does not have any more but to solve the obtained system.

So finding non linear transition functions for stream ciphers becomes urgent. Some propositions called T-functions were made by A. Klimov and A. Shamir in [12, 13, 14]. An other possible choice proposed in [2] and in [3] is to use an FCSR: a binary automaton with carries. All the results concerning the complexity, the provided period comes from the 2-adic theory. In [2], the authors proposed four constructions (F-FCSR-SF1, F-FCSR-SF8 F-FCSR-DF1 and F-FCSR-DF8) based on a same simple construction called F-FCSR. Two others constructions called F-FCSR-8 and F-FCSR-H based on the same principles were proposed in the call for stream cipher primitives of the European Network of Excellence ECRYPT (see [3] and [17]).

In this paper, we propose two algebraic attacks with known IV values against the F-FCSRs based upon a bad choice of the parameters when using the IV mode

© Springer-Verlag Berlin Heidelberg 2005

for the constructions proposed in [2] (not for the one described in [3]): even if the transition function is not linear, the degree between the key bits and the first output bit is very low. The first attack proposed is a traditional one but the second one uses some particular properties of the structure of the FCSRs. In fact, we could, if we made an exhaustive search on some particular key bits, control and lower the degree between the key bits and the first output bit. These two attacks are full key recovery attacks.

We want to point out one more time that those attacks do not threaten the FCSRs themselves but just shows that the parameters of [2] were not carefully chosen. We do not contest the security level provided by the FCSR (especially against the algebraic attacks), we only claim that the security margin induced by the total construction proposed in [2] is not sufficient. Notice also that two attacks with chosen IVs against the constructions proposed in [2] will be presented at SAC'05 by E. Jaulmes and F. Muller (see [10]). They have also studied the version presented in [3] and published their analyses on the ECRYPT web-site (see [9]).

This paper is organized as follows: after a short recall about the FCSRs themselves and about the constructions proposed in [2] and in [3], Section 2 describes the particular properties used to mount the proposed attacks whereas Section 3 describes the two proposed algebraic attacks.

## 2 Background on the F-FCSRs

The Feedback with Carry Shift Registers were introduced first by Klapper and Goresky in [11]. In [2], T. Berger and F. Arnault proposed to use them as the transition function of a filtered stream cipher. We first recall how an FCSR automaton works. For more details on the F-FCSRs, the reader could refer to [1, 2].

### 2.1 The FCSR Automaton

Let $q$ be a negative integer such as $|q|$ is prime and $p$ be a number such as $0 \leq p < |q|$. Then you could write $p$ as $p = \sum_{i=0}^{n-1} p_i 2^i$ and $d = \frac{1-q}{2} = \sum_{i=0}^{n-1} d_i 2^i$. The FCSR automaton with feedback prime $q$ and an initial value $p$ produces the 2-adic expansion of $p/q$ that could be seen as an infinite sequence of bits $a_i$ such as (see [15]):

$$p = q \cdot \sum_{i=0}^{\infty} a_i 2^i$$

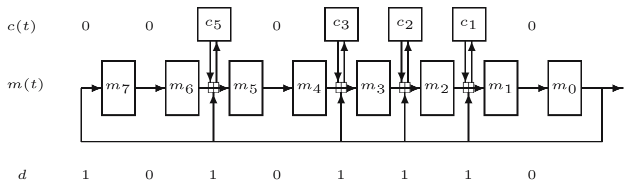
Let us consider the sequence of integers $p(t)$ defined by: $p(0) = p$, $p(t+1) = (p(t) - qa_i)/2$. It is easy to verify that $0 \leq p(t) < -q$ and $p(t)/q = \sum_{j=t}^{\infty} a_j 2^j$.

The sequences $(a_i)$ and $(p(t))$ could be generated from an FCSR automaton defined using two registers (sets of cells): a main register $M$ and a carry register $C$.

The main register $M$ contains $n$ binary cells, each bit is denoted by $m_i(t)$ $(0 \leq i \leq n-1)$. We call the integer $m(t) = \sum_{i=0}^{n-1} m_i(t) 2^i$ the content of $M$.

The carry register contains $\ell$ cells where $\ell + 1$ is the number of nonzero $d_i$ digits, i.e. the Hamming weight of $d$. More precisely, the carry register contains one cell for each nonzero $d_i$ with $0 \leq i \leq n-2$. We denote $c_i(t)$ the binary digit contained in this cell. We put $c_i(t) = 0$ when $d_i = 0$ or when $i = n-1$. We call the integer $c(t) = \sum_{i=0}^{n-2} c_i(t)2^i$ the content of $C$. The Hamming weight of the binary expansion of $c(t)$ is at most $\ell$. Note that, if $d_i = 0$, then $c_i(t) = 0$ for all $t$. We denote by $c_{j_1}(t), \ldots, c_{j_\ell}(t)$ the active carries cells, i.e. the $\ell$ cells corresponding with $d_i = 1$. We have $c(t) = \sum_{i=1}^{\ell} c_{j_i}(t)2^{j_i}$.

A simple example with $q = -347$, $d = 174 = 0\text{xAE}$, $k = 8$ and $\ell = 4$ is described on the figure just bellow.



The symbol $\boxplus$ denotes the addition with carry.

The transition function of the registers could be written

$$m(t+1) = (m(t))_{<<1} \oplus c(t) \oplus m_0(t)d \tag{1}$$
$$c(t+1) = (m(t))_{<<1} \otimes c(t) \oplus c(t) \otimes m_0(t)d \oplus m_0(t)d \otimes (m(t))_{<<1} \tag{2}$$

where $\oplus$ denotes bitwise XOR, $\otimes$ denotes bitwise AND, and $<< 1$ is a simple shift to the left.

Note that $m_0(t)$ is the least significant bit of $m(t)$ and represents the feedback bit. The integers $m(t)$, $c(t)$ and $d$ are integers of bit-size $n$ (or less).

So if $m(0) = p$, at time $t$, the following relations are always satisfied:

$$p(t) = m(t) + 2c(t).$$

The transition function could also be described at the cell level:

$$m_i(t+1) = m_{i+1}(t) \oplus d_i c_i(t) \oplus d_i m_0(t) \tag{3}$$
$$c_i(t+1) = d_i\,(m_{i+1}(t)c_i(t) \oplus c_i(t)m_0(t) \oplus m_0(t)m_{i+1}(t)) \tag{4}$$

The period $T$ of the FCSR automaton is maximal if $|q|$ is prime and the order of 2 modulo $q$ is exactly $|q|-1$. In that case, $T$ is equal to $|q|-1$, so we have: $2^n < T < 2^{n+1} - 1$. The number of the possible states of the FCSR automaton is $2^{n+\ell}$.

In [2], the authors proposed to use the following parameters $n = 128$ and $\ell = 68$ before to apply on the chosen FCSR a filtering function. They choose the prime number $q$ equal to:

$$q_1 = -493877400643443608888382048200783943827$$

In [3], the authors proposed another primitive called F-FCSR-H and designed for hardware utilization with a register length equal to $n = 160$ bits. The corresponding connection integer is:

$$q_2 = -1993524591318275015328041611344215036460140087963$$

that corresponds with $n = 160$ and $\ell = 82$.

## 2.2 The Proposed Constructions

The four constructions proposed in [2] filter some of the bits of the main register with $q = q_1$ in the following way:

- **F-FCSR-SF1:** The filtering function is known and consists of a linear function $f = (f_0, \cdots, f_{n-1})$ on $GF(2)^n$. If $s(t)$ denotes the output bit at time $t$, we have: $s(t) = \bigoplus_{i=0}^{n-1} f_i \cdot m_i(t)$.
- **F-FCSR-SF8:** The filtering function is also known but the aim here is to output one byte, so the filtering function consists in 8 sub-filters $F_0, \cdots, F_7$ on 16 bits linearly independent and publicly known. The output byte $S(t)$ is then the XOR at sixteen bits level between the eight sub-filters and the main register $M$ folded at byte level.
- **F-FCSR-DF1:** This is the same construction as SF1 but, this time, the filter is unknown and derived from the key.
- **F-FCSR-DF8:** This is the same construction as SF8 but, this time, the filter is unknown and the eight sub-filters are derived from the key.

In [3], the authors proposed a first construction called **F-FCSR-8** corresponding with the F-FCSR-DF8 construction. The second construction submitted called **F-FCSR-H** corresponds with the case where $n = 160$, $l = 82$ with the $q$ value equal to $q_2$ and F-FCSR-SF8 (the 8 sub-filters are constructed using the $d$ value) is applied with a key setup and an IV injection defined as follows: $M = K + 2^{80} \cdot IV$. The carry register $C$ is initialized to 0 and 128 iterations are discarded at each IV change (for the details of the used filter see [3]).

In the previous proposed constructions, the initialization of the FCSR using the key $K$ of length $l_K = n$ is $m(0) = K$ and $c(0) = 0$: $p(0) = m(0) + 2 \cdot c(0) = K$.

## 2.3 Description of the IV Mode

An IV mode is also proposed in [2] where the IV value is directly injected in the cells of the carry register at bit level whereas the key is injected in the main register and in the filter if required according the F-FCSR version used. After this initialization, the FCSR is clocked 6 times and the 6-th output bit or byte becomes the first output bit or byte according the version we use.

This article focus on some algebraic attacks using this IV mode: the number of clocks is not sufficient to prevent the stream-cipher from this kind of attacks: the degree of the first output is too small.

## 3 Particular Algebraic Properties of the FCSR Automaton

We focus on this section on several algebraic properties of the FCSR that will be used to mount the cryptanalyses presented in section 4.

### 3.1 Some Results on the Degree and the Number of Monomials of Algebraic Equations

We consider here that at time $t \geq 0$, the main register $M = m(t)$ is composed of $m_i(t)$ with $i \in [0..n-1]$ and the carry register $C = c(t)$ of $c_{j_i}(t)$ with $i \in [0..\ell-1]$. These values $m_i(t)$ and $c_{j_i}(t)$ could be seen as polynomials in the first indeterminates $(m_0(0), \cdots, m_{n-1}(0), c_{j_1}(0), \cdots, c_{j_\ell}(0))$. In order to perform algebraic attacks on the FCSR, we are going to study in this section the degree and the number of monomials occurring in these polynomials.

We denote by $deg(m(t))$ and $deg(c(t))$ the maximum of the degree of each $m_i(t)$, resp. $c_{j_i}(t)$ in terms of the monomials constructed from the unknowns $(m_0(0), \cdots, m_{n-1}(0), c_{j_1}(0), \cdots, c_{j_\ell}(0))$.

**Lemma 1.** *The following relations on the degree are satisfied:*

$$deg(m(t+1)) \leq Max(deg(m(t)), deg(c(t)))$$
$$deg(c(t+1)) \leq Max(deg(m(t)) + deg(c(t)), 2 \cdot deg(m(t)))$$

**Proof:** It is a direct consequence of the equations (1) and (2). □

**Proposition 1.** *We have $deg(m(t)) \leq Fib(t)$ and $deg(c(t)) \leq Fib(t+1)$, $\forall t \geq 1$ where $Fib(t+1)$ is the $(t+1)$-th term of the Fibonacci sequence such as $Fib(0) = 1$ and $Fib(1) = 1$.*

**Proof:** This proof could be made by induction:
At $t = 0$, we have $deg(m(0)) = 1 = Fib(0)$ and $deg(c(0)) = 1 = Fib(1)$.
Now, suppose that the relations $deg(m(t)) \leq Fib(t)$ and $deg(c(t)) \leq Fib(t+1)$ hold for $t$. Using Lemma 1, we deduce

$$deg(m(t+1)) \leq Max(Fib(n), Fib(n+1)) = Fib(n+1)$$
$$deg(c(t+1)) \leq Fib(n) + Fib(n+1) = Fib(n+2)$$ □

This bound is just an upper bound that could only be reached if $d_0 = 1$. In the FCSRs we study, due to the fact that $|q|$ is prime for a security aim (see [1, 2]), $d$ is always even and $d_0 = 0$. So, the degree of $m(t)$ for the FCSR defined using $q_1$ is under this bound.

We are also interested in the number of distinct monomials that can occur in the $m(t)$ and in the $c(t)$ polynomials.

**Proposition 2.** *The polynomials $m_i(t)$ and $c_i(t)$ only depend on the indeterminates $(m_0(0), \cdots, m_{t-1}(0), c_0(0), \cdots, c_{t-1}(0)$ and $(m_{i+1}(0), \cdots, m_{i+t}(0), c_i(0), \cdots, c_{i+t-1}(0))$.*

**Proof:** This result could be obtained by induction using equations (3) and (4). □

The most important consequence of this result is the fact that, even if the degree of an algebraic equation is $s$, this equation does not contain all the monomials of degree less or equal to $s$, but only a small part of them.

It seems difficult to determine exactly the number of monomials given by Proposition 2 but we have computed the degree of algebraic equations and the number of distinct monomials occurring in the main register (i.e. in the polynomials $m_i(t)$ for the value of $q = q_1$ given in [2] and $0 \leq t \leq 6$). For example, if $t = 6$, the degree of $m(t)$ is 10 and the maximal number of monomials of $m(6)$ is 274891. (Note that, due to the complexity of computing the algebraic equations, we were not able to obtain these values for $t \geq 7$.)

### 3.2 Algebraic Equations with Known Carries

We have seen in Section 2.3 that in the IV mode described in [2], the initial contents of carries are known, i.e. $c_{j_1}(0), \cdots, c_{j_\ell}(0)$ become fixed values. From Proposition 2, we deduce the following corollary:

**Corollary 1.** *If the initial contents of carries $c_{j_1}(0), \ldots, c_{j_\ell}(0)$ are known, the polynomials $m_i(t)$ only depend $(m_0(0), \ldots, m_{t-1}(0), m_{i+1}(0), \ldots, m_{i+t}(0))$ for $t \geq 1$. The maximal degree of $m(t)$ satisfies the upper bound deg$(m(t)) \leq 2t$.*

As previously, this upper bound is not reached as soon as $\ell < n$. We have computed the degree and the number of distinct monomials in $m(t)$ for the value of $q = q_1$ given in [2] and $0 \leq t \leq 7$ (see Table 1) and have compared them with the usual upper bound given by the sum of the binomial coefficients $(\sum_{i=0}^{d} C_{128}^{i}$ where $d$ is the degree given in Table 1).

**Table 1.** $m_i(0)$ unknown, $c_i(0)$ known

| nb of iterations | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| nb of monomials | 128 | 129 | 256 | 758 | 2490 | 8830 | 32836 | 125420 |
| Degree in $m_i(0)$ | 1 | 1 | 2 | 3 | 4 | 6 | 8 | 10 |
| Binomial coefficient | 129 | 129 | 8257 | 349633 | 11017633 | $\approx 2^{32}$ | $\approx 2^{40}$ | $\approx 2^{48}$ |

In the second attack presented here, we use a stronger property based on the fact that the knowledge of some feedback bits limits the increase of the degree and the number of monomials. This knowledge is equivalent to those of $m_0(0)$, $m_1(0), \ldots, m_{t-1}(0)$. We suppose now that not only the initial values of the carries are known but also the $t$ values $m_0(0)$, $m_1(0)$, $\ldots, m_{t-1}(0)$ of the main register.

**Proposition 3.** *Suppose that $c_{j_1}(0), \ldots, c_{j_\ell}(0), m_0(0), \ldots, m_{t-1}(0)$ are known. For $1 \leq s \leq t$, the monomials occurring in $m_i(s)$ are those obtained from the set of indeterminates $\{m_{i+1}(0), \cdots, m_{i+t}(0)\}$ and of degree strictly less than $s$. The degree of $m(s)$ satisfies the relation deg$(m(s)) < s$.*

**Table 2.** $m_i(0)$ known for $i := 0$ to 5, $c_i(0)$ known

| nb of iterations $s$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| nb of monomials in $m(s)$ | 123 | 123 | 123 | 244 | 484 | 960 | 1904 |
| $deg(m(s))$ | 1 | 1 | 1 | 2 | 3 | 4 | 5 |

*Moreover, if $I_s$ denotes the set of all possible monomials of $m_i(s)$, then the size of $I_s$ can be computed by the following recurring relations:*
$\#\{I_1\} = \#\{I_2\} = n - t + 1$, *and* $\#\{I_{s+1}\} = 2.\#\{I_s\} - 2^{s-1}$, $\forall s \; / \; 2 \leq s < t$.

**Proof:** The first part of the proposition is a direct consequence of the Proposition 2 considering that the variables $c_{j_1}(0), \cdots, c_{j_\ell}(0)$ and $m_0(0),\cdots,m_{t-1}(0)$ are known.

Using the equations (3) and (4), it is easy to verify that $deg(m(1)) = deg(m(2)) = 1$, and then that the number of possible monomials is $n - t + 1$ (including the constant term 1).

From equations (3) and (4) and from the knowledge of $c_{j_1}(0), \ldots, c_{j_\ell}(0)$ and $m_0(j)$ for $0 \leq j < t$, we deduce that $deg(m(s+1) = deg(c(t)) \leq deg(m(t)) + 1$. It implies that $deg(m(s)) \leq s - 1$ for $1 < s < t$. We also deduce from the same equations that

$$I_1 = I_2 = \{1, m_t(0), m_{t+1}(0), \ldots, m_{n-1}(0)\}.$$

The monomials of $I_s$ are exactly those of the form $m_{i_1}(0)m_{i_2}(0)\cdots m_{i_r}(0)$, with $t \leq i_1 < i_2 < \ldots < i_r < n$, $r < s$ and $i_r - i_1 < s$.

Clearly $I_s$ is a subset of $I_{s+1}$. Moreover, the new monomials of $I_{s+1}$ are obtained in the following way: each monomial $m_{i_1}(0)m_{i_2}(0)\ldots m_{i_r}(0)$ corresponds to a new one $m_{i_1}(0)m_{i_2}(0)\ldots m_{i_r}(0)m_{i_1+s}(0)$. It is possible if and only if $i_1 < n - s$. There are $2^{s-1}$ monomials in $I_d$ such that $i_1 \geq n - s$. This gives the recurring relation $\#\{I_{s+1}\} = 2.\#\{I_s\} - 2^{s-1}$. $\square$

The so obtained bound $b = \#\{I_t\}$ is a good approximation on the number of monomials in the algebraic equations after $t$ iterations.

Table 2 gives the results obtained with $q_1$, $t = 6$, $c_{j_1}(0) = \ldots = c_{j_{68}}(0) = 1$ and $m_0(0) = \ldots = m_5(0) = 1$. (Notice that all the values of the second row of this table reach the bound $\#\{I_s\}$.)

## 4 Algebraic Attacks with Known IV Values

The two attacks presented in this section are attacks with known IV values.

### 4.1 General Principle of an Algebraic Attack

Algebraic attacks were introduced by N. Courtois and W. Meier in [8] and in [7] and exploit the fact that the dependence between the key bits and the internal states at time $t$ is linear when using an LFSR. Suppose, for example, that the key $K$ is directly injected in the first initial state of size $n$ at $t = 0$: $Init_0 = (K_0, \cdots, K_{n-1})$ where $(K_0, \cdots, K_{n-1})$ is the representation of $K$ at bit level. Suppose also that the output bit (or word) $s(t)$ could be written at time $t$: $s(t) = f(L^t(K_0, \cdots, K_{n-1}))$ where $f$ is a boolean function from $GF(2)^m$ into $GF(2)^k$ and $L$ is the linear transition function. Then, you could built a system of equations of degree $deg(f)$ for different $t$ values where the unknown variables are the key bits. It is possible to solve the obtained system using a relinearization technique (see [5]) or a dedicated algorithm using the Gröbner basis (see [4]).

There are many improvements of such techniques: you could find some low degree multiples of $f$ to lower the general degree of the built system (see [16]), try to find a relation using several output words (see [7]) and so on.

If an FCSR is used as a transition function, the problem becomes more difficult due to the fact that this transition function is no more linear. However, if the number of iterations is sufficiently small, the degree of the corresponding system linking the output words and the key bits stays reasonable. Moreover, all the filtering function proposed in [2] and in [3] are linear and do not increase the degree of the system. More formally, the obtained system could be written as: $s(t) = f(T^t(K_0, \cdots, K_{n-1}))$ where $f$ is a linear function from $GF(2)^n$ into $GF(2)^k$ ($k = 1$ or $8$ for the constructions studied here) and where $T$ is the FCSR transition function: the degree of the $t$-th equation depends on the degree of $T^t$. The first equation at time $t = 0$ is linear, the following one is quadratic and the degree increases at each clock according the relation demonstrated in Section 3.

## 4.2 A First Simple Attack

The principle of this attack is very simple: the first output after a change of a known IV gives an algebraic equation which can be computed, since there are only 6 iterations before the first output.

So, suppose that, as described in [2], the initial value of the main register is $m(0) = (m_0(0), \cdots, m_{127}(0)) = (K_0, \cdots, K_{127})$ where each $K_i$, $\forall i \in [0..127]$ denotes a key-bit of the key $K$ and that the initial value of the carry register denoted by $c(0) = (c_0(0), \cdots, c_{67}(0))$ is known and is equal to $IV = (IV_0, \cdots, IV_{67})$. In [2], the first output $s(t) = s(6)$ (that could be a bit or a byte) is computed after six clocks, the previous outputs being discarded. So, we could construct the following simple algebraic attack against all the constructions proposed in [2] when using the IV mode:

- For a subset of $N$ known IV values, compute the first output bit (or byte) $s(6)$ and generate the corresponding system with $N'$ equations.
- Linearize the obtained system and use a Gaussian elimination to solve it.
- When you find a solution, test the obtained key for a known IV by generating few key-stream bits.

So, the complexity of this attack is about $(N')^3$ basic binary instructions. Let us now determinate the required number of known IV values for the four constructions described in Section 2.2.

First, we have seen in Section 3 that the number of possible monomials after 6 clocks, given in Table 1 is $32836 \simeq 2^{15}$. So, in the case of **F-FCSR-SF1**, this number corresponds exactly to the number of unknowns due to the fact that the filter is linear and completely known. So, the complexity of the previous attack is about $2^{45}$ basic binary operations for a number of known IV values equal to $N = N' = 2^{15}$.

For the **F-FCSR-SF8** construction, the number of monomials depending on the key bits is always the same $2^{15}$ but less known IV values are required: each output byte $S(6)$ gives 8 equations. So, the complexity is the same than previously but the number of known IV values required is equal to $N = 2^{12}$ whereas the number of equations is always the same $N' = 2^{15}$.

In the case where the **F-FCSR-DF1** construction is used, the filter is dynamic and constructed from the key. So we could consider it as 128 unknown coefficients denoted by $f_i$, $\forall i \in [0..127]$:

$$s(6) = \bigoplus_{i=0}^{127} f_i \cdot m_i(6).$$

So, taking into account the $f_i$ values as unknowns, the number of monomials is multiplied by a factor 128 and then we need about $N' \simeq 128 \cdot 2^{15} = 2^{22}$ equations generated from $N = 2^{22}$ known IV values for a complexity equal to $2^{22 \cdot 3} = 2^{66}$ basic binary instructions.

In the last case (**F-FCSR-DF8**), where the filter is unknown and derived from the key and where the output is one byte, you could write the output bits in the following way:

$$S_j(6) = \bigoplus_{i=0}^{15} f_{8j+i} \cdot m_{8j+i}(6)$$

for $j = 0, \cdots, 7$.

So if you only consider one output bit (the first one for example), the number of unknowns added is only 16 instead of 128. So, the number of monomials is multiplied by a factor 16 and you need $2^{19}$ known IV values to generate $2^{19}$ equations for a complexity equal to $2^{3 \cdot 19} = 2^{57}$ basic binary instructions.

All the previous results are summed up in Table 3.

**Table 3.** First attack

| Algorithm | Attack | Complexity | Data |
|---|---|---|---|
| F-FCSR-SF1 IV mode | algebraic | $2^{45}$ | $2^{15}$ |
| F-FCSR-DF1 IV mode | algebraic | $2^{66}$ | $2^{22}$ |
| F-FCSR-SF8 IV mode | algebraic | $2^{45}$ | $2^{12}$ |
| F-FCSR-DF8 IV mode | algebraic | $2^{57}$ | $2^{19}$ |

**Table 4.** Second attack

| Algorithm | Attack | Complexity | Data |
|---|---|---|---|
| F-FCSR-SF1 IV mode | exhaust. + alg | $2^{39}$ | $2^{11}$ |
| F-FCSR-DF1 IV mode | exhaust. + alg | $2^{60}$ | $2^{18}$ |
| F-FCSR-SF8 IV mode | exhaust. + alg | $2^{39}$ | $2^{8}$ |
| F-FCSR-DF8 IV mode | exhaust. + alg | $2^{51}$ | $2^{15}$ |

### 4.3 Improving the Previous Attack

We have seen in Section 3.2 that we could lower the degree and so the number of monomials of $m(t)$ by knowing $t$ feedback bits. So, we could improve the previous attack by sharing it in two parts, first we perform an exhaustive search on $t = 6$ feedback bits (i.e. the bits $m_0(0), \cdots, m_5(0)$) by generating, for each value, a system of equations and after by solving this simpler system.

The algorithm is then the following one:

– for each possible value of $m_0(0), \cdots, m_5(0)$ do
– for $N$ known IV values, compute the first corresponding output word $s(6)$ (a bit or a byte).(In case you use F-FCSR-DF8, take only into account the first output bit $S_0(t)$.)
– generate the system of $N'$ equations
– solve the corresponding system by linearization
– when you find a solution, test the obtained key by generating few key-stream bits.

We detail here the complexity of a such attack for F-FCSR-DF1 (the details of the other cases are left to the reader). The number of monomials is $1904 \approx 2^{10.89}$ (c.f. Table 2). So, taking into account the 128 unknown bits of the filter, $N = 128 \cdot 2^{10.89}$, $N' = 128 \cdot 2^{10.89}$ and the total complexity of the previous attack is $2^{3 \cdot 17.89} \cdot 2^6 \approx 2^{60}$ operations considering a resolution with a simple linearization.

The corresponding complexity for F-FCSR-DF8 is $2^{51}$ operations, for F-FCSR-SF1, that corresponds with $2^{39}$ operations and for F-FCSR-SF8 with $2^{39}$ operations.

We have implemented the first attack described (see section 4.2) on a small example to prove its relevance with $q = -112979$ and $d = 56490$. So, the main register $M$ contains 16 binary cells $m(t)$ and the carry register $c(t)$ 8 cells. We consider here that the number of initial clocks is 3. We solve the obtained system using the implementation of the Buchberger algorithm provided by magma 2.9.

**Table 5.** Experimental number of monomials for $n = 16$

| nb of iterations | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| nb of monomials | 16 | 17 | 32 | 86 | 250 | 766 | 2372 | 7148 |
| Degree in $m_i(0)$ | 1 | 1 | 2 | 3 | 4 | 6 | 8 | 10 |

To simplify the resolution we consider here that the unknown filtering variables are directly the key bits. Solve the obtained system takes about fifteen minutes on a Pentium 4 and gives 328 possible solutions including the good one: $K =$0xdde5.

More, when we compute the number of exact monomials for the previous example, we obtain the following results.

### 4.4 Could We Apply This Attack on F-FCSR-8 and on F-FCSR-H ?

Those attacks especially the second one could not be applied on the two versions proposed for the ECRYPT call for stream cipher primitives [17] (see [3]) due to a greater number of clocks before the first output generation.

To prevent the FCSR constructions using an IV mode from the algebraic attacks proposed in this paper, the required number of clocks $t$ before generating an output must verify the following inequality $2^n > 2^t \cdot (\#\{I_t\})^{2.37}$ where 2.37 is the coefficient of the resolution of a linear system given in [6] and where $\#\{I_t\}$ is the cardinal of the set defined in Proposition 3. For example, if $n = 128$, the minimal number of initial clocks must be at least equal to 34. If $n = 160$ for a key length equal to 80 bits, this lower bound is equal to 20.

The new parameters chosen for F-FCSR-8 (F-FCSR-DF8 using 64 initial clocks instead of 6, the number of monomials given by Proposition 3 is then close to $2^{64}$) and F-FCSR-H (the construction presented in Section 2.2 with 160 initial clocks and $n = 160$ for a key length equal to 80 bits, then the number of possible monomials is $2^{80}$) described in the ECRYPT submission (see [3] for more details) verifies the previous conditions and prevent the new proposed FCSR constructions from the two attacks described in this paper that become more expansive than the exhaustive key search.

In the estimation of the number of monomials made here, we do not take into account the time required to compute all those algebraic equations (we do not evaluate the corresponding complexity) but experimentally, it seems that this complexity becomes greater than the resolution of the system itself for more than 10 iterations.

## 5 Conclusion

We present in this paper two algebraic attacks against the F-FCSR constructions proposed in [2] based on some bad choices in the stream-cipher parameters but also on some particular algebraic properties of the FCSRs described here.

We do not contest the security level provided by the FCSR, we only claim that the security margin induced by the total construction proposed in [2] is not sufficient. The proposed parameters (size of the FCSR, number of clocks before the first output,...) must be enlarged. This is what have been done in the constructions submitted to the ECRYPT call for stream ciphers [3]. However, some other attacks could be applied on those two new versions as noticed by E. Jaulmes and F. Muller in [9].

## References

1. F. Arnault and T.P. Berger. Design and properties of a new pseudo-random generator based on a filtered FCSR automaton. In *IEEE, Transactions on Computers*, 2005. To appear.
2. F. Arnault and T.P. Berger. F-FCSR: design of a new class of stream ciphers. In *Fast Software Encryption - FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 83–97. Springer-Verlag, 2005.
3. F. Arnault, T.P. Berger, and C. Lauradoux. The FCSR: primitive specification and supporting documentation. ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives 2005. `http://www.ecrypt.eu.org/stream/`.
4. G. Ars and J.-C. Faugère. An algebraic cryptanalysis of nonlinear filter generators using Gröbner bases. Research Report INRIA Lorraine, number 4739, 2003.
5. D. Coppersmith and S. Winograd. On the asymptotic complexity of matrix multiplication. *SIAM Journal on Computing*, 11(3):472–492, August 1982.
6. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic programming. *Journal of Symbolic Computation*, 9(3):251–280, 1990.
7. N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 177–194. Springer-Verlag, 2003.
8. N. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer-Verlag, 2003.
9. E. Jaulmes and F. Muller. Cryptanalysis of ecrypt candidates F-FCSR-8 and F-FCSR-H. ECRYPT Stream Cipher Project Report 2005/046, 2005. `http://www.ecrypt.eu.org/stream`.
10. E. Jaulmes and F. Muller. Cryptanalysis of the F-FSCR stream cipher family. In *Selected Areas in Cryptography - SAC 2005*, *Lecture Notes in Computer Science*. Springer-Verlag, 2005. To appear.
11. A. Klapper and M. Goresky. 2-adic shift registers. In *Fast Software Encryption - FSE'93*, volume 809 of *Lecture Notes in Computer Science*, pages 174–178. Springer-Verlag, 1993.
12. A. Klimov and A. Shamir. A new class of invertible mappings. In *CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 470–483. Springer-Verlag, 2002.
13. A. Klimov and A. Shamir. Cryptographic applications of T-functions. In *Selected Areas in Cryptography - SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 248–261. Springer-Verlag, 2004.
14. A. Klimov and A. Shamir. New applications of T-functions in block ciphers and hash functions. In *Fast Software Encryption - FSE'05*, Lecture Notes in Computer Science, pages 19–32. Springer-Verlag, 2005. to appear.
15. N. Koblitz. p-adic numbers, p-adic analysis and zeta-functions. Springer-Verlag, 1997.
16. W. Meier, E. Pasalic, and C. Carlet. Algebraic attacks and decomposition of Boolean functions. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. Springer-Verlag, 2004.
17. Network of Excellence in Cryptology ECRYPT. Call for stream cipher primitives. `http://www.ecrypt.eu.org/stream/`.

# Cryptanalysis of Keystream Generator by Decimated Sample Based Algebraic and Fast Correlation Attacks

Miodrag J. Mihaljević[1], Marc P.C. Fossorier[2], and Hideki Imai[3]

[1] Mathematical Institute,Serbian Academy of Sciences and Arts,
Kneza Mihaila 35, 11001 Belgrade, Serbia and Montenegro
miodragm@turing.mi.sanu.ac.yu

[2] Department of Electrical Engineering, University of Hawaii,
2540 Dole St., Holmes Hall 483, Honolulu, HI 96822, USA
marc@spectra.eng.hawaii.edu

[3] University of Tokyo,Institute of Industrial Science,
4-6-1, Komaba,Meguro-ku, Tokyo, 153-8505, Japan
imai@iis.u-tokyo.ac.jp

**Abstract.** This paper proposes a novel approach for cryptanalysis of keystream generators consisting of the composition of a linear finite state machine (LFSM) and nonlinear mapping. The proposed approach includes a dedicated decimation of the sample for cryptanalysis based on the following: Suppose certain $B$ bits of the LFSM initial state as known and identify time instances where certain arguments of the nonlinear function depend only on these $B$ bits and are equal to zero. As opposed to previously reported methods, the proposed one also identifies and uses certain characteristics of the LFSM state-transition matrix in order to reduce the nonlinearity of the system of overdefined equations employed in an algebraic attack scenario, or to reduce the noise introduced by the linearization of the nonlinear function which corrupts the linear equations employed in a correlation attack scenario. The proposed method is employed for developing efficient algorithms for cryptanalysis of the nonlinear combination keystream generator reported at INDOCRYPT 2004.

**Keywords:** Stream ciphers, nonlinear combination keystream generator, state transition matrix, LFSRs, algebraic attacks, fast correlation attack, decimation.

## 1 Introduction

This paper points out novel algebraic and correlation attack techniques for cryptanalysis of certain keystream generators for stream ciphers known as the nonlinear combination generators (see [11], for example), and shows the insecurity of a particular keystream generator from this class reported in [4].

A general paradigm of the algebraic and correlation attacks is based on establishing and processing a system of overdefined equations which are: (i) nonlinear

© Springer-Verlag Berlin Heidelberg 2005

and (mainly) error free in the case of algebraic attacks; (ii) linear and (very) noisy in the case of correlation attacks (assuming that a noisy equation denotes an equation which is satisfied with a certain, known, probability). Some early algebraic attacks on stream and related ciphers have been reported in [6] as well as in [17] and [18]. Very recently, a number of algebraic attacks have been reported in [7], [8], [13], [1], [9] and [19]. All contemporary fast correlation attacks originate from [12] and they follow block or convolutional codes decoding approaches. An overview of contemporary fast correlation attacks is available in [10]. The reported iterative block decoding approaches include [14], [15], and the non-iterative approaches include these reported in [16] and [5], for example. The convolutional code based approaches for fast correlation attack have been considered in a number of papers including the recent one [20].

*Motivation for the work.*
The general powerful algebraic attacks that have been recently reported are based on the construction of an overdefined system of nonlinear equations employing only certain characteristics of the nonlinear function. Accordingly, the performance of these attacks strongly depends on the nonlinear part, and if this part does not have certain characteristics appropriate for cryptanalysis, the attacks could become very complex or even not feasible. A goal of this paper is to address the following issue: Find a way to involve into the algebraic attack certain characteristics of the linear part in order to obtain more powerful attacks against certain nonlinear functions (which could be heavily resistant against the reported algebraic attacks). An additional motivation for this work was a complementary extension of the algebraic attack approach reported in [19].

The paradigm of contemporary fast correlation attacks could be considered as consisting of the following main steps: (i) assuming that certain secret key bits are known, specification of an overdefined system of noisy linear equations; (ii) solving the specified system as a decoding problem via hypothesis testing and evaluation of parity checks. The noise involved in the system of equations is a consequence of linearization of a nonlinear system of equations which describes the considered stream cipher. As a result, this noise is not an usual random one and accordingly could be an objective of adjustment attempts. Accordingly, the motivations for this work include consideration of the possibilities for specifying the systems of equations with a noise level lower than the one obtained by a simple straightforward linearization of the initial system of nonlinear equations related to the nonlinear filter.

Finally, a motivation for this work was to show incorrect security claims regarding the keystream generator reported at INDOCRYPT 2004, [4] and to provide a warning regarding misleading examples of the Boolean functions with generalized cryptographic properties [4].

*Contributions of the paper.*
This paper proposes novel approaches for developing fast algebraic and correlation attacks on certain nonlinear combination keystream generators.

Regarding algebraic attacks, a novel approach for constructing the overdefined system of binary nonlinear algebraic equations with reduced nonlinearity relevant for cryptanalysis is pointed out. The main feature of the novel approach is that it employs certain characteristics of the generator linear part (the involved LFSM) to identify positions of the generator output sequence where certain arguments of the nonlinear function depend only on a part of the generator initial state and are equal to zero. Assuming that this part of the initial state can be considered as known, a more suitable system of nonlinear multivariate equations (with reduced nonlinearity) can be established. The assumed initial state part can be recovered via exhaustive hypothesis testing later on.

Regarding the fast correlation attack, following the above dedicated decimation approach, this paper yields a technique for reducing the noise which corrupts the parity-checks. This provides reduction of the processing complexity, usually at the expense of a longer sample required. The noise reduction originates from identification of suitable time instances for the modelling and linearization. The linearization is based on a sophisticated involvement of certain LFSMs and the nonlinear function characteristics.

The proposed frameworks are employed for attacking the nonlinear combination keystream generator consisting of 5 LFSRs (total length 209) and a novel nonlinear combining function reported at INDOCRYPT 2004 [4]. It is shown that the dedicated decimation based algebraic and correlation attacks can efficiently break the reported keystream generator and that they appear as the more powerful ones in comparison with the previously reported algebraic and fast correlation attacks. The explicit claim from [4] that the security parameter of the proposed generator against fast correlation attacks is equal to 60, becomes incorrect as the developed algorithm for fast correlation attack based on the dedicated sample decimation contradicts this statement - its main features are: (i) pre-processing time complexity $\sim 2^{56}$; (ii) processing time complexity $\sim 2^{44}$; (iii) required sample $\sim 2^{53}$; (iv) required memory $\sim 2^{49}$. Furthermore, for this system, the developed dedicated sample decimation based algebraic algorithm for cryptanalysis is even more efficient: (i) pre-processing time complexity $\sim 2^{39}$; (ii) processing time complexity $\sim 2^{29}$; (iii) required sample $\sim 2^{39}$; (iv) required memory $\sim 2^{36}$. The above results show that the proposed keystream generator is entirely insecure, that the explicit security claims regarding the generator are incorrect, and that, actually, no one appropriate example of the proposed Boolean functions with "generalized cryptographic properties" is given.

## 2 Basic Framework of the Dedicated Sample Decimation Based Algebraic and Fast Correlation Attacks

This section introduces the main ideas for developing certain algebraic and fast correlation attacks based on the dedicated decimation of the sample given for cryptanalysis.

### 2.1 Model of the Keystream Generators Under Consideration

We consider keystream generators belonging to the class of nonlinear combination generators which consist of a number of LFSMs whose outputs are combined by a nonlinear Boolean function, and a similar consideration holds for the nonlinear filter (for more details regarding these keystream generators see [11], for example).

A binary linear finite state machine (LFSM) can be described as $\mathbf{X}_t = \mathbf{A}\mathbf{X}_{t-1}$, where $\mathbf{A}$ is the **state transition matrix** (over GF(2)) of the considered LFSM. Let $\mathbf{X}_0$ be the column $(L \times 1)$ matrix $[X_{L-1}, ..., X_0]^T$ representing the initial contents or initial state of the LFSM, and let $\mathbf{X}_t = [X_{L-1}^{(t)}, ..., X_0^{(t)}]^T$ , be the $L$-dimensional column vector over GF(2) representing the LFSM state after $t$ clocks, where $\mathbf{X}^T$ denotes the transpose of the $L$-dimensional vector $\mathbf{X}$. We define

$$\mathbf{X}_t = \mathbf{A}\mathbf{X}_{t-1} = \mathbf{A}^t\mathbf{X}_0, \quad \mathbf{A}^t = \begin{bmatrix} \mathbf{A}_1^{(t)} \\ \cdot \\ \mathbf{A}_L^{(t)} \end{bmatrix}, \quad t = 1, 2, \ldots, \tag{1}$$

where $\mathbf{A}^t$ is the $t$-th power over GF(2) of the $L \times L$ state transition binary matrix $\mathbf{A}$, and each $\mathbf{A}_i^{(t)}$, $i = 1, 2, ..., L$, represents a $1 \times L$ matrix (a row-vector).

Let LFSM$^{(k)}$, denotes a known LFSM with only the initial state $\mathbf{X}_0^{(k)}$, $k = 1, 2, ..., K$, determined by the secret key, and $f(\cdot)$ is a known nonlinear memoryless function of $K$ arguments generated by the corresponding LFSMs.

The model assumes that the state $\mathbf{X}_t$ of the generator is the union of the component states corresponding to the involved LFSMs, $\mathbf{X}_t = \bigcup_{k=1}^{K} \mathbf{X}_t^{(k)}$ .

According to (1), the output $x_k(t)$ of LFSM$^{(k)}$, is given as

$$x_k(t) = \mathbf{A}_1^{(k,t)}\mathbf{X}_0^{(k)} , \tag{2}$$

where $\mathbf{A}_1^{(k,t)}$ denotes the first row of the $t$-th power of the state transition matrix $\mathbf{A}^{(k)}$, and $\mathbf{X}_0^{(k)}$ is its initial state, $k = 1, 2, ..., K$.

Finally, the output bit $z(t)$ of the considered keystream generator at the time instance $t$ is determined by $z(t) = f(.)$,

$$f(x_1, x_2, ..., x_K) = a_0 \oplus_{1 \le i \le K} a_i x_i \oplus_{1 \le i < j \le K} a_{ij} x_i x_j \oplus ... \oplus a_{12...K} x_1 x_2 ... x_K \ , \tag{3}$$

where the coefficients $a_0$, $a_i$, $a_{ij}$, ..., $a_{12...K} \in$ GF(2), and for simplicity $x_k$ stands for $x_k(t)$.

## 2.2 Underlying Ideas for the Decimated Sample Based Attacking

The attack consists of the following two main phases:

- *Pre-Processing*: Assuming that certain subset of the secret bits is known, decimate the sample so that at the selected points the nonlinear function degenerate to a more suitable one for the cryptanalysis.
- *Processing*: Perform the main steps of cryptanalysis taking into account only the sample elements selected in the pre-processing phase.

Note the following issues regarding the above proposed basic framework:

– Implementation of the framework includes a preprocessing phase which is independent of a particular sample (i.e. it should be done only once), and a processing phase which recovers the secret key based on the given sample.
– Assuming a nonlinear function suitable for the proposed attack, the gain in the processing phase is a consequence of the following:
  - a (highly) reduced nonlinearity of the related system of equations in the case of algebraic attacks;
  - a (highly) reduced correlation noise in the case of fast correlation attacks.

In a particular case we employ an approach based on the following assumption and its consequences.

**Assumption 1.** For a given pattern of $B$ elements of $\mathbf{X}_0$ with indices $i$, $i \in \mathcal{I}$, at certain time instances $t \in \mathcal{T}$, the following is valid:

• For each $k \in \mathcal{K}^*$, $\mathcal{K}^* \subset \mathcal{K}$, the output of LFSM$^{(k)}$ at $t \in \mathcal{T}$ is equal to zero.

Let the cardinality of $\mathcal{K}^*$ be $K^*$. Let $|\mathcal{T}|$ denotes the cardinality of $\mathcal{T}$, i.e. the total number of the time instances when Assumption 1 holds. The following statement can be readily proved: The content of $\mathcal{T}$ depends on the LFSMs involved and the assumed $B$ bits, and can be obtained by a straightforward evaluation employing (1)-(2).

The proposed attacking framework does not yield gain in all but in certain scenarios: On the other hand, it could be considered as a design guideline in order to avoid constructions vulnerable by the related attacking techniques.

## 2.3 Framework of the Decimated Sample Based Fast Correlation Attack

We propose the following framework for developing fast correlation attacks based on the dedicated sample decimation against the nonlinear combination keystream generators.

– *Pre-Processing*
  - Identify conditions which imply that the employed nonlinear function $f(\cdot)$ reduce to one which can be approximated with a linear one introducing the approximation (correlation) noise lower than in a case of the direct approximation.
  - According to the identified conditions, in a general case perform a search in order to determine an appropriate sampling where the Assumption 1 can be fulfilled.
– *Processing*
  - Perform appropriate fast correlation attack over the decimated sample where the correlation noise is reduced.

### 2.4 Framework of the Decimated Sample Based Algebraic Attack

We propose the following framework for developing algebraic attacks based on the dedicated sample decimation against the nonlinear combination keystream generators.

- *Pre-Processing*
  - Identify conditions which imply that the employed nonlinear function $f(\cdot)$ reduce to a function $f^*(\cdot)$ which has a lower algebraic degree.
  - According to the identified conditions, in a general case perform a search in order to determine an appropriate sampling where the Assumption 1 can be fulfilled.
- *Processing*
  - Perform appropriate algebraic attack over the decimated sample where the involved nonlinear function is $f^*(\cdot)$.

## 3 The Keystream Generator Proposed at INDOCRYPT 2004

The nonlinear combination generator recently proposed in [4], Section 3.5, consists of the following:

• 5 LFSRs: LFSR1, LFSR2, LFSR3, LFSR4 and LFSR5 of lengths $L1 = 61$, $L2 = 63$, $L3 = 21$, $L4 = 31$, and $L5 = 33$, respectively, combined via
• nonlinear function

$$f(x_1, x_2, x_3, x_4, x_5) = x_2x_3x_4x_5 \oplus x_1x_2x_3 \oplus x_1x_4 \oplus x_3x_5 \oplus x_1 \oplus x_2 \quad , \qquad (4)$$

and the generator is displayed in Fig. 1

It is claimed that due to the employed nonlinear Boolean function the generator is resistant against the known fast correlation attacks assuming "the security parameter equal to 60".

In the next two sections it will be shown that the generator reported in [4] is breakable employing the proposed framework for dedicated decimation based correlation and algebraic attacks. The algorithms developed for cryptanalysis of the

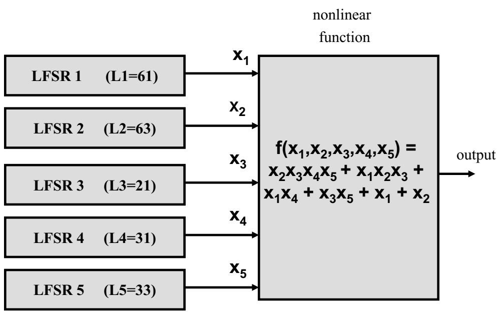


**Fig. 1.** Nonlinear combination keystream generator proposed in [4]

above keystream generator follow the proposed framework, but they also include the "customization", as well, regarding the considered particular keystream generator.

Finally, although it is claimed in [4] that the parameters of the reported keystream generator should be scaled up for a real use, this scaling up does not help regarding the problem of effective key size versus the formal one: namely the effective key size always remains much smaller than the formal one. Particularly, if the total length of the involved LFSRs is increased from 209 to 256 corresponding to a typical secret key size of 256 bits, and the corresponding scaling up of each LFSR is employed, the scheme again suffer from the same weakness and the cryptanalysis is feasible even if the parameters are scaled up for 50% (corresponding to the key of 314 bits).

## 4 A Cryptanalysis Employing Novel Correlation Approach

The dedicated decimation based correlation attacking approach against the keystream generator [4] assumes identification of the time instances $t$ at which a certain subset of the LFSRs outputs $x_1(t)$, $x_2(t)$, $x_3(t)$, $x_4(t)$, $x_5(t)$, equal to zero, so that after replacement in the nonlinear combination function $f(\cdot)$ a much lower approximation-correlation noise is obtained in comparison with the corresponding straightforward linearization.

Note that when the arguments $x_1$ and $x_3$ of $f(\cdot)$ are equal to zero the function reduces to $f^*(\cdot) = x_2$ which can considered as the most favorable case in which the source function is replaced by one of its inputs resulting in a correlation noise equal to zero, and accordingly yielding the direct recovering of LFSR2 initial state. Employing similar and related correlation considerations, all the initial states can be recovered.

### 4.1 Algorithm for Cryptanalysis

This section proposes a basic decimated sample based correlation attack (Algorithm I), and a time-memory trade-off based version of this approach (Algorithm II).

*Algorithm I*

- *Pre-Processing*
  1. Assume that certain $B < L1$ bits of the initial state of LFSR1 are known; search over the powers of LFSR1 state transition matrix, and identify a set $\mathcal{T}$ of the time instances where $x_1(t)$ depends only on the assumed $B$ bits;
  2. Record a set $\mathcal{T}$ of dimension $I = O(L2)$ (typically $I = 4(L2 + \Delta)$, $\Delta \sim L2$).
- *Processing*
  1. Assume previously not considered $B$ bits of LFSR1 initial state and the entire initial state of LFSR3, and construct a subset $\tau_B$ of $\mathcal{T}$ such that $x_1(t) = x_3(t) = 0$, $t \in \tau_B$.

2. Recover the initial state candidate of LFSR2 solving the system of $L2$ linear equations relating, via the state-transition matrix powers, the initial state bits and the given generator outputs $\{z(t)\}_{t\in\tau_B}$.
3. Set the recovered candidate as the initial states of LFSR2 and generate the corresponding generator output sequence $\{\hat{z}(t)\}$, $t \in \tau_B$.
4. If $\{\hat{z}(t)\} = \{z(t)\}$, $t \in \tau_B$, accept the current candidates for the initial states of LFSR3 and LFSR2 and $B$ bits of LFSR1 as the correct ones; Otherwise go to the step 1 and continue processing.
5. Employing the recovered initial states of LFSR2 and LFSR3, and the $B$ bits of LFSR1, consider the time instances $t$ where $x_2(t) = x_3(t) = 0$ so that
$$x_1(t)x_4(t) \oplus x_1(t) = z(t) \ , \tag{5}$$
and recover the remaining $L1 - B$ initial state bits of LFSR1 employing a fast correlation attack technique as follows.
   (a) Assume that
   $$x_1(t) \oplus e(t) = z(t) \ , \tag{6}$$
   where $e(t)$ is a realization of a random binary variable which takes value 1 with the probability $p = 0.25$ (i.e. the correlation noise is equal to 0.25).
   (b) Recover the $j$-th bit, $j = B+1, B+2, ..., L1$, of LFSR1 initial state as follows:
      i. For each $j$, $j = B+1, B+2, ..., L1$, collect $M$ powers $t_{j,m}$, $m = 1, 2, ..., M$, of LFSR1 state-transition matrix such that in its first row there are all zeros on the positions $B+1$ to $L1$ except 1 on the $j$-th position, and there is an arbitrary pattern on the positions 1 to $B$.
      ii. Based on the collected powers, specify $M$ parity-check equations related the $j$-th LFSR initial state bit and the corresponding $t_{j,m}$-th output bit, $m = 1, 2, ..., M$.
      iii. Recover the $j$-th bit of LFSR1 state employing the maximum likelihood decoding for given $M$ parity-checks.
6. Assuming that the current initial states of LFSR1, LFSR2, and LFSR3 are correct, taking into account the time instances $t$ such that $x_1(t) = x_2(t) = 0$ and $x_3(t) = 1$, recover unknown LFSR5 initial state via the following $x_5(t) = z(t)$.
7. Assuming that the current initial states of LFSR1, LFSR2, LFSR3 and LFSR5 are correct, taking into account the time instances $t = 1, 2, ...,$, directly recover the initial state of LFSR4 based on
$f(x_1(t), x_2(t), x_3(t), x_4(t), x_5(t)) = z(t)$.

The above basic dedicated sample decimation based fast correlation attack can be modified in a number of ways in order to provide more efficiency in certain implementation scenarios.

*Remarks.* (i) Steps 1 and 2 of the processing phase resembles the approach employed in [3]. (ii) Additional gains in Algorithm I can possibly be achieved considering more sophisticated fast correlation attack approaches as in [5].

A particular variant of Algorithm I which yields a time-memory trade off and a reduced time processing complexity at the expense of increased space complexity is given in bellow. The main difference is in the pre-processing of the required Gaussian elimination to retrieve the initial state of LFSR2.

*Algorithm II*

The differences between Algorithm II and Algorithm I are as follows.

– *Pre-Processing*
  After the pre-processing steps 1 and 2 of Algorithm I, the following ones are added.
  - For each of $2^{L3} - 1$ possibilities assume the entire initial state of LFSR3, construct a subset $\tau_B$ of $\mathcal{T}$ such that $x_3(t) = x_1(t) = 0$, $t \in \tau_B$ and do the following.
    * Determine the initial states of LFSR2 based on the equations $x_2(t) = Z(t)$ and the corresponding $t$-th powers of LFSR2 state-transition matrix, $t \in \tau_B$ , as a function of the variables $\{Z(t)\}_{t \in \tau_B}$ corresponding to the $L2$ first outputs from the generator assuming that the cardinality of $\tau_B$ is equal to $L2 + \Delta$;
    * Subsequently determine the general expression of the corresponding outputs for the last $\Delta$ time instances $t$, $t \in \tau_B$;
    * Record the above determined expressions for the initial state of LFSR2 and its corresponding outputs for the last $\Delta$ time instances $t$, $t \in \tau_B$.

– *Processing*
  Instead of the processing steps 1 - 3 of Algorithm I, the following ones are employed.
  - Assume a previously not considered hypothesis on $B$ bits of LFSR1 initial state and the entire initial state of LFSR3;
    * For $j = 1, 2, ..., L2$ do the following:
    * Evaluate the LFSR2 output bit employing the given sample $\{z_t\}_t$ and the preprocessed expression for LFSR2 output bit at time instance $t$ corresponding to the position $L2 + j$ in the increasing order of elements of $\tau_B$;
    * Reject the current hypothesis if the the evaluated value is not equal to the corresponding sample value.
  - Accept as the LFSR2 initial state the candidate corresponding to the not rejected hypothesis during the previous processing step.

Note that since $L3 + B < \Delta$, $\Delta \sim L2$, we assume that only one hypothesis will remain valid.

*Remark.* The pre-processing in Algorithm II regarding construction of the general solutions is similar to the pre-processing which yields the general inversion employed in [2].

## 4.2 Main Characteristics of the Proposed Correlation Cryptanalysis

According to Algorithm I and Algorithm II steps, it can be directly proved that the main features of these algorithms are given by the following propositions. Regarding the proofs of these statements, as well as for the other proofs, note the following: (i) factors of the form $2^X$ correspond to certain exhaustive searches; (ii) factors of the form $X^\omega$ correspond to the complexity of the Gaussian elimination; (iii) factors of the form $X^2$ correspond to the complexity of evaluation the $X$ general solutions for given particular values.

**Proposition 1.** Employment of Algorithms I or II assumes that the expected required sample is $O(2^{L1-B}L2)$.

**Proposition 2.** When Algorithm I is employed, the expected time complexity of pre-processing is $O(2^{L1-B}L2)$, and the expected space complexity is $O(L2)$.

When Algorithm II is employed, the expected time complexity of pre-processing is $\max\{O(2^{L1-B}L2), O(2^{B+L3}L2^\omega)\}$, and the expected space complexity is $O(2^{B+L3}L2^2)$.

**Proposition 3.** When Algorithm I is employed, the expected time complexity of the processing is $2^{B+L3}O(L2^\omega) + O((L1-B)M + L5^\omega) + O(L4)$, where $\omega = 2.7$, $M \leq 10$, and the expected space complexity is $O(L2)$.

When Algorithm II is employed, the expected time complexity of the processing is $2^{B+L3}O(L2) + O((L1-B)M + L5^\omega) + O(L4)$, where $\omega = 2.7$, $M \leq 10$, and the expected space complexity is $O(2^{B+L3}L2^2)$.

**Table 1.** Numerical summary of the requirements for cryptanalysis of the nonlinear combination keystream generator from [4] employing proposed correlation attacks

| | pre-processing time complexity | processing time complexity | required sample | required memory |
|---|---|---|---|---|
| proposed Algorithm I, $B = 17$ | $\sim 2^{53}$ | $\sim 2^{53}$ | $\sim 2^{53}$ | $\sim 2^{8}$ |
| proposed Algorithm II, $B = 17$ | $\sim 2^{56}$ | $\sim 2^{44}$ | $\sim 2^{53}$ | $\sim 2^{49}$ |

According to the Propositions 1 - 3, Table 1 yields a numerical summary of the attacking requirements of the proposed correlation based techniques for cryptanalysis. Note that different trade-offs, suitable for the particular scenarios, are possible between the pre-processing time complexity, processing time complexity, required memory and required sample. As opposed to the claims in [4] Table 1 implies the breakability of the considered keystream generator employing the developed dedicated decimation based correlation attack.

## 4.3 A Discussion on Reported Fast Correlation Attacks

It is correctly claimed in [4], that the best reported fast correlation attacks are not feasible because their direct employment require simultaneous consideration of two LFSRs, LFSR1 and LFSR2, of total length $L1 + L2 = 124$ and this is too long for feasible attacking (see [16] and [5], for example).

On the other hand if we have knowledge about the LFSR1 initial state, it is possible to mount an appropriate fast correlation attack in order to recover LFSR2 initial state, but the assumption on the LFSR1 initial state availability has cost proportional to its recovery via an exhaustive search, i.e. $2^{61} - 1$, implying the security parameter equal to 60, as claimed in [4].

## 5 A Cryptanalysis Employing Novel Algebraic Approach

The dedicated decimation based algebraic approach against the keystream generator [4] assumes identification of the time instances $t$ at which a certain subset of the LFSRs outputs $x_1(t)$, $x_2(t)$, $x_3(t)$, $x_4(t)$, $x_5(t)$, is equal to zero so that the combination function $f(\cdot)$ reduces to one with much lower algebraic degree.

Note that when the arguments $x_3$ and $x_4$ of $f(\cdot)$ are equal to zero the function reduces to $f^*(\cdot) = x_1 \oplus x_2$ with $d^* = 1$ which is very suitable for algebraic attacks.

### 5.1 Algorithm for Algebraic Cryptanalysis

The following is a basic form of the algorithm which performs the cryptanalysis employing the dedicated decimation based algebraic attack.

*Algorithm III*

- *Pre-Processing*
  1. Assuming that certain $B < L4$ bits of the initial state of LFSR4 are known, search over the powers of LFSR4 state transition matrix, and identify a set $\mathcal{T}$ of the time instances where $x_4(t)$ depends only od the bits considered as known.
  2. Record a set $\mathcal{T}$ of dimension $I = 4(L1 + L2 + \Delta)$, $\Delta \sim L1 + L2$.
  3. For each of $2^{L3} - 1$ possibilities assume the entire initial state of LFSR3, construct a subset $\tau_B$ of $\mathcal{T}$ such that $x_3(t) = x_4(t) = 0$, $t \in \tau_B$, and do the following.
     - Determine the initial states of LFSR1 and LFSR2 based on the set of equations
       $$x_1(t) \oplus x_2(t) = Z(t) \ , \ t \in \tau_B \ , \qquad (7)$$
       as a function of the variables $\{Z(t)\}_{t \in \tau_B}$ corresponding to the $L1+L2$ first outputs from the generator included in $\tau_B$;
     - Subsequently determine the general expression of the corresponding outputs of LFSR1 and LFSR2 for the last $L1 + L2$ time instances $t$, $t \in \tau_B$.
     - Record the above determined expressions for the initial states of LFSR1 and LFSR2 and theirs corresponding outputs for the last $L1 + L2$ time instances $t$, $t \in \tau_B$.

– *Processing*
  1. Assume a previously not considered hypothesis on $B$ bits of LFSR4 initial state and the entire initial state of LFSR3.
  2. For $j = 1, 2, ..., \Delta$ do the following:
     - evaluate the LFSR1 and LFSR2 output bits employing the given sample $\{z_t\}_t$ and the preprocessed expressions for LFSR1 and LFSR2 output bits at time instance $t$ corresponding to the position $L1+L2+j$ in the increasing order of elements of $\tau_B$;
     - Reject the current hypothesis if mod2 sum of the the evaluated values is not equal to the corresponding sample value.
  3. Accept as the LFSR1 and LFSR2 initial states the candidates corresponding to the not rejected hypothesis during the previous processing step.
  4. Assuming that the current initial states of LFSR1, LFSR2 and LFSR3, and the $B$ bits of LFSR4 are correct and taking into account that $x_4(t) = 0$, $t \in \mathcal{T}_\mathcal{B}$, recover LFSR5 initial state via the following set of equations

$$x_1(t)x_2(t)x_3(t) \oplus x_3(t)x_5(t) \oplus x_1(t) \oplus x_2(t) = z(t) \ , \ t \in \mathcal{T}_\mathcal{B} \ , \tag{8}$$

     where only $\{x_5(t)\}_{t \in \mathcal{T}_\mathcal{B}}$, is unknown.
  5. Assuming that the current initial states of LFSR1, LFSR2, LFSR3 and LFSR5, and the $B$ bits of LFSR4 are correct, recover unknown LFSR4 initial state part via the following set of equations

$$x_2(t)x_3(t)x_4(t)x_5(t) \oplus x_1(t)x_2(t)x_3(t) \oplus \\ x_1(t)x_4(t) \oplus x_3(t)x_5(t) \oplus x_1(t) \oplus x_2(t) = z(t) \ . \tag{9}$$

### 5.2 Main Characteristics of the Proposed Algebraic Cryptanalysis

According to the Algorithm III structure, it can be directly proved that the main features of Algorithm III are given by the following propositions.

**Proposition 4.** The expected required sample is $O((L1 + L2)2^{L4-B})$.

**Proposition 5.** The expected time complexity of pre-processing is $O(2^{B+L3}(L1+L2)^\omega) + O((L1 + L2)2^{L4-B})$, where $\omega = 2.7$, and the expected space complexity is $O(2^{B+L3}(L1 + L2)^2)$.

**Proposition 6.** The expected time complexity of the processing is $2^{B+L3}O(L1+L2)+O(L5^\omega))+O((L4-B)^\omega)$, where $\omega = 2.7$, and the expected space complexity is $O(2^{B+L3}(L1 + L2)^2)$.

### 5.3 Comparison of the Proposed and Reported Approaches for Algebraic Cryptanalysis

According to the results reported in [8], [1] and [9] it can be directly shown that the following proposition holds.

**Proposition 7.** Employment of an algebraic attack based on the results reported in [8], [1] and [9] for cryptanalysis of the nonlinear combination keystream generator [4] has the following main features:

**Table 2.** Numerical comparison of the requirements for cryptanalysis of the nonlinear combination keystream generator from [4] employing proposed algebraic attack and the algebraic attack based on the results reported in [8], [1] and [9] assuming in the second case that the algebraic degree is $d = 3$ instead the initial $d = 4$ due to the transformation $f'(\cdot) = (x_2 \oplus 1)f(\cdot)$.

| | pre-processing time complexity | processing time complexity | required sample | required memory |
|---|---|---|---|---|
| algebraic attack following reported in [8], [1], [9] | $\sim 2^{56}$ | $\sim 2^{36}$ | $\sim 2^{21}$ | $\sim 2^{36}$ |
| proposed Algorithm III, $B = 5$ | $\sim 2^{45}$ | $\sim 2^{33}$ | $\sim 2^{35}$ | $\sim 2^{40}$ |
| proposed Algorithm III, $B = 1$ | $\sim 2^{41}$ | $\sim 2^{29}$ | $\sim 2^{39}$ | $\sim 2^{36}$ |

• The expected required sample is $O(\binom{L1+L2+L3+L4+L5}{d})$;
• The expected time complexity of pre-processing is $O(\binom{L1+L2+L3+L4+L5}{d}^{\omega})$, and the expected space complexity is $O(\binom{L1+L2+L3+L4+L5}{d}(L1+L2+L3+L4+L5)^2)$;
• The expected time complexity of the processing is
$O(\binom{L1+L2+L3+L4+L5}{d}(L1+L2+L3+L4+L5)^2)$, and
the expected space complexity is
$O(\binom{L1+L2+L3+L4+L5}{d}(L1+L2+L3+L4+L5)^2)$;
where $\omega = 2.7$ and $d = 3$ due to the algebraic degree reduction via employment the function $f'(\cdot) = (x_2 \oplus 1)f(\cdot)$ instead $f(\cdot)$.

Regarding Proposition 7, particularly note that the fast preprocessing can not be employed because $L2 = 63$ and $L3 = 21$ are not co-prime, and this is in collision with a basic constraint regarding the fast pre-processing given in [1].

According to the Propositions 4 - 7, Table 2 yields an illustrative numerical summary of the attacking requirements of the reported and the proposed techniques for cryptanalysis.

Table 2 implies the breakability of the considered keystream generator employing the developed dedicated decimation based algebraic attack, and that the developed technique is, in the considered scenario, significantly superior than the best previously reported techniques. Comparing Table 1 and Table 2, we observe that the correlation based approach in this particular case is less efficient than the algebraic one. This is due to the fact that LFSR4 is shorter than LFSR1, which also results in a smaller value of $B$.

## References

1. F. Armknecht, "Improving fast algebraic attacks", FSE 2004, *Lecture Notes in Computer Science*, vol. 3017, pp. 65-82, 2004.
2. E. Barkan, E. Biham and N. Keller, "Instant ciphertext-only cryptanalysis of GSM encrypted communications", CRYPTO 2003, *Lecture Notes in Computer Science*, vol. 2729, pp. 600-616, 2003.

3. E. Biham and O. Dunkelman, "Cryptanalysis of the A5/1 GSM stream cipher", INDOCRYPT 2000, *Lecture Notes in Computer Science*, vol. 2977, pp. 43-51, 2000.
4. A. Braeken, V. Nikov, S. Nikova and B. Preneel, "On Boolean functions with generalized cryptographic properties", INDOCRYPT 2004, *Lecture Notes in Computer Science*, vol. 3348, pp. 120-135, 2004.
5. P. Chose, A. Joux and M. Mitton, "Fast correlation attacks: An algorithmic point of view", EUROCRYPT 2002, *Lecture Notes in Computer Science*, vol. 2332, pp. 209-221, 2002.
6. N.T. Courtois, "Higher order correlation attacks, XL algorithm and cryptanalysis of Toyocrypt", ICISC2002, *Lecture Notes in Computer Science*, vol. 2587, pp. 182-199, 2003.
7. N.T. Courtois and W. Meier, "Algebraic attacks on stream ciphers with linear feedback", EUROCRYPT2003, *Lecture Notes in Computer Science*, vol. 2656, pp. 345-359, 2003.
8. N.T. Courtois, "Fast algebraic attacks on stream ciphers with linear feedback", CRYPTO2003, *Lecture Notes in Computer Science* vol. 2729, pp. 176-194, 2003.
9. P. Hawkes and G. Rose, "Rewriting variables: the complexity of Fast algebraic attacks on stream ciphers", CRYPTO 2004, *Lecture Notes in Computer Science*, vol. 3152, pp. 390-406, 2004.
10. F. Jonsson, *Some Results on Fast Correlation Attacks.* Lund University, Lund, Sweden, Ph.D. Thesis, 141 pages, May 2002. ISBN: 91-7167-024-6.
11. A. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography.* Boca Roton: CRC Press, 1997.
12. W. Meier and O. Staffelbach, "Fast correlation attacks on certain stream ciphers," *Journal of Cryptology*, vol. 1, pp. 159-176, 1989.
13. W. Meier, E. Pasalic and C. Carlet, "Algebraic attacks and decomposition of Boolean functions", EUROCRYPT 2004, *Lecture Notes in Computer Science*, Vol. 3027, pp. 474-491, 2004.
14. M. J. Mihaljević, M. P. C. Fossorier and H. Imai, "On decoding techniques for cryptanalysis of certain encryption algorithms," *IEICE Trans. Fundamentals*, vol. E84-A, pp. 919-930, Apr. 2001.
15. M. J. Mihaljević and J. Dj. Golić, "A method for convergence analysis of iterative probabilistic decoding," *IEEE Trans. Inform. Theory*, vol. 46, pp. 2206-2211, Sept. 2000.
16. M. J. Mihaljević, M.P.C. Fossorier and H. Imai, "Fast correlation attack algorithm with list decoding and an application", FSE 2001, *Lecture Notes in Computer Science*, vol. 2355, pp. 196-210, 2002.
17. M. J. Mihaljević and H. Imai, "Cryptanalysis of TOYOCRYPT-HS1 stream cipher", *IEICE Transactions on Fundamentals*, vol. E85-A, pp. 66-73, Jan. 2002.
18. M. J. Mihaljević and R. Kohno, "Cryptanalysis of fast encryption algorithm for multimedia FEA-M", *IEEE Communications Letters*, vol. 6, pp. 382-384, Sept. 2002.
19. M. J. Mihaljević and H. Imai, "The decimated sample based improved algebraic attacks on the nonlinear filters", SCN 2004, *Lecture Notes in Computer Science*, vol. 3352, pp. 310-323, Jan. 2005.
20. H. Molland, J.E. Mathiassen and T. Helleseth, "Improved fast correlation attack using low rate codes", Cryptography and Coding 2003, *Lecture Notes in Computer Science*, vol. 2898, pp. 67-81, 2003.

# TMD-Tradeoff and State Entropy Loss Considerations of Streamcipher MICKEY

Jin Hong and Woo-Hwan Kim

National Security Research Institute,
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, Korea
{jinhong, whkim5}@etri.re.kr

**Abstract.** We give three weaknesses of a recently proposed streamcipher MICKEY. A small class of weak keys is found and we show time-memory-data tradeoff is applicable. We also show that the state update function reduces entropy of the internal state as it is iterated, resulting in keystreams that start out differently but become merged together towards the end.

**Keywords:** MICKEY, stream cipher, time memory data tradeoff, internal state entropy, weak key.

## 1 Introduction

MICKEY [4] is a streamcipher submitted to eSTREAM [2], targeting 80-bit security in low-end hardware environments. It was one of the algorithms selected and presented at the Symmetric Key Encryption Workshop (SKEW, Århus, Denmark, May, 2005).

Internal state of the cipher consists of two 80-bit feedback shift registers that are carefully designed to mutually and irregularly clock each other. The output keystream is a simple XOR of the two shift register outputs and hence not much more than the two registers themselves are needed for a hardware implementation of MICKEY. Since the minimum internal state size for a streamcipher aiming for 80-bit security is 160 bits, the cipher seems to be quite near the minimum point reachable with respect to hardware implementation cost and this makes it a very attractive proposal for constrained hardware environments.

In this paper, three undesirable properties of MICKEY, in relation to its security aspects, are discussed. The first is that time-memory-data tradeoff is applicable with online complexity lower than exhaustive search. It is widely believed that using an internal state of twice key size eliminates TMD-tradeoffs completely. We show that, with the BSW sampling technique, we can effectively reduce the search space and apply TMD-tradeoff to a reduced internal state.

Our second remark concerns the fact that the state update function is not bijective. This reduces entropy of the internal state as the generator is run forward and we study the accumulated entropy loss. We show that although the restriction to $2^{40}$-bits of keystream usage per key setup, set up by the designers,

© Springer-Verlag Berlin Heidelberg 2005

prevents any keystream from circling back through a whole period, it does not stop two different long keystreams from merging together towards the end.

The last observation we make is the existence of a small family of weak keys and the higher security cousin MICKEY-128 is considered in the appendix.

## 2 MICKEY

Let us very briefly recall the specifications of MICKEY and fix notation.

MICKEY uses two 80-bit registers named $\mathcal{R}$ and $\mathcal{S}$. Register $\mathcal{R}$ is a linear feedback shift register whose cells we denote by $r_i$ $(0 \leq i \leq 79)$. Depending on a *control bit*, it is clocked in two different ways. If the control bit is 0, it is clocked in the normal way. When the control bit is 1, it is clocked $2^{40} - 23$ times. A way to do this without repeatedly clocking the register is given, but we shall not be concerned with implementation aspects. Register $\mathcal{S}$ is a nonlinear feedback shift register whose cells we denote by $s_i$ $(0 \leq i \leq 79)$. As with $\mathcal{R}$, there are two ways to clock $\mathcal{S}$, the choice being made through another *control bit*.

Since $\mathcal{R}$ is an LFSR, it is clear that, once the control bit used for the most recent clocking is known, it may be inverted one step back. The inversion map will be linear for both cases of the control bit. Similarly, as the designers state in their specifications, the clocking for $\mathcal{S}$ is also invertible, once the control bit is fixed. The actual inverse clocking method is easy to find once the clocking method is fully understood.

One bit of keystream is produced *before* each internal state update by XOR-ing two bits contained in cells $r_0$ and $s_0$. To update the whole generator, the XOR $s_{27} \oplus r_{53}$ of two bits is set as control bit for register $\mathcal{R}$, the bit $s_{53} \oplus r_{26}$ is set as the control bit for register $\mathcal{S}$, and the two 80-bit registers are clocked accordingly. After each key-IV loading, which we do not describe, the keystream generator may be used to produce at most $2^{40}$ bits of keystream.

The general ideas of this paper should be understandable with what has so far been explained of MICKEY. But those interested in following actual computations should consult the original specification [4] for more information.

## 3 TMD-Tradeoff with BSW Sampling

In this section, we shall see that time-memory-data tradeoff is applicable to the internal state of MICKEY with online complexity less than exhaustive search.

### 3.1 BS-Tradeoff

The problem of recovering the internal state of a streamcipher, given multiple keystream segments, may be seen as a special case of the following more general *inversion* problem.

Given a one-way function $f : \mathcal{X} \to \mathcal{Y}$ and a set $\mathcal{D} = \{y_i\}_i \subset \mathcal{Y}$, find at least one $x_i \in X$ for which $f(x_i) = y_i$.

Our interest lies in the case where $\mathcal{X}$ is the set of all internal states, $\mathcal{Y}$ is the set of keystream segments long enough to reliably distinguish states, and $f$ is the mapping of state to finite keystream segment. Once the internal state corresponding to some keystream segment is obtained by inverting this mapping, the cipher can be run forward to obtain all future keystream for that session.

Starting with the work of Hellman [9], numerous studies on time-memory-(data) tradeoff attacks have appeared in the literature, but we shall focus on the TMD-tradeoff attack given by Biryukov and Shamir [5]. The BS-tradeoff, interpreted as a tool for inverting one-way functions $f : \mathcal{X} \to \mathcal{Y}$, is as follows.

Let $N = |\mathcal{X}|$ be the size of the search space. Given any triple $(T, M, D)$ satisfying the *tradeoff curve*

$$TM^2D^2 = N^2 \quad \text{with} \quad 1 \le D^2 \le T, \tag{1}$$

there exists an algorithm that solves the inversion problem in the following sense.

Before a target set $\mathcal{D}$ of size $D$ is given, the attacker prepares tables to be stored in memory of size $M$ through a pre-computation phase requiring (offline) time $P = N/D$. When the target data set $\mathcal{D}$ is given, with high probability, a single solution to the inversion problem is produced within (online) time $T$.

A typical point on the BS-tradeoff curve is

$$T = M = N^{\frac{1}{2}}, \quad D = N^{\frac{1}{4}}, \quad \text{with} \quad P = N^{\frac{3}{4}}. \tag{2}$$

Since the attack complexity of this approach is taken to be the maximum of $T$, $M$, and $D$, disregarding the offline time complexity, together with birthday paradox based TMD-tradeoff attacks [3, 8], this has worked as a reason for many recent streamcipher designs incorporating internal state size at least twice as big as intended security level. MICKEY follows this trend and uses 160-bit state size to aim for 80-bit security.

## 3.2 Restricting the Search Space

Let us explain the basics of BSW-sampling [6]. Suppose we can find a subset $\mathcal{X}' \subset \mathcal{X}$ for which elements of $\mathcal{Y}' = f(\mathcal{X}')$ can easily be singled out from $\mathcal{Y}$. Consider the restriction $f' : \mathcal{X}' \to \mathcal{Y}'$ of $f$ to $\mathcal{X}'$. Given a target data set $\mathcal{D} \subset \mathcal{Y}$, one *samples* the data by taking $\mathcal{D}' = \mathcal{D} \cap \mathcal{Y}'$ and apply TMD-tradeoff to $f'$ with the smaller data set $\mathcal{D}'$. As the search space $\mathcal{X}'$ is smaller than $\mathcal{X}$, this should be more efficient than applying TMD-tradeoff to $f$ itself, and it is clear that any inversion of $f'$ is also an inversion for $f$. Of course, the (pre-sampling) data requirement will be larger than the usual approach.

## 3.3 Sampling MICKEY Keystream

The usual way of applying TMD-tradeoff attack to MICKEY is to consider the case where $\mathcal{X}$ is the set of all 160-bit internal states and $\mathcal{Y}$ is the set of all keystream segments of 160-bit length. The mapping $f$ will send an internal state to the first 160 keystream bits obtained from the state.

To apply the sampling method, we define $\mathcal{Y}'$ to be the set of all 160-bit keystream segments which starts with 27 zeros. These are certainly easily distinguishable from the rest of the elements of $\mathcal{Y}$. With an understanding of BS-tradeoff algorithm, it should be clear that the only obstacle to its actually application on the restricted mapping $f' : \mathcal{X}' = f^{-1}(\mathcal{Y}') \to \mathcal{Y}'$ is the construction of an arbitrary, efficiently computable, preferably injective, mapping[1]

$$h : \mathcal{Y}' \to \mathcal{X}'. \tag{3}$$

The construction of this mapping for MICKEY is a bit tedious, so let us go through this step by step.

1. View a random element $\mathbf{y} \in \mathcal{Y}'$ as a 133-bit value by disregarding the 27 initiating zeros.
2. Fill register $\mathcal{S}$ with the first 80 bits of $\mathbf{y}$.
3. Fill cells $r_1, \ldots, r_{53}$ of register $\mathcal{R}$ with the remaining 53 bits of $\mathbf{y}$. We shall consider the remaining 27 cells, $r_0$ and $r_{54}, \ldots, r_{79}$ as having been filled with indeterminate variables $x_0$ and $x_{54}, \ldots, x_{79}$. To define the mapping (3), it suffices to determine the correct values for these variables which allow the keystream generated from this state to start with 27 zeros.
4. Since the first output bit $r_0 \oplus s_0$ must be zero, and since $s_0$ is already fixed, we immediately obtain $x_0$. Hence forth, we shall treat $x_0$ as a constant rather than as a variable.
5. Calculate $s_{53} \oplus r_{26}$, the control bit for $\mathcal{S}$, and also $s_{27} \oplus r_{53}$, the control bit for $\mathcal{R}$.
6. Clock register $\mathcal{S}$ according to its control bit.
7. Clock register $\mathcal{R}$ according to its control bit. In saying this, we mean to write the contents of the cells as linear functions of the remaining variables $x_{54}, \ldots, x_{79}$. In particular, using the feedback equation for $\mathcal{R}$, one can show that the new content of $r_0$ will be
   - $x_{79}$, if the control bit for $\mathcal{R}$ is zero, and
   - $x_{79} \oplus x_0$, if the control bit for $\mathcal{R}$ is one.

   Recalling the fact that $x_0$ has already been determined, and using the condition for the second output bit to be zero, one determines $x_{79}$ which is actually the feedback bit. We can now treat $x_{79}$ also as a constant, rather than as a variable.
8. The update method of $\mathcal{R}$ is so that determination of the feedback bit allows cells $r_0, \ldots, r_{53}$ to be determined explicitly, and for the rest of the cells to be written as a function of the remaining 25 variables $x_{54}, \ldots, x_{78}$. For example, the current updated value of $r_{79}$ will be either $x_{78} \oplus x_{79}$ or $x_{78}$, depending on the (known) control bit for $\mathcal{R}$.
9. Once again, calculate both control bits and clock the two registers accordingly.
10. This time, the new value for $s_0$ and the zero-output condition will determine the variable $x_{78}$ and also the second feedback bit.

[1] This mapping need not be related to the inverse of $f$ in any way.

11. Continue in this manner until all variables are determined. Each added clocking determines one more indeterminate and hence at most 26 clockings are needed.

We have thus defined an appropriate mapping (3) and successfully reduced the search space size from $2^{160}$ to $N = 2^{133}$.

### 3.4 TMD-Tradeoff with Sampling on MICKEY

Let us now see what TMD-tradeoff complexities we can obtain on the smaller space $\mathcal{X}'$.

Start with unfiltered data size of $2^{60}$. For example, these may be obtained by sliding a window of 160-bit size over $2^{20}$-many keystreams, each of $(2^{40} + 159)$-bit length. Only about $2^{-27}$ of these will start with 27 zeros. Hence the sampled data to be used in our TMD-tradeoff attack is $D = 2^{33} = 2^{60-27}$. Values $T = 2^{66}$ and $M = 2^{67}$, together with $D = 2^{33}$ and $N = 2^{133}$, satisfy the TMD-tradeoff curve (1), and the offline pre-computation complexity becomes $P = 2^{100}$.

In summary, once a table costing offline time $2^{100}$ is built, TMD-tradeoff attack with BSW sampling on MICKEY is possible with online complexity $2^{67}$. This is (almost) the minimum online complexity achievable with our sampling.

If lowering the pre-computation time is more desirable, we start with unfiltered data size of $2^{66.5}$. This results in the tradeoff point $T = 2^{79}$, $M = 2^{54}$, and $D = 2^{39.5}$, with pre-computation time $P = 2^{93.5}$.

Owing to the pre-computation complexity larger than exhaustive search of key, some would not view this technically as a *break* of MICKEY. But still, it does show that we cannot treat MICKEY as providing absolutely full 80-bit security. Furthermore, the fact that such sampling is possible can be viewed as a weakness in itself. This might open doors to other exploits.

Notice that ECRYPT's current recommendation for key sizes [1] seem to be viewing anything less than 81 bits to be vulnerable to exhaustive search by large agencies, hence the pre-computation time $2^{100}$ or $2^{93.5}$ cited above should be seen as reachable in the near future.

We refer readers to Appendix A for a related general treatment of complexities involving TMD-tradeoff with BSW sampling.

## 4 State Entropy Loss

The state update function of MICKEY starts by calculating two control bits. These bits determine the fate of the very bits used to obtain the control bits. This kind of double-use of information usually produces collisions. The state update function is not one-to-one and through repeated application of the state update function, entropy of the state is bound to decrease. In this section, through heuristic arguments, we study the actual amount of entropy lost.[2]

[2] Readers should not confuse the state update function with the state-to-keystream function of the previous section. Also, neither is related to the initialization process.

## 4.1 Preliminary

Given a set $X = \{x_i\}_{i \in I}$, with each element $x_i$ appearing with probability $p_i$, the *entropy* of set $X$ is defined as

$$H(X) = -\sum_{i \in I} p_i \log_2 p_i. \tag{4}$$

For example, if some set $Y$ contains $N$ elements and all elements occur with equal probability, then it is easy to show that $H(Y) = \log_2 N$. Hence, a set of size $2^n$ with uniform probability distribution has entropy $n$.

Given a mapping $\varphi$, acting on a space of size $N$ and of uniform distribution, let us define the notation

$$\mathrm{EL}(\varphi) = \log_2 N - H(\mathrm{Image}(\varphi)), \tag{5}$$

$$\overline{\mathrm{EL}}(\varphi) = \log_2 N - \log_2 |\mathrm{Image}(\varphi)|. \tag{6}$$

The operator EL measures the *entropy loss* suffered by a uniform space through application of a mapping. Operator $\overline{\mathrm{EL}}$ measures the same value in a rough way, by assuming that all elements of the image space occur with equal probability. We will always have $\mathrm{EL}(\varphi) \geq \overline{\mathrm{EL}}(\varphi)$ for any map $\varphi$.

## 4.2 Initial Comparison of Update Function and Random Mapping

There following behavior of random mappings is well known [7, 10].

**Lemma 1.** *The expected number of image points under a random mapping on $N$ elements, has the asymptotic form $(1 - \frac{1}{e})N$, as $N \to \infty$.*

Using our notation, this may be written equivalently in the following way.

**Lemma 2.** *For random mappings $\varphi_N$ on $N$ elements, the expected value of $\overline{\mathrm{EL}}(\varphi_N)$ approaches $-\log_2(1 - \frac{1}{e}) \sim 0.6617$ as $N \to \infty$.*

So, if a random function were chosen as the state update function for some streamcipher, the internal state would lose more than 0.66-bit entropy on its first update.

Let us see how the MICKEY's state update function behaves in this respect. For the rest of this section, $\varphi$ will denote a *random mapping* and $f$ will denote the state update function of MICKEY. Consider the following procedure.

1. Choose random states for both registers $\mathcal{R}$ and $\mathcal{S}$.
2. Calculate the reverse clocking of register $\mathcal{R}$, assuming control bit set to 0, and call the new state $\mathcal{R}_0$. Similarly, previous state of $\mathcal{R}$ assuming control bit 1 is calculated and named $\mathcal{R}_1$. The same is done for register $\mathcal{S}$ with results named $\mathcal{S}_0$ and $\mathcal{S}_1$.
3. For each of the four possible $(\mathcal{R}_i, \mathcal{S}_j)$ pairs, calculate the two control bits $(s_{27} \oplus r_{53}, s_{53} \oplus r_{26})$ and check to see if these match the control bits $(i, j)$ that was used. Count the number of matches.

This procedure counts the number of internal states that map to the chosen random state under the state update function $f$. Note that a state may have anywhere from zero up to four pre-image states. We repeated[3] this process for a total of $2^{20}$ times, while tallying the occurrence of random states having each number of pre-images. The result is recorded in Table 1.

**Table 1.** Distribution of $2^{20}$ randomly chosen states according to pre-image counts

| inverse count | 0 | 1 | 2 | 3 | 4 | total |
|---|---|---|---|---|---|---|
| randomly chosen states | 307988 | 452017 | 279418 | 0 | 9153 | $2^{20}$ |

The table shows, in particular, that of the $2^{20}$ states we had chosen at random, 307988-many of them corresponded to no pre-image, where as 740588-many were image points under at least one pre-image. So about 70.63% of the states we had tried were image points. Assuming that this proportion extends to the set of all states, we can calculate $\overline{\mathrm{EL}}(f) \sim -\log_2(0.7063) \sim 0.5017$. Comparing this with the corresponding value 0.6617 for a random function stated by Lemma 2, we can expect the behavior of $f$ with respect to entropy loss to be somewhere between a permutation and a random function.

Let us next work with $f \circ f$, i.e., $f$ iterated twice, in seeing the distribution of states according to inverse image counts. For every pre-image of a random state under $f$, we checked if this pre-image again had a pre-image under $f$. Result of twice iterated inverse image counting is given in Table 2.

**Table 2.** Distribution of $2^{20}$ randomly chosen states according to pre-image counts under mapping $(f \circ f)$

| inverse count | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 ∼ 16 | total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| randomly chosen states | $2^{18.81}$ | $2^{17.95}$ | $2^{17.92}$ | $2^{15.82}$ | $2^{14.65}$ | $2^{10.35}$ | $2^{11.20}$ | $2^{7.93}$ | $2^{7.48}$ | $2^{5.55}$ | $2^{3.32}$ | 0 | $2^{4.00}$ | 0 | $2^{20}$ |

The total number of twice iterated images is 588990 and accounts for about 56.17% of all $2^{20}$ states tried. As before, we can calculate $\overline{\mathrm{EL}}(f \circ f) \sim 0.8321$. Hence, from the view point of entropy preservation, $f \circ f$ is worse than a random mappings.

One of our goals for this section is to see how much internal state entropy is lost through $2^{40}$ iterated applications of state update function $f$. Notice that $2^{39}$ applications of $f \circ f$ is equal to $2^{40}$ applications of the single $f$. So even though $f$ is better than a random function in relation to entropy loss, as we have seen evidence that $f \circ f$ is worse than a random function, working with random mapping $\varphi$ as an approximation of $f$ could be a possibility. We shall see more justification for this in the next subsection.

[3] Random states were chosen with the C-language `rand()` function which is known to be not so random, but as it is unrelated to $f$, this should not affect our results.

## 4.3 Iterations of Update Function and Random Mapping

Let us calculate the entropy of Image($f$). Going back to Table 1, it appears that about

$$\frac{307988}{2^{20}} \cdot 2^{160} \sim 2^{158.23}$$

internal states never appear as image point under $f$. Probability of any one of these $2^{158.23}$-many states to be equal to a randomly produced image point under $f$ is 0.

Similarly, $2^{158.79} \sim \frac{452017}{2^{20}} \cdot 2^{160}$ internal states have exactly one pre-image under $f$. If we fix any one of these and are given a randomly produced image point under $f$, then the two are equal with probability roughly $1/2^{160}$. There are $2^{158.09} \sim \frac{279418}{2^{20}} \cdot 2^{160}$ internal states with two $f$ pre-images. If we fix any one of these, the probability of it coinciding with a randomly produced $f$-image state is $2/2^{160}$. Finally, for $2^{153.16} \sim \frac{9153}{2^{20}} \cdot 2^{160}$ states, their probability of coinciding with a randomly produced $f$-image is $4/2^{160}$.

The sum of all probabilities

$$0 \cdot 2^{158.23} + \frac{1}{2^{160}} \cdot 2^{158.79} + \frac{2}{2^{160}} \cdot 2^{158.09} + \frac{3}{2^{160}} \cdot 0 + \frac{4}{2^{160}} \cdot 2^{153.16} \sim 0.99894$$

is not exactly 1, but reasonably close, so we shall ignore this. With samples bigger than $2^{20}$, the sum would be closer to 1.

Placing all information into the definition (4) for entropy gives us

$$\begin{aligned} H(\text{Image}(f)) &= -\Big\{2^{158.79}\frac{1}{2^{160}}\log_2(\frac{1}{2^{160}}) + 2^{158.09}\frac{2}{2^{160}}\log_2(\frac{2}{2^{160}}) \\ &\qquad + 0\frac{3}{2^{160}}\log_2(\frac{3}{2^{160}}) + 2^{153.16}\frac{4}{2^{160}}\log_2(\frac{4}{2^{160}})\Big\} \\ &\sim 160 - 0.6028. \end{aligned}$$

We have calculated the entropy loss $\text{EL}(f) = 0.6028$ of state update function $f$. As expected, this is greater than the rough measure $\overline{\text{EL}}(f) = 0.5017$, which we obtained earlier in Section 4.2. This process can be repeated for $f \circ f$ to obtain $\text{EL}(f \circ f) = 0.9913$.

Let us denote $k$-many compositions of $f$ as $f^k$. Through the method just explained, we explicitly obtained $\text{EL}(f^k)$ values for many values of $k$. The results are given as $\circ$ -points in Figure 1. The $x$-axis gives the logarithm of $k$-values and the $y$-axis gives the entropy loss in bits. For small values of $k$, the entropy loss values we obtained seemed quite accurate, but for larger $k$, multiple tests resulted in slightly varying entropy loss values. So we used larger sample sizes (some as large as $2^{24}$) for larger $k$ to keep the variance to an acceptable degree as can be witnessed from the graph.

The graph also contains three *rough* entropy measures $\overline{\text{EL}}(f^k)$, $\overline{\text{EL}}(\varphi^k)$, and $\overline{\text{EL}}((f \circ f)^k)$. The graph for $\overline{\text{EL}}(f^k)$ comes from the same test data that gave $\text{EL}(f^k)$. Multiple tests indicated that our $\overline{\text{EL}}(f)$ data points were accurate enough for multiple points to be almost undistinguishable on the graph. The curve for $\overline{\text{EL}}((f \circ f)^k)$ contains points that do not correspond to integer values

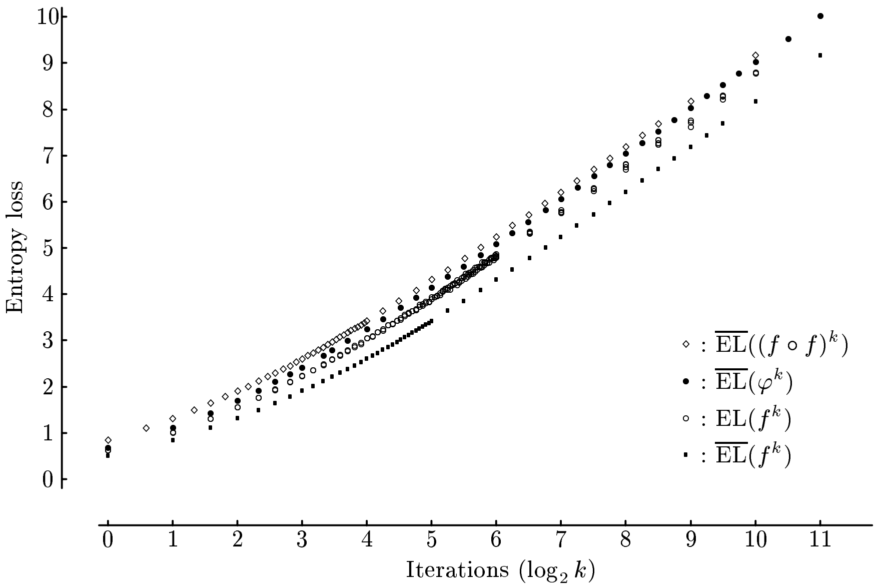


**Fig. 1.** Entropy loss under iteration

of $k$ and hence may seem strange, but this is because we had just taken the unit left-shift of $\overline{\mathrm{EL}}(f^k)$ in its place. Finally, the graph for $\overline{\mathrm{EL}}(\varphi^k)$ relies on the following well-known lemma [7, 10].

**Lemma 3.** *For random mappings on $N$ elements, the expected number of $k$-th iterate image points has the asymptotic form $(1 - \tau_k)N$, as $N \to \infty$. Here, $\tau_k$ is given by the recursion formula, $\tau_0 = 0$, $\tau_{k+1} = \exp(\tau_k - 1)$.*

For any fixed $N$, since repeated iterations of $\varphi$ eventually results in a cycle of expected length $O(\sqrt{N})$, the number of $k$-th iterate image points must stop decreasing at some point as $k$ nears $N$. But in our situation, even with $N = 2^{160}$ fixed, the values of $1 \le k \le 2^{40}$ we are interested in are much smaller than $\sqrt{N}$, so using $(1 - \tau_k)N$ as the approximate number of image points should not be too unreasonable. Our •-points are actually nothing more than calculations of $-\log_2(1 - \tau_k)$, which is the value for $\overline{\mathrm{EL}}(\varphi)$ under the assume that the number of $k$-th iterate image points is $(1 - \tau_k)N$.

As was argued in Section 4.2, we can confirm that the curve $\overline{\mathrm{EL}}(\varphi^k)$, which gives a rough measure of entropy loss for random mappings, sits comfortable in between respective curves for $f$ and $f \circ f$. We can also see that the real entropy loss $\mathrm{EL}(f^k)$ for state update function follows the rough entropy loss $\overline{\mathrm{EL}}(\varphi^k)$ of random mapping very closely.

We can expect these trends to continue for higher values of $k$, as long as they are very small relative to $\sqrt{N}$. As this is true for $k$ values we are interested in, we shall use $-\log_2(1 - \tau_k)$ to approximate $\mathrm{EL}(f^k)$ in the next subsection. In any case, there is less than 1-bit difference between any of the four graphs, and the exact value will not be critical in our discussions.

**Table 3.** Rough entropy loss estimate for iterated random mapping

| $k$ | 1 | 2 | $2^2$ | $2^3$ | $2^4$ | $2^5$ | $2^6$ | $2^7$ | $2^8$ | $2^9$ | $2^{10}$ | $2^{11}$ | $2^{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $-\log_2(1-\tau_k)$ | 0.66 | 1.09 | 1.68 | 2.40 | 3.23 | 4.13 | 5.07 | 6.04 | 7.02 | 8.011 | 9.006 | 10.003 | 11.0016 |

The following conjecture gives some explicit numbers to approximate $\mathrm{EL}(f^k)$.

*Conjecture 1.* We have

$$-\log_2(1-\tau_k) \to \log_2(k) - 1$$

as $k \to \infty$.

We have no proof for this statement, but experimental results given in Table 3 supports this very strongly.

Let us summarize what we have discussed so far.

1. State update function $f$ behaves close to a random function with respect to entropy preservation properties, although some difference can be seen.
2. We saw evidence that the entropy loss $\mathrm{EL}(f^k)$ may be approximated by $\overline{\mathrm{EL}}(\varphi^k) \sim -\log_2(1-\tau_k) \sim \log_2(k) - 1$ for those values of $k$ much smaller than $\sqrt{N}$, but not too small.
3. Based on the previous statement, we can expect 39-bit entropy loss of internal state after $2^{40}$ iterations of state update function $f$.

We do not claim that the 39-bit entropy loss is an exact value. But we have provided heuristic argument that it can be seen as a rough approximation.

### 4.4 Security Implications

We have, so far, confirmed that the state update function of MICKEY is not a permutation of the internal state but rather behaves like a random function. This, in itself, is not a desirable property for a state update function.

Turning to more specific numbers, first recall that the designers restricted keystream usage to $2^{40}$ bits for each key-IV setup. From discussions of the previous subsection, we can expect entropy loss of internal state to be equivalent to 39 bits after $2^{40}$ iterations of the state update function. As the internal state started out with 160-bit entropy, at the end of the keystream of length $2^{40}$, state entropy would be equivalent to about 121 random bits.

Now, suppose we collect long keystreams, each of length $2^{40} - \varepsilon$, where $\varepsilon$ is small relative to $2^{40}$, for example, $\varepsilon = 2^{20}$. We pair these keystreams with their respective final internal states. If we collect $2^{60.5}$-many of these, due to the birthday paradox, we are highly likely to find two keystreams with identical final internal states. Once the states are equal, keystreams of length $\varepsilon$ produced from that point on will also be equal.

In short, if we collect large number of long MICKEY keystreams, we are bound to find two keystreams that started out differently, but ends the same, irrespective of whether the collected keystreams correspond to one or multiple keys. This may not qualify as an attack, but does show that using multiple long keystreams from MICKEY can be dangerous.

The large number $2^{60.5}$ may give a wrong impression about the related threat level. Notice that any $(k+1)$-th iterate image state is at the same time a $k$-th iterate image state. This observation implies that multiple shifts of a single keystream all qualify as (slightly shorter) long keystreams appearing in the above argument.

Let us try some more explicit numbers. One keystream of length $2^{40}$ can be shifted multiple times to produce $(2^{39} - \varepsilon)$-many keystreams, each of length (at least) $2^{39} + \varepsilon$. A gathering of $2^{22}$-many $2^{40}$-length keystreams can thus also be interpreted as about $2^{61} \sim 2^{22} \cdot (2^{39} - \varepsilon)$ keystreams of length $2^{39} + \varepsilon$. Since entropy of state after $2^{39}$ iterated state updates is expected to be about 122, one could say that we are likely to obtain a matching state.

In reality, more than $2^{22}$ base keystreams of length $2^{40}$ will need to be gathered before a match is found. This is because no matching will occur within the set of shifted sequences originating from the same base sequence. The exact analysis of these numbers seems to be complicated, but it is clear that the threat caused by state entropy loss cannot be dismissed lightly.

So far, we have stated the entropy loss as an undesirable property, rather than as an attack on MICKEY. But if one wants to consider the related data complexity, it should be noticed that the whole keystream need not be stored. Only the ending parts are used in looking for a match. The actual amount of data that needs processing is $C \cdot 2^{60.5} \sim 2^{68}$, where constant $C$ is the number of keystream bits needed to reliably distinguish between different states. It is not totally clear as to which number should be taken as data complexity.

To develop a concrete attack scenario, it could be worthwhile to think about the case where an appropriate keystream set is pre-generated and compared with target online data.

*Remark 1.* Let us add a few words of explanation on the use of EL versus $\overline{\text{EL}}$. Some might believe that $\overline{\text{EL}}$, which corresponds to iterated image point counts, should be considered in these arguments. The difference between the two measures is whether the uneven distribution within the image set was taken into account. Given two sets of the same size, a collision is more likely to appear in the one with a more uneven probability distribution. Hence, the distribution within the set does matter in these arguments, with the conclusion that $\text{EL}(f)$, rather than $\overline{\text{EL}}(f)$, is the correct measure to look at.

*Remark 2.* This subsection has shown that any streamcipher with a random, as opposed to bijective, state update function should take care to use a state size larger than twice security level.

## 5 A Small Class of Weak Keys

Let us first quote from the MICKEY specification [4].

> There is a small class of arguably weak keys for MICKEY: namely, those $(K, IV)$, pairs for which the state of R after loading is all zeroes. ...,

> if an attacker assumes that this is the case, she can readily confirm her assumption and deduce the remainder of the generator state by analyzing a short sequence of keystream. But, because this can be assumed to occur with probability roughly $2^{-80}$ we do not think it necessary to prevent it (and so in the interests of efficiency we do not do so).

Notice that the all-zero state of register $\mathcal{R}$ is fixed under either value of its control bit. Given the full specification for $\mathcal{S}$, one can easily check that the following is an $\mathcal{S}$-state which is fixed under either value of its control bit. The state is given in hexadecimal notation with the left end corresponding to $s_0$.

```
10f0 02a1 000b 853f fff8
```

This fixed point was found by guessing just two cell values $s_{78}$ and $s_{79}$. The rest follows sequentially from the fixed point requirement.[4]

If register $\mathcal{S}$ is ever set to the above fixed $\mathcal{S}$-state, since the content of cell $s_0$ is 0, the keystream produced will be equal to whatever register $\mathcal{R}$, with its irregular clocking, produces. Those (key, IV) pairs, which sets register $\mathcal{S}$ to the above state after loading, forms a weak key class in the sense given by the designers. The probability of encountering one of these at random is roughly $2^{-80}$.

We now have two weak key classes, and we can expect the probability of encountering one of these to be roughly $2^{-79}$, which is strictly greater than $2^{-80}$. Deciding not to prevent either of these in view of efficiency will be harder to justify.

## 6 Conclusion

We have studied security aspects of the eSTREAM proposal MICKEY. Three undesirable properties have been discovered.

1. Time-memory-data tradeoff is applicable with online complexity smaller than exhaustive key search. Pre-computation needed is more demanding than exhaustive search but this could be reachable in the near future.
2. The internal state loses entropy with repeated applications of the state update function, resulting in keystreams that start out differently but merge together towards the end.
3. There exists a small family of weak keys.

The first of these may be removed through internal state size increase, or with a more complicated output filter. But neither method would be desirable in view of efficiency. The second weakness seems more fundamental to the design of MICKEY and does not seem to be easy to fix. The third can be avoided easily with small extra cost.

It can be said that none of these weaknesses are absolutely critical, but these must still be taken into account by anyone intending to use MICKEY.

[4] It turns out that the above $\mathcal{S}$-state is the unique state fixed under both control bit values.

## References

1. ECRYPT, ECRYPT yearly report on algorithms and keysizes (2004). Version 1.1, March, 2005. Available from `http://www.ecrypt.eu.org`.
2. ECRYPT, eSTREAM - the ECRYPT Stream Cipher Project. Information available from `http://www.ecrypt.eu.org/stream/`
3. S. H. Babbage, Improved exhaustive search attacks on stream ciphers. *European Convention on Security and Detection,* IEE Conference publication No. 408, pp. 161–166, IEE, 1995.
4. S. Babbage and M. Dodd, The stream cipher MICKEY (version 1). ECRYPT Stream Cipher Project Report 2005/015, 2005. Available from [2].
5. A. Biryukov and A. Shamir, Cryptanalytic time/memory/data tradeoffs for stream ciphers. *Asiacrypt 2000,* LNCS 1976, pp. 1–13, Springer-Verlag, 2000.
6. A. Biryukov, A. Shamir, and D. Wagner, Real time cryptanalysis of A5/1 on a PC. *FSE 2000*, LNCS 1978, pp. 1–18, Springer-Verlag, 2001.
7. P. Flajolet and A. M. Odlyzko, Random mapping statistics. *Eurocrypt '89*, LNCS 434, pp. 329–354, Springer-Verlag, 1990.
8. J. Dj. Golić, Cryptanalysis of alleged A5 stream cipher. *Eurocrypt'97,* LNCS 1233, pp. 239–255, Springer-Verlag, 1997.
9. M. E. Hellman, A cryptanalytic time-memory trade-off. *IEEE Trans. on Infor. Theory,* vol 26, pp. 401–406, 1980.
10. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

## A General Argument for TMD-Tradeoffs with BSW Sampling

Let us work with BS-tradeoff and BSW sampling in a more general setting and find the tradeoff point of minimal online complexity. Notation of Section 3 will be used.

Set $N' = |\mathcal{X}'| = |\mathcal{Y}'| = R{\cdot}N$ so that given $D$ random target points, $D' = R{\cdot}D$ of them will belong to $\mathcal{Y}'$. The tradeoff curve will be $TM^2D'^2 = N'^2$, $P = N'/D'$, with $1 \le D'^2 \le T$, or equivalently, $TM^2D^2 = N^2$, $P = N/D$, $D' = RD$ with $1 \le D'^2 \le T$.

A typical point on this curve is

$$T = M = D = N^{\frac{2}{5}}, \quad D' = R = N^{\frac{1}{5}}, \quad \text{and} \quad P = N^{\frac{3}{5}}. \tag{7}$$

Compared with (2), we have achieved lowered time and memory complexity at the expense of higher data complexity. In all, the overall online complexity has dropped from $N^{\frac{1}{2}}$ to $N^{\frac{2}{5}}$. Notice that pre-computation time has decreased also.

We add a word of caution, as the complexities $T$ and $P$ of (2) are counted in terms of number of applications of $f$, where as those of (7) refer to applications of $f'$, composed with $h$ appearing in (3). So if the efficiency of function $h$ is not comparable to that of $f$, the two time complexities cannot be compared in their raw form. Of course, similar remarks apply to $M$, with (7) at an advantage this time, but its importance is smaller.

## B MICKEY-128

It is quite clear that arguments we've given for MICKEY should be applicable in much the same way to its 128-bit security cousin MICKEY-128. We briefly write down the respective results concerning TMD-tradeoff and weak keys. Due to lack of time and computational power, we have not been able to study the entropy loss of MICKEY-128 state under its update function.

### B.1 TMD-Tradeoff with BSW Sampling

TMD-tradeoff is applicable to MICKEY-128 in the same way as with MICKEY. Through BSW sampling, the search space size is reduced from $2^{256}$ to $2^{213}$. Keystream segments starting with 43-many zero bits are the data we should be looking for.

A typical tradeoff point would be $T = 2^{106}$, $M = 2^{107}$, $D = 2^{53}$, with $P = 2^{160}$. The $D$ refers to filtered data, and the total data needed before filtering step is $2^{96}$.

The tradeoff point giving minimal pre-computation time is $T = 2^{127}$, $M = 2^{86}$, $D = 2^{63.5}$, with $P = 2^{149.5}$. The pre-filter data requirement is $2^{106.5}$.

### B.2 A Small Class of Weak Keys

There exists one fixed $\mathcal{S}$-state which remains fixed under either value of the control bit. It is given below with the left end corresponding to $s_0$.

```
05ff d071 d020 c3fc 0c1c 037f 1f3f ef98
```

This $\mathcal{S}$-state gives a small family of weak keys which may be encountered at random with probability $2^{-128}$.

# Time-Memory Trade-Offs: False Alarm Detection Using Checkpoints

Gildas Avoine[1], Pascal Junod[2], and Philippe Oechslin[1,3]

[1] EPFL, Lausanne, Switzerland
[2] Nagravision SA (Kudelski Group), Cheseaux, Switzerland
[3] Objectif Sécurité, Gland, Switzerland

**Abstract.** Since the original publication of Martin Hellman's cryptanalytic time-memory trade-off, a few improvements on the method have been suggested. In all these variants, the cryptanalysis time decreases with the square of the available memory. However, a large amount of work is wasted during the cryptanalysis process due to so-called "false alarms". In this paper we present a method of detection of false alarms which significantly reduces the cryptanalysis time while using a minute amount of memory. Our method, based on "checkpoints", reduces the time by much more than the square of the additional memory used, e.g., an increase of 0.89% of memory yields a 10.99% increase in performance. Beyond this practical improvement, checkpoints constitute a novel approach which has not yet been exploited and may lead to other interesting results. In this paper, we also present theoretical analysis of time-memory trade-offs, and give a complete characterization of the variant based on rainbow tables.

**Keywords:** Time-memory trade-off, cryptanalysis, precomputation.

## 1 Introduction

Many cryptanalytic problems can be solved in theory using an exhaustive search in the key space, but are still hard to solve in practice because each new instance of the problem requires to restart the process from scratch. The basic idea of a time-memory trade-off is to carry out an exhaustive search once for all such that following instances of the problem become easier to solve. Thus, if there are $N$ possible solutions to a given problem, a time-memory trade-off can solve it with $T$ units of time and $M$ units of memory. In the methods we are looking at $T$ is proportional to $N^2/M^2$ and a typical setting is $T = M = N^{2/3}$.

The cryptanalytic time-memory trade-off has been introduced in 1980 by Hellman [8] and applied to DES. Given a plaintext $P$ and a ciphertext $C$, the problem consists in recovering the key $K$ such that $C = \mathsf{S}_K(P)$ where $\mathsf{S}$ is an encryption function assumed to follow the behavior of a random function. Encrypting $P$ under all possible keys and storing each corresponding ciphertext allows for immediate cryptanalysis but needs $N$ elements of memory. The idea of a trade-off is to use chains of keys. It is achieved thanks to a reduction function $\mathsf{R}$ which generates a key from a ciphertext. Using $\mathsf{S}$ and $\mathsf{R}$, chains of alternating

© Springer-Verlag Berlin Heidelberg 2005

ciphertexts and keys can thus be generated. The key point is that only the first and the last element of each chain are stored. In order to retrieve $K$, a chain is generated from $C$. If at some point it yields a stored end of chain, then the entire chain is regenerated from its starting point. However, finding a matching end of chain does not necessarily imply that the key will be found in the regenerated chain. There exist situations where the chain that has been generated from $C$ merges with a chain that is stored in the memory which does not contains $K$. This situation is called a *false alarm*. Matsumoto, with Kusuda [10] in 1996 and with Kim [9] in 1999, gave a more precise analysis of the parameters of the trade-off. In 1991, Fiat and Naor [6, 7] showed that there exist cryptographically sound one-way functions that cannot be inverted with such a trade-off.

Since the original work of Hellman, several improvements have been proposed. In 1982, Rivest [5] suggested an optimization based on *distinguished points* (DP) which greatly reduces the amount of look-up operations which are needed to detect a matching end point in the table. Distinguished points are keys (or ciphertexts) that satisfy a given criterion, e.g., the last $n$ bits are all zero. In this variant, chains are not generated with a given length but they stop at the first occurrence of a distinguished point. This greatly simplifies the cryptanalysis. Indeed, instead of looking up in the table each time a key is generated on the chain from $C$, keys are generated until a distinguished point is found and only then a look-up is carried out in the table. If the average length of the chains is $t$, this optimization reduces the amount of look-ups by a factor $t$. Because merging chains significantly degrades the efficiency of the trade-off, Borst, Preneel, and Vandewalle [4] suggested in 1998 to clean the tables by discarding the merging and cycling chains. This new kind of tables, called *perfect table*, substantially decreases the required memory. Later, Standaert, Rouvroy, Quisquater, and Legat [14] dealt with a more realistic analysis of distinguished points and also proposed an FPGA implementation applied to DES with 40-bit keys. Distinguished points can also be used to detect collisions when a function is iterated, as proposed by Quisquater and Delescaille [13], and van Oorschot and Wiener [15].

In 2003, Oechslin [12] introduced the trade-off based on *rainbow tables* and demonstrated the efficiency of his technique by recovering Windows passwords. A rainbow table uses a different reduction function for each column of the table. Thus two different chains can merge only if they have the same key at the same position of the chain. This makes it possible to generate much larger tables. Actually, a rainbow table acts almost as if each column of the table was a separate single classic[1] table. Indeed, collisions within a classic table (or a column of a rainbow table) lead to merges whereas collisions between different classic tables (or different columns of a rainbow table) do not lead to a merge. This analogy can be used to demonstrate that a rainbow table of $mt$ chains of length $t$ has the same success rate as $t$ single classic tables of $m$ chains of length $t$. As the trade-off based on distinguished point, rainbow tables reduce the amount of look-ups by a factor of $t$, compared to the classic trade-off. Up until now, trade-

[1] By *classic* we mean the tables as described in the original Hellman paper.

off techniques based on rainbow tables are the most efficient ones. Recently, an FPGA implementation of rainbow tables has been proposed by Mentens, Batina, Preneel, and Verbauwhede [11] in order to retrieve Unix passwords.

Whether it is the classic Hellman trade-off, the distinguished points or the rainbow tables, they all suffer from a significant quantity of false alarms. Contrarily to what is claimed in the original Hellman paper, false alarms may increase the time complexity of the cryptanalysis by more than 50%. We will explain this point below. In this paper, we propose a technique whose goal is to reduce the time spent to detect false alarms. It works with the classic trade-off, with distinguished points, and with rainbow tables. Such an improvement is especially pertinent in practical cryptanalysis, where time-memory trade-offs are generally used to avoid to repeat an exhaustive search many times. For example, when several passwords must be cracked [12], each of them should not take more than a few seconds. In [1], the rainbow tables are used to speed up the search process in a special database. In such a commercial application, time is money, and therefore any improvement of time-memory trade-off also.

In Section 2, we give a rough idea of our technique based on checkpoints. We provide in Section 3 a detailed and formal analysis of the rainbow tables. These new results allow to formally compute the probability of success, the computation time, and the optimal size of the tables. Based on this analysis we can describe and evaluate our checkpoint technique in detail. We illustrate our method by cracking Windows passwords based on DES. In Section 4, we show how a trade-off can be characterized in general. This leads to the comparison of the three existing variants of trade-off. Finally, we give in Section 5 several implementation tips which significantly improve the trade-off in practice.

## 2 Checkpoint Primer

### 2.1 False Alarms

When the precalculation phase is achieved, a table containing $m$ starting points $S_1, \ldots, S_m$ and $m$ end points $E_1, \ldots, E_m$ is stored in memory. This table can be regenerated by iterating the function $f$, defined by $f(K) := \mathsf{R}(\mathsf{S}_K(P))$, on the starting points. Given a row $j$, let $X_{j,i+1} := f(X_{j,i})$ be the $i$-th iteration of $f$ on $S_j$ and $E_j := X_{j,t}$. We have:

$$\begin{array}{rcccccccccl}
S_1 = & X_{1,1} & \xrightarrow{f} & X_{1,2} & \xrightarrow{f} & X_{1,3} & \xrightarrow{f} & \ldots & \xrightarrow{f} & X_{1,t} & = E_1 \\
S_2 = & X_{2,1} & \xrightarrow{f} & X_{2,2} & \xrightarrow{f} & X_{2,3} & \xrightarrow{f} & \ldots & \xrightarrow{f} & X_{2,t} & = E_2 \\
\vdots & & & & & & & & & & \vdots \\
S_m = & X_{m,1} & \xrightarrow{f} & X_{m,2} & \xrightarrow{f} & X_{m,3} & \xrightarrow{f} & \ldots & \xrightarrow{f} & X_{m,t} & = E_m
\end{array}$$

In order to increase the probability of success, i.e., the probability that $K$ appears in the stored values, several tables with different reduction functions are generated.

Given a ciphertext $C = \mathsf{S}_K(P)$, the on-line phase of the cryptanalysis works as follows: $\mathsf{R}$ is applied on $C$ in order to obtain a key $Y_1$, and then the function $f$ is iterated on $Y_1$ until matching any $E_j$. Let $s$ be the length of the generated chain from $Y_1$:

$$C \quad \xrightarrow{\mathsf{R}} \quad Y_1 \quad \xrightarrow{f} \quad Y_2 \quad \xrightarrow{f} \quad \dots \quad \xrightarrow{f} \quad Y_s$$

Then the chain ending with $E_j$ is regenerated from $S_j$ until yielding the expected key $K$. Unfortunately $K$ is not in the explored chain in most of the cases. Such a case occurs when $\mathsf{R}$ collides: the chain generated from $Y_1$ merged with the chain regenerated from $S_j$ after the column where $Y_1$ is. That is a false alarm, which requires $(t-s)$ encryptions to be detected.

Hellman [8] points out that the expected computation due to false alarms increases the expected computation by at most 50 percent. This reasoning relies on the fact that, for any $i$, $f^i(Y_1)$ is computed by iterating $f$ $i$ times. However $f^i(Y_1)$ should be computed from $Y_i$ because $f^i(Y_1) = f(Y_i)$. In this case, the computation time required to reach a chain's end is significantly reduced on average while the computation time required to rule out false alarms stays the same. Therefore, false alarms can increase by more than 50 percent the expected computation. For example, formulas given in Section 3 allow to determine the computation wasted during the recovering of Windows passwords [12]: false alarms increase by 125% the expected computation.

## 2.2 Ruling Out False Alarms Using Checkpoints

Our idea consists in defining a set of positions $\alpha_i$ in the chains to be checkpoints. We calculate the value of a given function $G$ for each checkpoint of each chain $j$ and store these $G(X_{j,\alpha_i})$ with the end of each chain $X_{j,t}$. During the on-line phase, when we generate $Y_1, Y_2, \dots, Y_s$, we also calculate the values for $G$ at each checkpoint, yielding the values $G(Y_{\alpha_i+s-t})$ . If $Y_s$ matches the end of a chain that we have stored, we compare the values of $G$ for each checkpoint that the chain Y has gone through with the values stored in the table. If they differ at least for one checkpoint we know that this is a false alarm. If they are identical, we cannot determine if a false alarm will occur without regenerating the chain.

In order to be efficient, $G$ should be easily computable and the storage of its output should require few bits. Below, we consider the function $G$ such that $G(X)$ simply outputs the less significant bit of $X$. Thus we have:

$$\Pr\{G(X_{j,\alpha}) \neq G(Y_{\alpha+s-t}) \mid X_{j,\alpha} \neq Y_{\alpha+s-t}\} = \frac{1}{2}\left(1 - \frac{1}{2^{|K|}}\right) \approx \frac{1}{2}.$$

The case $X_{j,\alpha} \neq Y_{\alpha+s-t}$ occurs when the merge appears after the column $\alpha$ (Fig 1). The case $X_{j,\alpha} = Y_{\alpha+s-t}$ occurs when either $K$ appears in the regenerated chain or the merge occurs before the column $\alpha$ (Fig. 2).

In the next section we will analyze the performances of perfect rainbow tables in detail. Then, we will introduce the checkpoint concept in rainbow tables and analyze both theoretical and practical results.

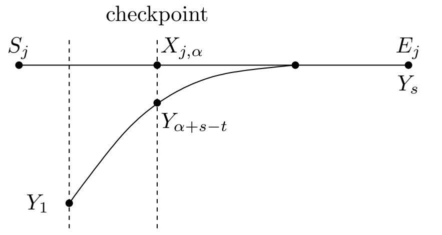


**Fig. 1.** False alarm detected with probability 1/2

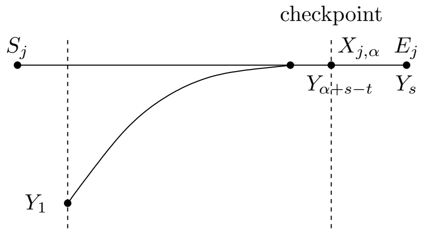


**Fig. 2.** False alarm not detected

## 3 Perfect Rainbow Tables and Checkpoints

### 3.1 Perfect Tables

The key to an efficient trade-off is to ensure that the available memory is used most efficiently. Thus we want to avoid the use of memory to store chains that contain elements which are already part of other chains. To do so, we first generate more chains than we actually need. Then we search for merges and remove chains until there are no merges. The resulting tables are called perfect tables. They have been introduced by [4] and analyzed by [14]. Creating perfect rainbow and DP tables is easy since merging chains can be recognized by their identical end points. Since end points need to be sorted to facilitate the look-ups, identifying the merges comes for free. Classic chains do not have this advantage. Every single element of every classic chain that is generated has to be looked up in all elements of all chains of the same table. This requires $mt\ell$ look-ups in total where $\ell$ is the number of stored tables. A more efficient method of generating perfect classic tables is described in [2]

Perfect classic and DP tables are made of unique elements. In perfect rainbow tables, no element appears twice in any given column, but it may appear more than once across different columns. This is consistent with the view that each column of a rainbow table acts like a single classic table. In all variants of the trade-off, there is a limit to the size of the perfect tables that can be generated. The brute-force way of finding the maximum number of chains of given length $t$ that will not merge is to generate a chain from each of the $N$ possible keys and remove the merges.

In the following sections, we will consider perfect tables only.

### 3.2 Optimal Configuration

From [12], we know that the success rate of a single un-perfect rainbow table is $1 - \prod_{i=1}^{t} \left(1 - \frac{m_i}{N}\right)$ where $m_i$ is the number of different keys in column $i$. With perfect rainbow tables, we have $m_i = m$ for all $i$ s.t. $1 \le i \le t$. The success rate of a single perfect rainbow table is therefore

$$P_{\text{rainbow}} = 1 - \left(1 - \frac{m}{N}\right)^t . \tag{1}$$

The fastest cryptanalysis time is reached by using the largest possible perfect tables. This reduces the amount of duplicate information stored in the table and reduces the number of tables that have to be searched. For a given chain length $t$, the maximum number $m_{\max}(t)$ of rainbow chains that can be generated without merges is obtained (see [12]) by calculating the number of independent elements at column $t$ if we start with $N$ elements in the first column. Thus we have

$$m_{\max}(t) = m_t \text{ where } m_1 = N \text{ and } m_{n+1} = N\left(1 - e^{-\frac{m_n}{N}}\right) \text{ where } 0 < n < t.$$

For non small $t$ we can find a closed form for $m_{\max}$ (see [2]):

$$m_{\max}(t) \approx \frac{2N}{t+2}.$$

From (1), we deduce the probability of success of a single perfect rainbow table having $m_{\max}$ chains:

$$P_{\text{rainbow}}^{\max} = 1 - \left(1 - \frac{m_{\max}}{N}\right)^t \approx 1 - e^{-t\frac{m_{\max}}{N}} \approx 1 - e^{-2} \approx 86\%.$$

Interestingly, for any $N$ and for $t$ not small, this probability tends toward a constant value. Thus the smallest number of tables needed for a trade-off only depends on the desired success rate $P$. This makes the selection of optimal parameters very easy (see [2] for more details):

$$\ell = \left\lceil \frac{-\ln(1-P)}{2} \right\rceil, \quad m = \frac{M}{\ell}, \text{ and } t = \frac{\ln(1-P)}{\ln(1-\frac{M}{\ell N})\ell} \approx \frac{-N}{M}\ln(1-P).$$

## 3.3 Performance of the Trade-Off

Having defined the optimal configuration of the trade-off, we now calculate the exact amount of work required during the on-line phase. The simplicity of rainbow tables makes it possible to include the work due to false alarms both for the average and the worst case.

Cryptanalysis with a set of rainbow tables is done by searching for the key in the last column of each table and then searching sequentially through previous columns of all tables. There are thus a maximum of $\ell t$ searches. We calculate the expectation of the cryptanalysis effort by calculating the probability of success and the amount of work for each search $k$. When searching a key at position $c$ of a table, the amount of work to generate a chain that goes to the end of the table is $t-c$. The additional amount of work due to a possible false alarm is $c$ since the chain has to be regenerated from the start to $c$ in order to rule out the false alarm. The probability of success in the search $k$ is given below:

$$p_k = \frac{m}{N}\left(1 - \frac{m}{N}\right)^{k-1}. \qquad (2)$$

We now compute the probability of a false alarm during the search $k$. When we generate a chain from a given ciphertext and look-up the end of the chain in

**Table 1.** Calculated and measured performance of rainbow tables

| $N = 8.06 \times 10^{10}$, $t = 10000$, $m = 15408697$, $\ell = 4$ | theory | measured over 1000 experiments |
|---|---|---|
| encryptions (average) | $1.55 \times 10^7$ | $1.66 \times 10^7$ |
| encryptions (worst case) | $2.97 \times 10^7$ | $2.96 \times 10^8$ |
| number of false alarms (average) | 1140 | 1233 |
| number of false alarms (worst case) | 26048 | 26026 |

the table, we can either not find a matching end, find the end of the correct chain or find an end that leads to a false alarm. Thus we can write that the probability of a false alarm is equal to one minus the probability of actually finding the key minus the probability of finding no end point. The probability of not finding an end point is the probability that all points that we generate are not part of the chains that lead into the end points. At column $i$, these are the $m_i$ chains that we used to build the table. The probability of a false alarm at search $k$ (i.e., in column $c = t - \lfloor \frac{k}{\ell} \rfloor$) is thus the following:

$$q_c = 1 - \frac{m}{N} - \prod_{i=c}^{i=t} \left(1 - \frac{m_i}{N}\right) \tag{3}$$

where $c = t - \left\lfloor \frac{k}{\ell} \right\rfloor$, $m_t = m$, and $m_{i-1} = -N \ln(1 - \frac{m_i}{N})$. When the tables have exactly the maximum number of chains $m_{\max}$ we find a short closed form for $q_c$ (see [2] for more details):

$$q_c = 1 - \frac{m}{N} - \frac{c(c+1)}{(t+1)(t+2)}. \tag{4}$$

The average cryptanalysis time is thus:

$$T = \sum_{\substack{k=1 \\ c=t-\lfloor \frac{k}{\ell} \rfloor}}^{k=\ell t} p_k \left(W(t-c-1) + Q(c)\right) \ell + (1 - \frac{m}{N})^{\ell t} \left(W(t) + Q(1)\right) \ell \tag{5}$$

where $$W(x) = \sum_{i=1}^{i=x} i \quad \text{and} \quad Q(x) = \sum_{i=x}^{i=t} q_i i.$$

The second term of (5) is the work that is being carried out every time no key is found in the table while the first term corresponds to the work that is being carried out during the search $k$. $W$ represents the work needed to generate a chain until matching a end point. $Q$ represents the work to rule out a false alarm. We can rewrite (5) as follows:

$$T = \sum_{\substack{k=1 \\ c=t-\lfloor \frac{k}{\ell} \rfloor}}^{k=\ell t} p_k \left( \sum_{i=1}^{i=t-c-1} i + \sum_{i=c}^{i=t} q_i i \right) \ell + (1-\frac{m}{N})^{\ell t} \left( \sum_{i=1}^{i=t} i + \sum_{i=1}^{i=t} q_i i \right) \ell$$

$$= \sum_{\substack{k=1 \\ c=t-\lfloor \frac{k}{\ell} \rfloor}}^{k=\ell t} p_k \left( \frac{(t-c)(t-c-1)}{2} + \sum_{i=c}^{i=t} q_i i \right) \ell + (1-\frac{m}{N})^{\ell t} \left( \frac{t(t-1)}{2} + \sum_{i=1}^{i=t} q_i i \right) \ell$$

We have run a few experiments to illustrate $T$. The results are given in Table 1.

### 3.4 Checkpoints in Rainbow Tables

From results of Section 3.3, we establish below the gain brought by the checkpoints. We firstly consider only one checkpoint $\alpha$. Let $Y_1 \dots Y_s$ be a chain generated from a given ciphertext $C$. From (3), we know that the probability that $Y_1 \dots Y_s$ merges with a stored chain is $q_{t-s}$. The expected work due to a false alarm is therefore $q_{t-s}(t-s)$.

We now compute the probability that the checkpoint detects the false alarm. If the merge occurs before the checkpoint (Fig. 2) then the false alarm cannot be detected. If the chain is long enough, i.e., $\alpha + s > t$, the merge occurs after the checkpoint (Fig. 1) with probability $q_\alpha$. In this case, the false alarm is detected with probability $\Pr\{G(X_{j,\alpha}) \neq G(Y_{\alpha+s-t}) \mid X_{j,\alpha} \neq Y_{\alpha+s-t}\}$.

We define $g_\alpha(s)$ as follows:

$$g_\alpha(s) = \begin{cases} 0 \text{ if there is no checkpoint in column } \alpha, \\ 0 \text{ if } (\alpha+s) \leq t \text{, i.e. the generated chain does not reach column } \alpha, \\ \Pr\{G(X_{j,\alpha}) \neq G(Y_{\alpha+s-t}) \mid X_{j,\alpha} \neq Y_{\alpha+s-t}\} \text{ otherwise.} \end{cases}$$

We can now rewrite $Q(x) = \sum_{i=x}^{i=t} i \left( q_i - q_\alpha \cdot g_\alpha(t-i) \right)$.

We applied our checkpoint technique with $N = 8.06 \times 10^{10}$, $t = 10000$, $m = 15408697$, $\ell = 4$ and $G$ as defined in Section 2.2. Both theoretical and experimental results are plotted on Fig. 3.

We can generalize to $t$ checkpoints. We can rewrite $Q(x)$ as follows:

$$Q(x) = \sum_{i=x}^{i=t} i \left( q_i - q_i \cdot g_i(t-i) - \sum_{j=i+1}^{j=t} \left( q_j \cdot g_j(t-j) \prod_{k=i}^{k=j-1} (1 - g_k(t-k)) \right) \right).$$

We now define memory cost and time gain. Let $M$, $T$, $N$ and $M'$, $T'$, $N'$ be the parameters of two trade-offs respectively. We define $\sigma_M$ and $\sigma_T$ as follows:

$$M' = \sigma_M \cdot M \quad \text{and} \quad T' = \sigma_T \cdot T.$$

The memory cost of the second trade-off over the first one is straightforwardly defined by $(\sigma_M - 1) = M'/M - 1$ and the time gain is $(1-\sigma_T) = 1 - T'/T$. When

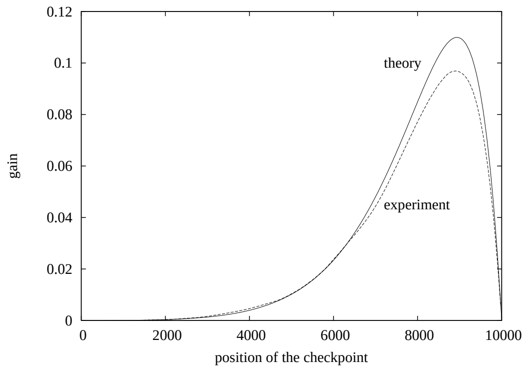


**Fig. 3.** Theoretical and experimental gain when one checkpoint is used

**Table 2.** Cost and gain of using checkpoint in password cracking, with $N = 8.06 \times 10^{10}$, $t = 10000$, $m = 15408697$, and $\ell = 4$

| Number of checkpoints | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Cost (memory) | 0.89% | 1.78% | 2.67% | 3.57% | 4.46% | 5.35% |
| Gain (time) storing chains | 1.76% | 3.47% | 5.14% | 6.77% | 8.36% | 9.91% |
| Gain (time) storing checkpoints | 10.99% | 18.03% | 23.01% | 26.76% | 29.70% | 32.04% |
| Optimal checkpoints | 8935 | 8565<br>9220 | 8265<br>8915<br>9370 | 8015<br>8655<br>9115<br>9470 | 7800<br>8450<br>8900<br>9250<br>9550 | 7600<br>8200<br>8700<br>9000<br>9300<br>9600 |
| | ± 5 | ± 5 | ± 5 | ± 5 | ± 50 | ± 100 |

a trade-off stores more chains, it implies a memory cost. Given that $T \propto N^2/M^2$ the time gain is:

$$(1 - \frac{T'}{T}) = 1 - \frac{1}{\sigma_M^2}.$$

Instead of storing additional chains, the memory cost can be used to store checkpoints. Thus, given a memory cost, we can compare the time gains when the additional memory is used to store chains and when it is used to store checkpoints. Numerical results are given in Table 2.

The numerical results are amazing. An additional 0.89% of memory saves about 10.99% of cryptanalysis time. This is six times more than the 1.76% of gain that would be obtained by using the same amount of memory to store additional chains. Our checkpoints thus perform much better than the basic trade-off. As we add more and more checkpoints, the gain per checkpoint decreases. In our example it is well worth to use 6 bits of checkpoint values (5.35% of additional memory) per chain to obtain a gain of 32.04%. The 0.89% of memory per checkpoint are calculated by assuming that the start and the end of the chains are stored in 56 bits each, as our example uses DES keys. As we explain in Section 5 the amount of bits used to store chain can be optimized and reduced to 49 bits in our example. In this case a bit of checkpoint data adds 2% of memory and it is still well worth using three checkpoints of one bit each to save 23% of work.

## 4 Characterization and Comparison of Trade-Offs

In this section we give a generic way of characterizing the different variants of the trade-off. We calculate the characteristic of rainbow tables exactly and compare it to measured characteristics of other variants.

### 4.1 Time-Memory Graphs

Knowing how to calculate the success rate and the number of operations needed to invert a function, we can now set out to plot the time-memory graphs. In order to do so, we fix a given success rate and for each memory size we find the table configuration that yields the fastest trade-off and plot the time that it takes. The graphs show that cryptanalysis time decreases with the square of the memory size, independently of the success rate. We can thus write the time-memory relation as

$$T = \frac{N^2}{M^2}\gamma(P) \tag{6}$$

where $\gamma(P)$ is a factor that depends only on the success probability. It is interesting to note that for $P = 86\%$ which is the the maximum success probability of a single rainbow table, the factor is equal to 1. In that case we find the typical trade-off which was already described by Hellman, namely that $M = T = N^{\frac{2}{3}}$.

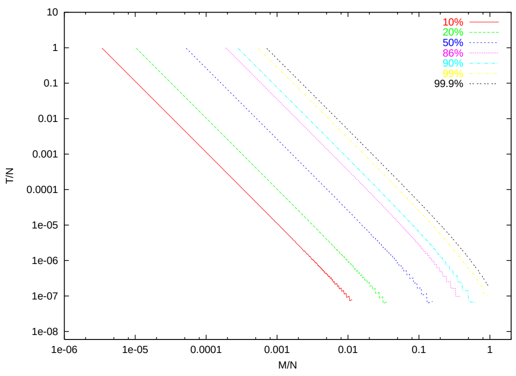


**Fig. 4.** Time-Memory graphs for rainbow tables, with various success rates.For $P_{\text{rainbow}} = 86\%$ the graph follows exactly $T = N^2/M^2$

Note that this simple expression of the trade-off performance was not possible for the previous variants. In those cases, calculations were always based on non-perfect tables, on the worst case (the key is not found in any table) and ignoring the amount of work due to false alarms. Optimizations have been proposed with these limitations, but to our knowledge the actual average amount of work, including false alarms has never been used to find optimal parameter. Our simple formula allows for a very simple calculation of the optimal parameters when any two of the success rate, the inversion time or the memory are given.

## 4.2 The Time-Memory Characteristic

The previous section confirms that rainbow tables follow the same $T \propto N^2/M^2$ relation as other variants of the trade-off. Still, they seem to perform better. We thus need a criterion to compare the trade-offs. We propose to use $\gamma(P)$ as the trade-off characteristic. The evolution of $\gamma$ over a range of $P$ shows how a variant is better than another. Figure 5 shows a plot of $\gamma(P)$ for rainbow tables:

In the following sections, we compare the performance of rainbow tables with the performance of classic tables and DP tables. DP tables are much harder to analyze because of the variable length of the chains. We will thus concentrate on classic tables first.

## 4.3 Classic and DP Tables

The trade-off using classic or DP tables can also be characterized using the $\gamma$ factor. Indeed both trade-offs follow the $T \propto N^2/M^2$ relation in a large part of the parameter space up to a factor which depends of the success rate and the type of trade-off. We have first devised a strategy to generate the largest possible perfect tables for each variant of the trade-off and have then used as many tables as necessary to reach a given success rate. The details of this work

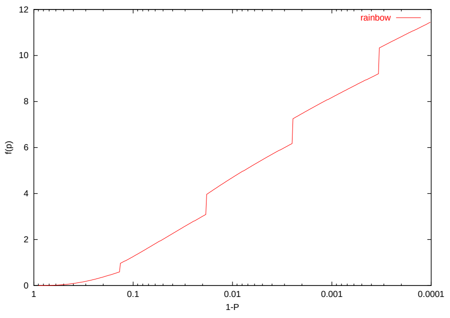


**Fig. 5.** The Time-Memory characteristic of rainbow tables.Steps happen every time an additional table has to be used to achieve the given success probability.

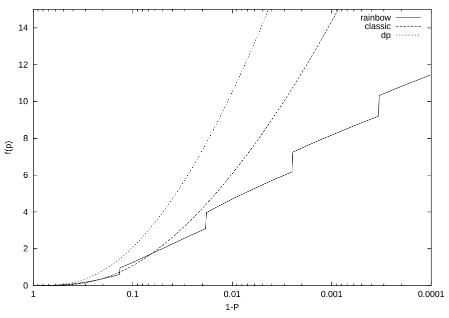


**Fig. 6.** The Time-Memory characteristics of rainbow, classic and DP tables compared

and the resulting time-memory graphs are available in [2]. In Figure 6 we show the evolution of the trade-off characteristic of classic tables and of DP tables.

The experiments and analysis show that rainbow tables outperform classic tables and DP tables for success rates above 80%. Below this limit, perfect classic tables are slightly better than perfect rainbow tables in terms of hash operations needed for cryptanalysis. However, the price of using classic tables is that they need $t$ times more table look-ups. Since these do not come for free in most architectures (content addressable memory could be an exception), rainbow tables seem to be the best option in any case.

## 5 Implementation Tips

For the sake of completeness we want to add some short remarks on the optimized implementation of the trade-offs. Indeed, an optimized implementation can yield performance gains almost as important as the algorithmic optimizations. We limit our tips to the implementation of rainbow tables.

### 5.1 Storing the Chain End Points

The number of operations of the trade-off decreases with the square of the available memory. Since available memory is measured in bytes and not in number of chains, it is important to choose an efficient format for storing the chains. A first issue is whether to use inputs or outputs of the function to be inverted (keys or ciphertexts) as beginning and end of chains. In practice the keys are usually smaller than the ciphertexts. It is thus more efficient to store keys (the key at the end of the chain has no real function but the extra reduction needed to generate it from the last ciphertext is well worth the saved memory). A second and more important issue is that we can take advantage of the way the tables are organized. Indeed a table consists of pairs of beginnings and ends of chains. To facilitate look-ups the chains are sorted by increasing values of the chain ends. Since the ends are sorted, successive ends often have an identical prefix. As suggested in [3] we can thus remove a certain length of prefix and replace it by an index table that indicates where every prefix starts in the table.

In our Windows password example, there are about $2^{37}$ keys of 56 bits. Instead of storing the 56 bits, we store a 37 bit index. From this index we take 21 bits as prefix and store only the last 16 bits in memory. We also store a table with $2^{21}$ entries that point to the corresponding suffixes for each possible prefix.

### 5.2 Storing the Chain Starting Points

The set of keys used for generating all the chains is usually smaller that the total set of keys. Since rainbow tables allow us to choose the starting points, we can use keys with increasing value of their index. In our example we used about 300 million starting points. This value can be expressed in 29 bits, so we only need to store the 29 lower bits of the index. The total amount of memory needed to store a chain is thus $29 + 16$ bits for the start and the end. The table that relates the prefixes to the suffixes incurs about 3.5 bits per chain. Altogether we thus need 49 bits per chain. A simple implementation that stores the full 56 bits of the start and end chain would need 2.25 times more memory and be 5 times slower.

### 5.3 Storing the Checkpoints

For reasons of efficiency of memory access it may in some implementations be more efficient to store the start and the end of a chain (that is, its suffix) in multiples of 8 bits. If the size of some parameters does not exactly match the size of the memory units, the spare bits can be used to store checkpoints for free. In our case, the 29 bits of the chain start are stored in a 32 bit word, leaving 3 bits available for checkpoints.

## 6 Conclusion

We have introduced a new optimization for cryptanalytic time-memory trade-offs which performs much better than the usual $T \propto N^2/M^2$. Our method works by reducing the work due to false alarms. Since this work is only a part of the total work our method can not reduce the work indefinitely. Besides having better performance, checkpoints can be generated almost for free while generating the trade-off tables. There is thus no indication for not using checkpoints and we conjecture that they will be used in many future implementations of the trade-off. Also, checkpoints are a new concept in time-memory trade-offs and they may lead to further optimizations and applications. In order to analyze the gain due to checkpoints we have presented a complete analysis of the rainbow tables. Using this analysis we are able to predict the gain that can be achieved with checkpoints. Finally we have also presented a simple way of comparing the existing variants of the trade-off with a so-called trade-off characteristic. We have calculated this characteristic for rainbow tables and measured it for the other variants. The results show that rainbow tables outperform the other variants in all cases except when table look-ups are free and the success probability is below 80%. The fact that the cryptanalysis time decreases with the square of the number of elements stored in memory indicates that it is very important to reduce the memory usage. This is why we have shared our tips on how this can be achieved in practice.

## References

1. Gildas Avoine, Etienne Dysli, and Philippe Oechslin. Reducing time complexity in RFID systems. In *Selected Areas in Cryptography – SAC 2005*, Lecture Notes in Computer Science, Kingston, Canada, August 2005. Springer-Verlag.
2. Gildas Avoine, Pascal Junod, and Philippe Oechslin. Time-memory trade-offs: False alarms detection using checkpoints. Technical Report LASEC-REPORT-2005-002, Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASEC), Lausanne, Switzerland, September 2005.

3. Alex Biryukov, Adi Shamir, and David Wagner. Real time cryptanalysis of A5/1 on a PC. In *Fast Software Encryption – FSE'00*, volume 1978 of *Lecture Notes in Computer Science*, pages 1–18, New York, USA, April 2000. Springer-Verlag.
4. Johan Borst, Bart Preneel, and Joos Vandewalle. On the time-memory tradeoff between exhaustive key search and table precomputation. In *Symposium on Information Theory in the Benelux*, pages 111–118, Veldhoven, The Netherlands, May 1998.
5. Dorothy Denning. *Cryptography and Data Security*, page 100. Addison-Wesley, Boston, Massachusetts, USA, June 1982.
6. Amos Fiat and Moni Naor. Rigorous time/space tradeoffs for inverting functions. In *ACM Symposium on Theory of Computing – STOC'91*, pages 534–541, New Orleans, Louisiana, USA, May 1991. ACM, ACM Press.
7. Amos Fiat and Moni Naor. Rigorous time/space tradeoffs for inverting functions. *SIAM Journal on Computing*, 29(3):790–803, December 1999.
8. Martin Hellman. A cryptanalytic time-memory trade off. *IEEE Transactions on Information Theory*, IT-26(4):401–406, July 1980.
9. Iljun Kim and Tsutomu Matsumoto. Achieving higher success probability in time-memory trade-off cryptanalysis without increasing memory size. *IEICE Transactions on Communications/Electronics/Information and Systems*, E82-A(1):123–, January 1999.
10. Koji Kusuda and Tsutomu Matsumoto. Optimization of time-memory trade-off cryptanalysis and its application to DES, FEAL-32, and Skipjack. *IEICE Transactions on Fundamentals*, E79-A(1):35–48, January 1996.
11. Nele Mentens, Lejla Batina, Bart Preneel, and Ingrid Verbauwhede. Cracking Unix passwords using FPGA platforms. SHARCS - Special Purpose Hardware for Attacking Cryptographic Systems, February 2005.
12. Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *Advances in Cryptology – CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630, Santa Barbara, California, USA, August 2003. IACR, Springer-Verlag.
13. Jean-Jacques Quisquater and Jean-Paul Delescaille. How easy is collision search? Application to DES (extended summary). In *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 429–434, Houthalen, Belgium, April 1989. IACR, Springer-Verlag.
14. François-Xavier Standaert, Gael Rouvroy, Jean-Jacques Quisquater, and Jean-Didier Legat. A time-memory tradeoff using distinguished points: New analysis & FPGA results. In *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 593–609, Redwood Shores, California, USA, August 2002. Springer-Verlag.
15. Michael Wiener and Paul van Oorschot. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, March 1999.

# Cryptanalysis of Barni et al. Watermarking Scheme

Tanmoy Kanti Das and Jianying Zhou

Infocomm Security Department, Institute for Infocomm Research,
21 Heng Mui Keng Terrace, Singapore 119613
{tkdas, jyzhou}@i2r.a-star.edu.sg

**Abstract.** In this paper, we analyze a wavelet based watermarking scheme proposed by Barni et al in 2001. The said scheme modifies high frequency wavelet coefficients to embed a watermark in an image. The scheme employs well known HVS model during embedding to embed the watermark in a perceptually transparent manner. Here we present a successful cryptanalysis of the scheme using single watermarked copy. In the process of cryptanalysis, we recover the watermark signal from the marked image and the attack is similar to cryptographic key recovery attack. Furthermore, we not only successfully recover the secret watermark signal from the attacked image, but also we are able to remove the watermark. This raises a serious question about the security and usability of the watermarking scheme in any practical system.

**Keywords:** Cryptanalysis, Watermark Recovery, Wavelet Transform, Copy Attack.

## 1 Introduction

Rapid growth in the popularity of Internet file sharing applications has led to a situation where protection of copyright became a serious problem for the content providers. Entertainment industry is loosing millions of dollars due to widespread illegal sharing of copyrighted material. Several technologies were developed to tackle the rampant abuse of copyright and watermarking is one such technology which is ( probably ) most effective. A lot of watermarking techniques have been developed to embed invisible copyright information in the image and they employ different kinds of signal processing techniques. Some of these schemes also embed user specific watermark(s) for fingerprinting. Further, these schemes should withstand several benchmark attacks available in the existing literature. However, "*Benchmarking is not really a security evaluation, but mainly a robustness evaluation*" [6]. In the absence of proper security evaluation, these schemes depend on the sophisticated signal processing techniques to earn user confidence and acceptability. Unlike cryptographic schemes, few attempts have been made to analyze each of the popular watermarking schemes and to study customized attacks in highlighting their weaknesses. Commercial use of watermarking schemes without proper security analysis is a risky endeavor as hackers may find it easy

© Springer-Verlag Berlin Heidelberg 2005

to break, causing considerable financial loss to the media owner. In this paper, we concentrate on a specific watermarking scheme described in [2] and present a very strong attack that successfully *recovers* and *removes* the secret watermark bits from the watermarked image.

Most of the watermarking schemes follow the same design paradigm and embed a secret signal in the host media to generate the marked media. Also, embedding process should not introduce any visible or statistical distortions. Let $I$ be the original host image and $s$ be the secret watermark, then the embedding process can be presented as $I^w = I + s$. Note that, we consider the image $I$ as a two dimensional matrix only. And *minor* changes in values of the matrix cause very little visible changes in the resultant image and it remains visually indistinguishable from the original image $I$. During the lifetime of a watermarked image, the image may undergo different kinds of processing, some of them are malicious and directed towards the watermark removal. Thus, watermark signal should be robust enough to withstand malicious processing. To resolve any copyright dispute, the owner extracts the mark from the disputed image $I^{\#}$ to prove his/her ownership. Most of these watermark verification processes are correlation based, i.e., they rely on the correlation between the recovered signal $s^{\#} = I^{\#} - I$ and watermark signal $s$ as the measure of confidence in the detection process. Many watermarking schemes embed user specific watermark $s^{(i)}$ for forensic tracking of users violating the copyright. Thus a robust watermarking scheme should not only ensure that it correctly identify the malicious buyer, but also the probability of false positive ( i.e. probability of wrongly implicating an honest buyer ) should be very small.

A scheme is considered to be secured if an attacker, who has access to the algorithmic principle of the scheme but has no access to the key, should not be able to tamper with the watermark [1, 9]. This cryptographic principle was first introduced by Kerckohoffs [11] in 1883. There are several examples to prove that "Security-by-Obscurity" (the assumption that opponent will remain ignorant about the system being used ) can't work – the latest one being the Secure Digital Music Initiative (SDMI) challenge [3]. Thus, given a watermarked image $I^w$, the challenge in front of the attacker is to construct an image $I^{\#}$ in such a manner that $I^{\#}$ is indistinguishable from I and $s^{\#}$ is uncorrelated with $s^{(i)}$. Then it is not possible to identify the malicious buyer $i$ any more. A robust watermarking scheme should defend such an attack.

Image processing based benchmark attacks like Stirmark [17, 16] are very popular with the designers of watermarking schemes and most of the recently proposed schemes are robust to it. However, security against collusion attack [8] is a serious issue, as most of the existing schemes are vulnerable against it. On the other hand, collusion attack may not be practical as it requires a large number of watermarked copies to be successful. Thus attacks based on single watermarked copy are gaining lot of attention [12, 10, 15, 6]. According to [12], there are lots of redundant clips in an audio/video. These clips can be used interchangeably as they are almost replica of each other, or they can be combined to form a new clip to replace one of the existing clips. Now, these type of alteration is enough to defeat many watermarking schemes and this is the basic idea of replacement attack reported in [12]. Attacks reported in [10, 15] can be mounted on those schemes where decoder is in public domain for watermark verification. Another class of attack directed towards the correlator detectors that does not use original image for detection, is ambiguity attack [4]. Also authors of [5] use the ambiguity attack to design a new zero-knowledge watermarking scheme. Recently Cayre et al [6] successfully mounted an attack on "wide spread spectrum(WSS)" based watermarking techniques, in which authors propose a key recovery attack using Blind Source Separation tools like Principal Component Analysis and Independent Component Analysis. Here, we introduce an attack which employs completely different methodology from those attacks and it is analogous to cryptographic *key recovery attack*. We not only recover most of the secret watermark signal, but also we remove the watermark signal from the attacked images. For the attacker, we also introduce techniques to indirectly measure the absence of watermark in the attacked image.

In this paper, we study the watermarking scheme by Barni et al [2] from the cryptographic point of view. Our attack is based on single watermarked copy and recovers most of the watermark successfully. Further, we are able to remove the watermark information from the attacked images. Successful recovery of secret watermark can pave the way for mounting the "copy attack" [13] on the scheme. Also, the original and recovered watermarks possess high correlation with the watermarked image, thus both the attacker and the owner can claim the ownership of the image in the absence of trusted third party registration of watermarks. In Section 2, we describe the scheme proposed in [2] and the attack is described in Section 3.

## 2 The Wavelet Based Watermarking Scheme

Recent growth of interest in the wavelet based watermarking schemes and wavelet transform in general started with the introduction of JPEG 2000, a wavelet based compression technique for images. Basic idea behind the wavelet transform is to study different frequency components of a signal. Given a signal $f(t)$, one may want to study the frequency components locally in time. The popular fourier transform can provide the frequency components of $f$ but time localization cannot be easily achieved by fourier transform. Windowed fourier transform can achieve time localization to certain extent by applying fourier transform on a slice of $f$ appeared between the time interval $t_i$ and $t_j$. However, wavelet transform [7] can provide a better time localization than windowed fourier transform.

Let us explain the wavelet transform using Haar basis [18]. Wavelet transform is basically a one dimensional transform and to apply it to images, one must perform it along the row first, along the column next. Consider the following one dimensional signal $I = [11, 5, 7, 15]$ consisting of four samples. Haar wavelet transform first takes the pairwise averages and then calculates the half of the subtracted values. So, after first level of wavelet transform the coefficients look like $I_{wav}^{L=1} = [\frac{11+5}{2}, \frac{7+15}{2}, \frac{11-5}{2}, \frac{7-15}{2}] = [8, 11, 3, -4]$. First two coefficients are av-

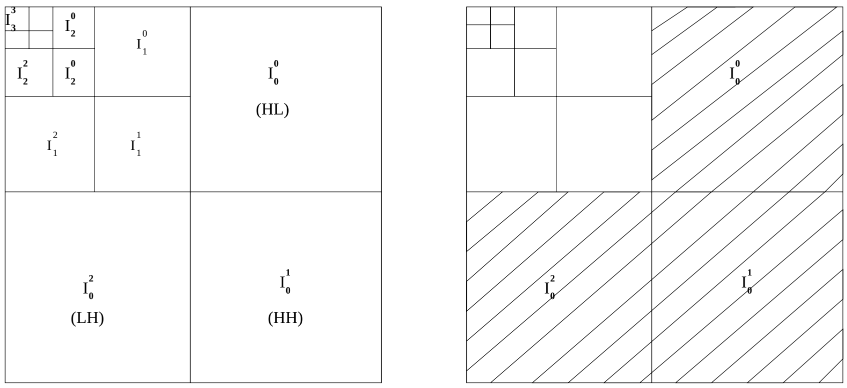


**Fig. 1.** Left: 2D Wavelet Transform, Right: Coefficients used for Watermarking (Shaded Region)

erages and known as low frequency components. Last two coefficients (half of the subtracted values) are known as detail or high frequency coefficients. In the next level of wavelet transform, normally the low frequency coefficients are subjected to further processing. Thus, $I_{wav}^{L=2} = [\frac{8+11}{2}, \frac{8-11}{2}, 3, -4] = [9.5, -1.5, 3, -4]$. As, in this example, there is only one low frequency component, we can not proceed further. Note that, one can also choose the high frequency coefficients at any level for next level of wavelet transform. This recursive computation of wavelet coefficients is based on the filter bank. Form the wavelet coefficients one can get back the original signal by pairwise addition and subtraction. For example, we can get back the coefficients of the previous level by the operation $9.5 \pm -1.5$.

To extend these ideas to images, we consider an image as a 2D signal and apply wavelet transform separately, first along the rows and then along the columns. In each level of wavelet transform four different bands are generated and they are denoted as $LL, HL, LH, HH$. One of the four bands is selected for next level of wavelet transform and the structure looks like a tree.

## 2.1 Watermark Embedding

The wavelet based watermarking algorithm proposed by Barni et al [2] embeds the watermark in the host image by modifying the high frequency wavelet coefficients. During watermark embedding, the host image $I$ is subjected to 4 levels of wavelet transformations. As can be seen from Figure 1, there are four wavelet subbands at each level. Let us denote these subbands by $I_l^\theta$, where $l$, $\theta$ denote the level and orientation of wavelet subbands, respectively. In popular literature, these four subbands $\theta \in 0, 1, 2, 3$ at each level are denoted by $HL, HH, LH, LL$. During watermark embedding, though forward wavelet transform is performed upto four levels, the wavelet coefficients generated from the first level of wavelet transform are only used for watermarking. This ensures minimal distortions to the watermarked image. Note that, coefficients from all the levels are used during measurement of sensitivity of human eyes to local image changes. We like to point out that for the wavelet transform Daubechis-6 [7] filter bank is used in [2].

A pseudorandom binary sequence consisting of $\pm 1$ is used as the watermark. The sequence is selected in such a manner that the mean and standard deviation of the sequence is equal to $0, 1$ respectively. Let us denote the watermark sequence by $x^\theta, \theta \in (0, 1, 2)$. Watermark is inserted in the host image by modifying the wavelet coefficients in the following manner.

$$\tilde{I}_l^\theta(i,j) = I_l^\theta(i,j) + \alpha x^\theta(i,j) w^\theta(i,j) \tag{1}$$

Here $\alpha$ is used to control the embedding strength and perceptual weighing function $w^\theta(i,j)$ is used as the measure of local sensitivity of the image $I$ to noise. As watermark is embedded in the high frequency wavelet coefficients at level 0 only, we can rewrite the equation 1 as follows.

$$\tilde{I}_0^\theta(i,j) = I_0^\theta(i,j) + \alpha x^\theta(i,j) w^\theta(i,j), \quad \theta \in 0,1,2 \tag{2}$$

**Perceptual Weighing.** A modified method of Lewis and Knowles [14] is used in [2] to compute the perceptual weighing function $w^\theta$ and its value is always positive. We introduce the method used to compute $w^\theta$ briefly here and refer to [2–page 785] for details.

$$w^\theta(i,j) = \frac{\Theta(l,\theta)\Lambda(l,i,j)\Xi(l,i,j)^{0.2}}{2} \tag{3}$$

Here $\Theta(l,\theta)$ is given by the following equation.

$$\Theta(l,\theta) = \begin{Bmatrix} \sqrt{2}, & if\ \theta = 1 \\ 1, & otherwise \end{Bmatrix} \cdot \begin{Bmatrix} 1.00, if\ l = 0 \\ 0.32, if\ l = 1 \\ 0.16, if\ l = 2 \\ 0.10, if\ l = 3 \end{Bmatrix} \tag{4}$$

One can compute $\Lambda(l,i,j)$ in the following manner.

$$\Lambda(l,i,j) = 1 + L'(l,i,j) \tag{5}$$

$$L'(l,i,j) = \begin{Bmatrix} 1 - L(l,i,j)\ if \quad L(l,i,j) \le 0.5 \\ L(l,i,j), \quad otherwise \end{Bmatrix} \tag{6}$$

$$L(l,i,j) = \frac{1}{256} I_3^3 \left(1 + \left\lfloor \frac{i}{2^{3-l}} \right\rfloor, 1 + \left\lfloor \frac{j}{2^{3-l}} \right\rfloor\right) \tag{7}$$

$\Xi$ gives an idea about the image texture around the pixel $(i,j)$ and consists of product of two terms. First term is the mean square value of all detail subbands

( i.e., $\theta \in 0, 1, 2$) and the second term is the variance of the subband at level $I_3^3$. $\Xi$ can be computed as follows.

$$\Xi(l,i,j) = \sum_{k=0}^{3-l} \frac{1}{16^k} \sum_{\theta=0}^{2} \sum_{x=0}^{1} \sum_{y=0}^{1} \left[ I_{k+l}^{\theta} \left( y + \frac{i}{2^k}, x + \frac{j}{2^k} \right) \right]^2 .$$

$$Var \left\{ I_3^3 \left( 1 + y + \frac{i}{2^{3-l}}, 1 + x + \frac{j}{2^{3-l}} \right) \right\}_{\substack{x=0,1 \\ y=0,1}} \quad (8)$$

## 2.2 Watermark Detection

Watermark detection is oblivious, i.e., original image is not required during watermark verification. The correlation between the marked wavelet coefficients and the embedded watermark acts as the measure of confidence in the detection process. If the correlation is greater than certain predefined threshold $T_\rho$, then the image is watermarked. Threshold $T_\rho$ is set in such a manner that the probability of false positive $P_f$ is negligible. If the size of the watermarked image is $2M \times 2N$, then the correlation $\rho$ can be computed as follows.

$$\rho = \frac{1}{3MN} \sum_{\theta=0}^{2} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \tilde{I}_0^{\theta}(i,j) x^{\theta}(i,j) \quad (9)$$

In [2] the false positive is set at $P_f \leq 10^{-8}$. Then the threshold is computed as $T_\rho = 3.97\sqrt{2\sigma_{\rho B}^2}$. Here $\sigma_{\rho B}$ is given by the following equation.

$$\sigma_{\rho B}^2 \approx \frac{1}{(3MN)^2} \sum_{\theta=0}^{2} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (\tilde{I}_0^{\theta}(i,j))^2 \quad (10)$$

## 3 Proposed Attack

Most of the recently proposed state of the art attacks are based on single watermarked copy and the proposed attack is no different. We successfully remove the watermark from the watermarked image using a single watermarked copy only. In addition to this, we are able to extract the secret watermark bits from the marked images which shows little attention being paid to the security of the scheme during design phase. Particularly the ease with which it is possible to extract the secret watermark bits, poses serious threat to security. Consider the following scenario. Alice registers her secret watermark bits with the certifying authority and uses the watermark to mark her image prior to distribution. This is done to protect the copyright from infringement and authentication. Now, Bob gets a watermarked copy from Alice and extracts her secret watermark. Then Bob can pass any image as originated from Alice by embedding Alice's watermark in it. This will undermine the credibility of Alice. If we replace Alice by a security agency which issues important document like passport with watermarked images of the passport holder and protects its authenticity by watermarking, then the security risk of using Barni et al scheme is too obvious. Also successful recovery of "secret watermark" can facilitate the way for mounting the *copy attack* [13] as mentioned here.

We will present the proposed attack in a stepwise manner. First, we will describe the watermark removal techniques which acts as the precursor to more significant *watermark recovery attack*. Barni et al watermarking scheme [2] modifies the high frequency wavelet coefficients at the first level to embed the watermark. A very naive attack would be to remove the high frequency wavelet coefficients at the first level altogether. Thus no trace of watermark will remain in the attacked image. However, this may degrade the quality of the attacked image. Let us first study the effect of wavelet coefficients removal on the image quality. We will use peak signal to noise ratio (PSNR) as a measure of image quality. During the attack we first perform forward wavelet transform upto level 4 of an image $I$. And after that we remove all the high frequency wavelet coefficients, i.e., $I_0^\theta(i,j) = 0$ for $\theta \in (0,1,2)$. Then reverse wavelet transform is performed to get back the attacked image. In Table 1, we present results related to this attack. Note that, wavelet coefficients removed from the first level are known as detail coefficients and they essentially store information regarding the edges present in an image. Thus removing these coefficients will ( to a little extent ) smudge the edges present in the image. However, as we can see from the experimental results presented in Table 1 and the image presented in Figure 2, the images are of acceptable quality. Next we present an attack which is as effective as this one but preserves the image quality better.

Addition of watermark can be treated as addition of noise to the host image. It is well known in image processing that filtering is an effective tool for denoising images. Thus filtering can be used to remove the watermark. Also, if the noise (i.e., watermark) gets removed (even partially) the resultant image gets closer to the original image. This implies that proper filtering can improve the image quality also. As the watermark/noise is inserted in the high frequency wavelet coefficients only, we can apply averaging filter to remove the watermark. In Table 3, we present experimental results related to averaging filter operation. It can be seen that watermark is not detectable in the attacked images.

Let us now analyze the situation from the attacker's point of view. Even if image processing operations are able to remove the watermark from the watermarked image, there is no way to verify whether the watermarks got removed or not as the attacker does not have any access to secret watermark bits. In the next subsection, we will show how to get back the actual watermarking bits which enables the attacker to verify whether watermarking signal got removed or not.

## 3.1 Recovering the Secret Watermark

Watermarking algorithm of [2] only verifies the presence or absence of the mark. Let us for the moment consider that the algorithm is non-oblivious and decoder has access to original host image $I$. Under these circumstances, one can easily extract the watermark from the marked image in the following manner.

$$x^\theta(i,j) = \frac{\tilde{I}_0^\theta(i,j) - I_0^\theta(i,j)}{\alpha w^\theta(i,j)} \tag{11}$$

Note that, if we assume that watermark consists of $\pm 1$ only, we may not need to know the actual values of $\alpha$ and $w^\theta(i,j)$. As both $\alpha$ and $w^\theta(i,j)$ are positive, we can easily compute the $x^\theta(i,j)$. However, the attacker does not have any access to the original image to extract the watermark. In the previous section, we have commented that the filtering removes the noise and the resultant image is much closer to the original image. Thus we can use the filtered image $I^A$ as approximate original in place of original image $I$. The exact algorithm to extract the watermark is as follows.

**Algorithm 1**

```
1. Read the watermarked image $\tilde{I}$ having size $2M \times 2N$.
2. Replace each pixel of $\tilde{I}$ by the average of itself and its neighbours. Let us
   denote the resultant image by $I^A$.
3. Perform forward wavelet transform on $\tilde{I}, I^A$ upto level 4 to get $\tilde{I}_0^\theta$ and $I_0^{A\theta}$.
4. For $i = 0 \dots M-1$ Do
   (a) For $j = 0 \dots N-1$ Do
       i. For $\theta = 0 \dots 2$ Do
          A. If $(\tilde{I}_0^\theta(i,j) - I_0^{A\theta}(i,j)) > 0$ Then $x^{A\theta}(i,j) = 1$
          B. Else $x^{A\theta}(i,j) = -1$
5. If $x^{A\theta}$ have mean $\approx 0$ and s.d. $\approx 1$ then the extracted watermark is a valid
   candidate.
```

Step 5 of algorithm 1 actually checks whether the extracted watermark is valid. This check is done to ensure that the extracted watermark satisfies the criteria used to generate the watermark at first place, i.e., mean and standard deviation of the watermark bits should be $0, 1$ respectively. However, this check does not conclusively ensure that the $\tilde{I}$ is actually watermarked by the owner using $x^{A\theta}$. In the next subsection we propose a method to verify that the extracted watermark is indeed used to mark the image.

## 3.2 Verification of Extracted Watermark

It is known that the presence of watermark can be detected without the original image. As the correlation between the two different pseudorandom sequences is zero, if we check the presence of watermark using bit sequence which is different from the one used to mark the image, the correlation $\rho$ will be negligible. Now this property can be used to predict whether the extracted watermark was actually used for watermarking the present image. One can see that the success of extraction procedure rest on proper approximation of original image. If the extracted watermark is correct, then the correlation between the extracted watermark and watermarked image will be significant. Thus we can clearly verify correctness of the extracted watermark by checking whether the extracted watermark possesses significant correlation with watermarked image. Here we consider the correlation $\rho$ as significant if $\rho > 0.5$. The exact algorithm is as follows.

**Algorithm 2**

1. *Read the watermarked image* $\tilde{I}$ *having size* $2M \times 2N$*.*
2. *Extract the watermark* $x^{A\theta}$ *using algorithm 1.*
3. *Detect the presence of watermark by Barni et al algorithm and using* $x^{A\theta}$ *as the watermark.*
4. *If* $\rho > 0.5$ *report* $I^A$ *as the attacked image and* $x^{A\theta}$ *as the correct watermark.*
5. *Otherwise report failure.*

Now we know whether the extracted watermark is valid or not. Let us assume that the extracted watermark is valid, then we can easily check whether the watermark got removed or not. We compute the correlation $\rho$ between the attacked image and the extracted watermark $x^{A\theta}$. If the correlation is negligible, i.e., $\rho \approx 0.00$, then the attacker is successful in removing the watermark from the attacked image. Also proper extraction of the watermark itself is a very strong attack which is similar to *key recovery attack* in cryptography. As the watermark is known to the attacker, the attacker can mount several other attacks including the copy attack [13].

## 4 Experimental Results

In this section, we present experimental results in support of our claims using a similar experimental setup as used in [2]. Daubechis-6 filterbank is used for wavelet transformations and different standard images having size $512 \times 512$ are used for experimentations. Let us first present the results regarding the removal of wavelet coefficients. An image is subjected to forward DWT up to four levels and high frequency coefficients at level $l = 0$ are removed. Here by removal, we

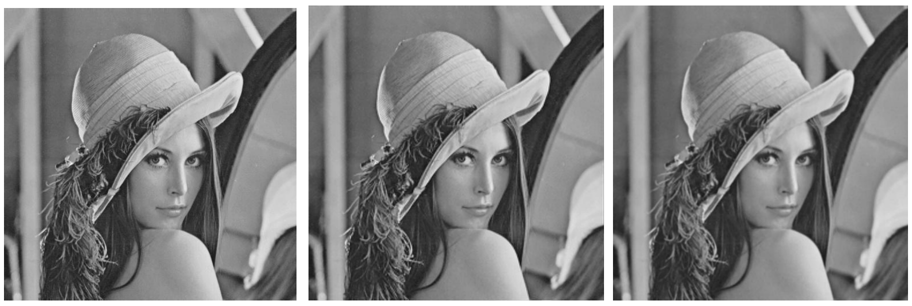


**Fig. 2.** Removal: Left:original, Middle: Coefficients Removed, Right: Filtered Images

**Table 1.** Coefficients Removal

| Image | PSNR(w,a) | PSNR(o,a) | $\rho$ | $T_\rho$ | Detector Response |
|---|---|---|---|---|---|
| Lena | 31.91 | 33.79 | 0.0013 | 0.0065 | Absent |
| Peppers | 29.86 | 30.74 | 0.0250 | 0.0666 | Absent |
| Boat | 30.13 | 30.88 | 0.0073 | 0.0483 | Absent |
| Elaine | 30.99 | 32.78 | 0.0006 | 0.0100 | Absent |
| F16 | 29.43 | 31.30 | 0.0003 | 0.0306 | Absent |

**Table 2.** Average Filtering

| Image | PSNR(w,a) | PSNR(o,a) | $\rho$ | $T_\rho$ | Detector Response |
|---|---|---|---|---|---|
| Lena | 32.06 | 33.83 | 0.0069 | 0.0329 | Absent |
| Peppers | 30.24 | 32.97 | 0.0110 | 0.0343 | Absent |
| Boat | 30.83 | 31.70 | 0.0230 | 0.0409 | Absent |
| Elaine | 30.94 | 32.91 | 0.0213 | 0.0303 | Absent |
| F16 | 29.77 | 31.88 | 0.0222 | 0.0439 | Absent |

mean that the coefficient value is replaced by zero. After that, reverse DWT is performed to get back the image. In Figure 2, we present the resultant image. As can be seen from Figure 2, as well as from the PSNR values in Table 1 that the resultant image is of very good quality. In table 1, PSNR(w,a) and PSNR(o,a) represent PSNR of attacked images with respect to watermarked and original images, respectively. It can be seen that the watermark cannot be detected from any of the attacked images. This proves effectiveness of our attack. Next we study the effectiveness of our second attack, where watermarked images are subjected to average filtering. During average filtering, we replace each pixel with the average of it and its neighboring pixel values. Actually the idea is to minimize the effect of high frequency noise. The resultant image is presented in Figure 2. Note that, Barni et al watermarking scheme embeds the watermark in the high frequency zone only. This is also evident from the figure 3 presented in [2]. Table 2 presents results related to this attack and it is evident from the results that the attack is effective in removing the watermark. PSNR of attacked images with respect to original and watermarked images are reported in the table. It is apparent that the PSNR of attacked images with respect to original images are better than the PSNR of the attacked images w.r.t watermarked images.

Most significant part of our attack is to recover the watermark from the watermarked images. We recover the watermark from the watermarked image using the average filtered image as the approximate original image. The result is presented in Table 3. The mean and standard deviation of the extracted watermark should be equal to zero and one, respectively. This is because the embedded watermarks are of similar statistical characteristics. Note that, mean and standard deviation of the extracted watermark, as reported in Table 3, is very close to the actual theoretical value. Successful extraction of watermark help us to verify whether our attack is successful or not. Also, successful recovery of watermark

**Table 3.** Watermark Recovery

| Image | Correctly Recovered Bits | Mean | Standard Deviation | $\rho_w$ | $\rho_{avg}$ |
|---|---|---|---|---|---|
| Lena | 86.40% | 0.0000 | 1.0001 | 0.5347 | 0.0091 |
| Peppers | 83.31% | -0.0035 | 0.9999 | 0.5641 | 0.0054 |
| Boat | 83.23% | -0.0027 | 0.9999 | 0.6322 | 0.0089 |
| Elaine | 80.30% | -0.0039 | 0.9999 | 0.6365 | -0.0020 |
| F16 | 94.15% | 0.0017 | 0.9999 | 0.6764 | 0.0021 |

may have other serious implications as discussed in the earlier section. In Table 3, we present results related to recovery of watermark. In all the cases we can successfully recover at least 80% of the watermark bits. In Table 3, $\rho_w$ is the correlation between the watermarked image and extracted watermark. Similarly $\rho_{avg}$ is the correlation between the filtered image and extracted watermark. It is also evident that extracted watermark possesses high correlation with the watermarked images and insignificant correlation with the attacked (i.e., filtered) images. This gives confidence to the attacker in the watermark removal procedure because high correlation between the extracted watermark and watermarked image is interpreted as the successful extraction of watermark. Similarly, negligible correlation between the extracted watermark and attacked images indicates the absence of watermark in those images.

## 5 Conclusion

In this paper, we systematically study the Barni et al watermarking scheme from cryptanalytic point of view and introduce techniques to remove and recover the secret watermark signal. Experimental results confirm our claims regarding the recovery of secret watermark which is a very serious flaw of the scheme. It will be interesting to see whether it is further possible to recover the original image itself from the watermarked copy. Another possibility that should be investigated in greater detail is the possibility of mounting the ambiguity and copy attack on the scheme.

## References

1. R. J. Anderson and F. A. P. Petitcolas. On The Limits of Steganography. *IEEE Journal of Selected Areas in Communications. Special Issue on Copyright and Privacy Protection*, 16(4):474-481, May 1998.
2. M. Barni, F. Bartolini and A. Piva. Improved Wavelet-Based Watermarking Through Pixel-Wise Masking. *IEEE Transactions on Image Processing*, vol. 10, No. 5, May 2001.
3. J. Boeuf and J. P. Stern. An Analysis of One of the SDMI Candidates. *Information Hiding 2001*, pages 395-410, vol 2137 of Lecture Notes in Computer Science. Springer-Verlag, 2001.

4. S. Craver, N. Memon, B.-L. Yeo and M. M. Yeung. Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications. *IEEE Journal of Selected Areas in Communications*, vol. 16, no. 4, pp. 573586, May 1998, ISSN 0733-8716, special issue on copyright & privacy protection.
5. S. Craver, Bede Liu and W. Wolf. An Implementation of, and Attacks on, Zero-Knowledge Watermarking. In *Information Hiding 2004*, pages 1-12, vol 3200 of Lecture Notes in Computer Science. Springer-Verlag, 2004.
6. F. Cayre, C. Fontaine and T. Furon. Watermarking Attack: Security of WSS Techniques. In *IWDW 2004*, pages 171–183, vol 3304 of Lecture Notes in Computer Science. Springer-Verlag, 2005.
7. I. Daubechies. Ten Lectures on Wavelets. SIAM, 1992.
8. F. Ergun, J. Kilian and R. Kumar. A Note on the Limits of Collusion-Resistant Watermarks. In *Eurocrypt 1999*, Lecture Notes in Computer Science, Springer Verlag, 1999.
9. F. Hartung and M. Kutter. Multimedia Watermarking Techniques. *Proceedings of IEEE*, 87(7), July 1999.
10. T. Kalker, J. P. M. G. Linnartz and M. v. Dijk. Watermark Estimation through Detector Analysis. In *International Conference on Image Processing*, 1998.
11. A. Kerckhoffs. La Cryptographie Militaire. *Journal des Sciences Militaires*, $9^{th}$ series, IX(Jan 1883):pages 5-38, FEB(1883):pages 161-191.
12. D. Kirovski and F. A. P. Petitcolas. Replacement Attack on Arbitrary Watermarking Systems. In ACM workshop on *Digital Rights Management*, 2002.
13. M. Kutter, S. Voloshynovskiy and A. Herrigrl. The Watermark Copy Attack. In *Security and Watermarking of Multimedia Contents II*, SPIE–3971, pages 371-380, 2000.
14. A. S. Lewis and G. Knowles. Image Compression Using the 2-D Wavelet Transform. *IEEE Transactions on Image Processing*, vol 1, pp.244-250, Apr. 1992.
15. J. P. M. G. Linnartz and M. v. Dijk. Analysis of the Sensitivity Attack Against Electronic Watermarks in Images. In *Workshop on Information Hiding 1998*, Lecture Notes in Computer Science, volume 1525, pages 258–272, Springer-Verlag, 1998.
16. F. A. P. Petitcolas, R. J. Anderson, M. G. Kuhn and D. Aucsmith. Attacks on Copyright Marking Systems. In *2nd Workshop on Information Hiding*, Lecture Notes in Computer Science, volume 1525, pages 218–238, Springer Verlag, 1998.
17. F. A. P. Petitcolas and R. J. Anderson. Evaluation of Copyright Marking Systems. In *IEEE Multimedia Systems*, Florence, Italy, June 1999.
18. M. Vetterli and J. Kovacevic. *Wavelets and Subband Coding.* Prentice-Hall PTR, 1995.

# Completion Attacks and Weak Keys of Oleshchuk's Public Key Cryptosystem

Heiko Stamer

University of Kassel,
Department of Mathematics and Computer Science,
Heinrich-Plett-Straße 40, D-34132 Kassel, Germany
stamer@theory.informatik.uni-kassel.de

**Abstract.** This paper revisits a public key cryptosystem which is based on finite string-rewriting systems. We consider a new approach for cryptanalysis of such proposals—the so-called completion attack. If a particular kind of weak key is generated, then a passive adversary is able to retrieve secret messages with a significant probability. Our idea can be applied to other rewriting based cryptosystems as well. Finally we discuss issues concerning the practical usage and present some experimental results. The described vulnerabilities lead to the conclusion that at least the key generation of Oleshchuk's cryptosystem has to be revised.

**Keywords:** Cryptanalysis, completion attack, string-rewriting systems, Church-Rosser property, Knuth-Bendix completion, weak keys.

## 1 Introduction

The security of almost every public key cryptosystem relies on the intractability of only a few number-theoretic problems, e.g., factoring large integers or computing discrete logarithms in finite groups. Unfortunately, no proof of hardness (from a complexity theoretical point of view) is known for these assumptions. Therefore it sounds reasonable to look for other possible trapdoor functions in different areas of mathematics or computer science. Further there is the hope that such proposals [11, 13, 14, 9] will provide some kind of "provable security", because their underlying questions (e.g. the word problem for finitely presented groups) are undecidable in general. Beside other difficulties a primary problem in the design of such cryptosystems remains: The gap between the average and the worst case hardness of the instances, and hence the possibility of weak keys.

Vladimir A. Oleshchuk [1] proposed a public key cryptosystem that relies on the undecidability of the word problem in semigroups. A basic ingredient of his approach are string-rewriting systems. Each system is represented by a rule set containing ordered pairs of strings over a finite alphabet. Bidirectional rewriting on a string is performed through replacing (non-deterministically chosen) occurrences of the left-hand side by the right-hand side of a rule or vice versa. This operation induces an equivalence relation and we say that two strings are

© Springer-Verlag Berlin Heidelberg 2005

congruent, if they can be rewritten to each other in finitely many steps. The uniform word problem is the question of whether two given strings are congruent modulo a given rewriting system. This problem is undecidable in general, i.e. there exists no algorithm which terminates for all instances with the correct answer. However, if the set of rules is finite and the string rewriting system has the Church-Rosser property, then the word problem can be solved in polynomial time. This fact has been used by Oleshchuk [1, 2] to construct a trapdoor function and later the so-called Church-Rosser codes.

We consider a quite straightforward technique to attack such kind of cryptosystems. The so-called completion attack employs the well-known Knuth-Bendix algorithm on an improper chosen public key and constructs an equivalent string-rewriting system which has the Church-Rosser property. We show that it is possible to recover secret messages with significant probability, if the previous step was successful. Due to the undecidable termination of the completion procedures the (in)security of Oleshchuk's cryptosystem with respect to our attack remains somewhat unsettled.

The paper is organized as follows: The next section provides some notations and preliminaries. Then we briefly repeat the definition of Oleshchuk's public key cryptosystem and present an obvious variant. The fourth section is devoted to ideas for cryptanalysis, and in particular to the completion attack. Finally we consider practical issues and discuss experimental results with the attack.

## 2 Preliminaries

Let $\Sigma$ be a finite alphabet. $\Sigma^*$ denotes the set of all strings over this alphabet including the empty word $\epsilon$. The concatenation of two strings $x$ and $y$ is simply written as $xy$. Further, $|x|$ denotes the *length* of a string $x$, where $|\epsilon| = 0$, $|a| = 1$ for $a \in \Sigma$, and $|xa| = |x| + 1$ for $x \in \Sigma^*, a \in \Sigma$. If $A, B \subseteq \Sigma^*$ are languages, then their product is defined to be $AB = \{xy \mid x \in A, y \in B\}$.

A *string-rewriting system* $R$ on $\Sigma$ is a subset of $\Sigma^* \times \Sigma^*$. Each pair $(\ell, r) \in R$ is called *rewrite rule*. The word $\ell \in \Sigma^*$ is the left-hand side and the word $r \in \Sigma^*$ the right-hand side of such a rule. Here we will only be dealing with finite string-rewriting systems, i.e. $R$ is finite and its size is given by $||R|| = \sum_{(\ell,r)\in R} |\ell| + |r|$. Each string-rewriting system induces a *reduction relation* $\rightarrow_R^*$ on $\Sigma^*$, which is the reflexive transitive closure of the single-step reduction relation $\rightarrow_R = \{(x\ell y, xry) \mid x, y \in \Sigma^* \text{ and } (\ell, r) \in R\}$. If there is no $v \in \Sigma^*$ such that $u \rightarrow_R v$ holds, then the string $u$ is called *irreducible modulo* $R$. We denote the set of all irreducible words modulo $R$ by $\mathrm{IRR}(R)$. For finite string-rewriting systems $\mathrm{IRR}(R)$ is regular, i.e. a finite-state acceptor recognizing $\mathrm{IRR}(R)$ can be effectively constructed from the rules of $R$. The reflexive, symmetric, and transitive closure of $\rightarrow_R$ is the *Thue congruence* $\leftrightarrow_R^*$. We define the *congruence class* of a word $u \in \Sigma^*$ as $[u]_R = \{v \in \Sigma^* \mid v \leftrightarrow_R^* u\}$. This notation is expandable to $A \subseteq \Sigma^*$ by $[A]_R = \{v \in \Sigma^* \mid \exists u \in A : v \leftrightarrow_R^* u\}$.

An arbitrary string-rewriting system $R$ is called

– *noetherian* if there exists no infinite sequence of reductions w.r.t. $R$,
– *confluent* if, for all $u, v, w \in \Sigma^*$, $u \rightarrow_R^* v$ and $u \rightarrow_R^* w$ imply that $v$ and $w$ have a common descendant w.r.t. $R$ (i.e. $\exists z \in \Sigma^* \ : \ v \rightarrow_R^* z$ and $w \rightarrow_R^* z$),
– *convergent* if $R$ is noetherian and confluent,
– *length-reducing* if $|\ell| > |r|$ holds for each rewrite rule $(\ell, r) \in R$.

A string-rewriting system $R$ is *Church-Rosser* (i.e. $R$ has the Church-Rosser property) if, for all $x, y \in \Sigma^*$ with $x \leftrightarrow_R^* y$, there exists a word $z \in \Sigma^*$ such that $x \rightarrow_R^* z$ and $y \rightarrow_R^* z$. Hence, $R$ is Church-Rosser if and only if $R$ is confluent [3]. For finite length-reducing systems this property is decidable in polynomial time [7]. If a finite length-reducing system $R$ is Church-Rosser, then each congruence class has a unique irreducible element (modulo $R$) called *normal form.* Thus the corresponding word problem

**Instance:** Two arbitrary strings $x, y \in \Sigma^*$.
**Question:** Are $x$ and $y$ congruent modulo $R$, i.e. does $x \leftrightarrow_R^* y$ holds?

can be solved in linear time by comparing the normal forms [4].

Let $R_1$ and $R_2$ be string-rewriting systems on $\Sigma$. $R_1$ *refines* $R_2$, if for all $u, v \in \Sigma^*$ the congruence $u \leftrightarrow_{R_1}^* v$ implies $u \leftrightarrow_{R_2}^* v$. If $R_1$ refines $R_2$ and $R_2$ refines $R_1$, then they generate the same Thue congruence and are called *equivalent.* It is straightforward that $R_1$ refines $R_2$, if and only if the congruence $\ell \leftrightarrow_{R_2}^* r$ holds for each rewrite rule $(\ell, r) \in R_1$.

Let $>$ be a strict partial ordering on $\Sigma^*$. This ordering is called *admissible* if $u > v$ implies that $xuy > xvy$ holds for all $x, y \in \Sigma^*$, and it is called *well-founded* if there is no infinite strictly descending sequence $u_1 > \cdots > u_i > u_{i+1} > \cdots$. A string-rewriting system $R$ on $\Sigma$ is *compatible* with a given ordering $>$, if $\ell > r$ holds for each rewrite rule $(\ell, r) \in R$.

A nonempty set $C \subseteq \Sigma^*$ is called a *code*, if for all words $u_{i_1}, \ldots, u_{i_n} \in C$, $u_{j_1}, \ldots, u_{j_m} \in C$, the equality of $u_{i_1} \cdots u_{i_n} = u_{j_1} \cdots u_{j_m}$ implies $u_{i_1} = u_{j_1}$. By induction we deduce $n = m$ and $u_{i_k} = u_{j_k}$ for all $1 \leq k \leq n$. If $C$ is a code then any word from $C^*$ has a unique factorization over $C$.

## 3 Oleshchuk's Public Key Cryptosystem

We briefly repeat the original definition of Oleshchuk's public key cryptosystem [1]. Afterwards a subsection appends a necessary requirement for unique decryption, which was established later in [2]. Finally we slightly vary the discussed cryptosystem by introducing an additional secret morphism.

Let $\Sigma$ be the plaintext alphabet of possible messages $\mathcal{M} = \{w \mid w \in \Sigma^*\}$. Without loss of generality, we consider only the binary case $\Sigma = \{x_0, x_1\}$. The ciphertext alphabet $\Delta$ should be larger than $\Sigma$, i.e. $|\Delta| > |\Sigma|$.

*Key Generation.* Let $T$ be a finite and length-reducing Church-Rosser string-rewriting system on $\Delta$. We choose $u_1, u_2, \ldots, u_t \in \mathrm{IRR}(T)$ such that, for all $i, j = 1, \ldots, t$, the word $u_i u_j$ is irreducible modulo $T$ and the set $\{u_1, u_2, \ldots, u_t\}$ is a code. Now let $R_0, R_1 \subset \{u_1, u_2, \ldots, u_t\}$ be two nonempty disjoint sets, i.e. $R_0 \cap R_1 = \emptyset$. Further let $L_0 \subseteq [R_0]_T$ and $L_1 \subseteq [R_1]_T$ be two nonempty regular languages. Note that by construction $L_0 \cap L_1 = \emptyset$ because $T$ is confluent and $R_0, R_1$ are disjoint. The next step of key generation picks a finite string-rewriting system $S$ on $\Delta$ such that $\ell \leftrightarrow_T^+ r$, for all rewrite rules $(\ell, r) \in S$, i.e. $S$ refines $T$. This property can be tested easily because $T$ is Church-Rosser and thus the corresponding word problem is decidable in linear time [4]. However, it is not clear how to construct $S$ efficiently without leaking information about $T$.

The finite string-rewriting system $S$ and the languages $L_0, L_1$ form the public key $K_{\text{pub}}$. The finite Church-Rosser system $T$ and the sets $R_0, R_1$ should be kept secret, because they represent the private key $K_{\text{sec}}$.

*Encryption.* The encryption of the letter $x_i$ is a random word $y \in [L_i]_S$. Therefore the non-deterministic function $\mathsf{Encrypt} : \mathcal{M} \to \mathcal{C}$ maps a possible message $m = x_{i_1} x_{i_2} \cdots x_{i_n}$, where $x_{i_k} \in \Sigma,\ k = 1, \ldots, n$, to a random ciphertext $c \in [L_{i_1} L_{i_2} \cdots L_{i_n}]_S$. In practice one will do the following two steps:

1. Encode the plaintext $m = x_{i_1} x_{i_2} \cdots x_{i_n}$ into $\hat{m} = \hat{x}_{i_1} \hat{x}_{i_2} \cdots \hat{x}_{i_n}$, where each string $\hat{x}_{i_k}$ is randomly chosen from the corresponding language $L_{i_k}$.
2. Rewrite the $\hat{m}$ randomly and uniformly according to the rules of $S$.

*Decryption.* For the decryption of a secret message $c \in \mathcal{C}$ one has to find a word $\hat{m} \in (L_0 \cup L_1)^*$ such that $c \leftrightarrow_S^* \hat{m}$ holds. In general the finite string-rewriting system $S$ may have an undecidable word problem [4] and even decidability does not guarantee that the problem is computationally feasible [6].

With the secret key $K_{\text{sec}} = (T, R_0, R_1)$ decryption becomes easy, because $T$ has the Church-Rosser property and thus there exists a uniquely defined word $\tilde{m} \in \text{IRR}(T)$ such that $c \to_T^* \tilde{m}$. This normal form can be found in linear time [4] and its factorization $\tilde{m} = u_{i_1} u_{i_2} \cdots u_{i_n}$ (where $u_{i_k} \in R_{i_k}$) obviously reveals the plaintext $m = x_{i_1} x_{i_2} \cdots x_{i_n}$ of the encrypted message.

## 3.1 Necessary Requirement for Unique Decryption

It was observed [2] that the condition $u_i u_j \in \text{IRR}(T)$ for all $i, j = 1, \ldots, t$ is not sufficient for a unique decoding. In fact one has to ensure (during the process of key generation) that $(R_0 \cup R_1)^* \subseteq \text{IRR}(T)$ holds. However, this generalization can be effectively tested, since both sides are regular languages.

Let $\psi_T = \max\{|\ell_1|, \ldots, |\ell_{|T|}|\}$ for all rewrite rules $(\ell_i, r_i) \in T$ and let $\psi_R = \max\{|u_1|, \ldots, |u_t|\}$ for all words $u_i \in (R_0 \cup R_1)$. Now it is sufficient to check whether the inclusion $(R_0 \cup R_1)^{\leq 2\max\{\psi_T, \psi_R\}} \subseteq \text{IRR}(T)$ holds. For example, this can be performed by generating all concatenations of length less than or equal to $2\max\{\psi_T, \psi_R\}$ and verifying the irreducibility for each of them.

## 3.2 Morphism Strengthened Variant

The following idea is based on a well-known technique [12, 9, 19] to hide the structure of the rewriting steps by an additional morphism. Let $\Gamma$ be a new ciphertext alphabet of much greater cardinality than $\Delta$ and let $g : \Gamma^* \to \Delta^*$ be a monoid homomorphism which meets the following conditions:

1. For each letter $\gamma \in \Gamma$ we either have $g(\gamma) = \epsilon$ or $g(\gamma) \in \Delta$. In the first case we will call $\gamma$ a *dummy symbol.*
2. At least for each letter $\delta \in \Delta$ there exists a $\gamma \in \Gamma$ such that $g(\gamma) = \delta$.

In the strengthened variant of Oleshchuk's cryptosystem such a morphism $g$ is part of the secret key, i.e. $K_{\text{sec}} = (T, R_0, R_1, g)$. The modified public key $K_{\text{pub}} = (\mathsf{S}, \mathsf{L}_0, \mathsf{L}_1)$ is obtained from the original $(S, L_0, L_1)$ as follows:

$$\mathsf{S} = \{(g^{-1}(\ell), g^{-1}(r)) \mid (\ell, r) \in S\} \cup D$$

The finite set $D$ contains *dummy rules* $(\ell, r)$ whose words $\ell, r \in \Gamma^*$ satisfy $g(\ell) = g(r) = \epsilon$. Obviously, the refinement property of $S$ remains valid under the images of $g$, i.e. the congruence $g(\ell) \leftrightarrow_T^* g(r)$ holds for all $(\ell, r) \in \mathsf{S}$.

The generation of $\mathsf{L}_0$ respectively $\mathsf{L}_1$ depends on their representation: If $L_i$ is a finite set one can easily compute $\mathsf{L}_i = \{g^{-1}(w) \mid w \in L_i\}$. Otherwise, if they are represented by a grammar $G_{L_i}$ one might apply $g^{-1}$ to each terminal symbol $\delta \in \Delta$ in all derivation rules of $G_{L_i}$.

Finally the decryption has to be changed slightly: In a new initial step the legal recipient of an encrypted message $c \in \mathcal{C}$ can apply the morphism $g$ in linear time. Afterwards he proceed as usual, i.e. reducing the result $g(c)$ modulo $T$.

## 4 Cryptanalysis

Like similar cryptosystems [15, 16] the discussed approach is vulnerable to particular cryptanalytic attacks, if weak keys are chosen during the key generation. Specifically, here we are concerned about "bad string-rewriting systems".

### 4.1 Completion Attack on $S$

Oleshchuk [1] already noticed that it is not necessary to find the exact secret system $T$ generated by the legal key owner. Any Church-Rosser system $T'$ where all of the conditions

1. $S$ refines $T'$
2. $[L_0]_{T'} \cap [L_1]_{T'} = \emptyset$
3. $([L_0]_{T'} \cup [L_1]_{T'}) \cap \text{IRR}(T')$ is a code
4. $([L_0]_{T'} \cup [L_1]_{T'})^* \subseteq \text{IRR}(T')$ (reformulated according to [2])

apply, can be used to decrypt messages. In general, there is no algorithm to decide whether a finite string-rewriting system is equivalent to any finite Church-Rosser system [5]. But a cryptanalyst can try techniques known as *completion procedures* to construct such a convergent system $T'$ from $S$.

The Knuth-Bendix completion [17] takes as input a finite string-rewriting system $S$ on $\Delta$ and an admissible well-founded strict partial ordering $>$ on $\Delta^*$. Based on this ordering the system $S$ is turned into an equivalent system $T'$ that is compatible with $>$ by orienting each rule with respect to this ordering. Thus $T'$ will be noetherian. Then the critical pairs of $T'$ are computed, and for each critical pair that does not resolve a new rule is introduced. Unfortunately, each new rule can lead to new unresolvable critical pairs, and hence, this process may not terminate. Moreover, the termination highly depends on the used ordering $>$. A detailed description of the Knuth-Bendix completion is omitted here. Interested readers are referred to the existing literature [10, 17, 3].

Let $(S, L_0, L_1)$ be a *KBC-weak* public key, i.e. the Knuth-Bendix completion procedure terminates and outputs a length-reducing system $T'$ which is equivalent to $S$. Thus a cryptanalyst can reduce an observed ciphertext $c \in \mathcal{C}$ to a normal form $\tilde{m}' \in \mathrm{IRR}(T')$ in linear time [4], i.e. $c \rightarrow^*_{T'} \tilde{m}'$. Further let us assume that the regular languages $L_0$ and $L_1$ are finite. By reducing all words of the $L_i$'s to their normal forms modulo $T'$ the adversary will get the finite languages $R'_i = \{u' \in \mathrm{IRR}(T') \mid \exists \hat{x} \in L_i : \hat{x} \rightarrow^*_{T'} u'\}$ efficiently. Moreover, these sets are of the same size as the $L_i$'s.

First, consider the restricted case of one-bit messages ($n = 1$): We can reduce the hard problem of deciding whether $c \in [L_0]_S$ or $c \in [L_1]_S$ to a much easier question modulo $T'$.

**Lemma 1 (One-bit Messages).** *$c \in [L_i]_S$ if and only if $\tilde{m}' \in R'_i$, for $i = 0, 1$.*

*Proof.* We prove both cases $i = 0$ and $i = 1$ in common:

$\Rightarrow$ For $c \in [L_i]_S$ there exists a word $\hat{x} \in L_i$ such that $c \leftrightarrow^*_{T'} \hat{x} \leftrightarrow^*_{T'} \tilde{m}'$ since $S$ and $T'$ are equivalent. Further, by construction each $\hat{x}$ has an irreducible word $u' \in R'_i$ modulo $T'$ where $\hat{x} \rightarrow^*_{T'} u'$. Finally $u' = \tilde{m}'$, because $T'$ is confluent and consequently the normal forms are unique.

$$\begin{tikzcd}
c \arrow[rrrr, leftrightarrow, "*", "{S,T'}"'] \arrow[dr, "*", "T'"'] \arrow[dd, "*"', "T"] & & & & \hat{x} \arrow[dl, "*"', "T'"] \arrow[dd, "*"', "T"] \\
& \tilde{m}' \arrow[dl, "*"', "T"] & = & u' \arrow[dr, "*", "T"'] & \\
\tilde{m} & & = & & u
\end{tikzcd}$$

$\Leftarrow$ The other direction follows by similar arguments. □

Now we turn to longer messages, i.e. $n > 1$. Here appears the problem that not necessarily $(R'_0 \cup R'_1)^* \subseteq \mathrm{IRR}(T')$ and thus the decoding may be ambiguous.

Let $\bar{L}'_0 \subseteq L_0$ respectively $\bar{L}'_1 \subseteq L_1$ be the finite set of all strings $\hat{x}_{i_j} \in L_{i_j}$ used during the encryption of a fixed ciphertext $c$, i.e. $c \in [\bar{L}'_{i_1} \bar{L}'_{i_2} \cdots \bar{L}'_{i_n}]_S$. Further denote by $\bar{R}'_i = \{u' \in \mathrm{IRR}(T') \mid \exists \hat{x} \in \bar{L}'_i : \hat{x} \rightarrow^*_{T'} u'\}$ the corresponding normal forms modulo $T'$. Under three additional conditions, namely that $T'$ is code preserving w.r.t. $\bar{L}'_i$, we can generalize the previous Lemma 1.

**Lemma 2 (Arbitrary Messages).** *If $[\bar{L}'_0]_{T'} \cap [\bar{L}'_1]_{T'} = \emptyset$, $([\bar{L}'_0]_{T'} \cup [\bar{L}'_1]_{T'})$ is a code, and $([\bar{L}'_0]_{T'} \cup [\bar{L}'_1]_{T'})^* \subseteq \mathrm{IRR}(T')$, then*

1. $(\bar{R}'_0 \cup \bar{R}'_1)^* \subseteq \mathrm{IRR}(T')$ *and* $(\bar{R}'_0 \cup \bar{R}'_1)$ *is a code, and*
2. $c \in [\bar{L}'_{i_1}\bar{L}'_{i_2}\cdots\bar{L}'_{i_n}]_S$ *if and only if* $\tilde{m}' \in \bar{R}'_{i_1}\bar{R}'_{i_2}\cdots\bar{R}'_{i_n}$. *(for all* $n \in \mathbb{N}$*)*

*Proof.* The first claim is obvious, because $\bar{R}'_0 \subseteq [\bar{L}'_0]_{T'}$ and $\bar{R}'_1 \subseteq [\bar{L}'_1]_{T'}$. The remaining part follows by an inductive application of Lemma 1 using the fact, that $(\bar{R}'_0 \cup \bar{R}'_1)$ is a code and that all possible concatenations of the included words are irreducible modulo $T'$. □

**Proposition 1.** *The congruence classes* $[L_0]_{T'}$ *and* $[L_1]_{T'}$ *are disjoint.*

*Proof.* Assume the contrary. $T'$ and $S$ are equivalent, i.e. $\leftrightarrow^*_S = \leftrightarrow^*_{T'}$, and thus for each $w \in ([L_0]_{T'} \cap [L_1]_{T'})$ the congruence $\hat{w}_0 \leftrightarrow^*_S w \leftrightarrow^*_S \hat{w}_1$ holds for at least two words $\hat{w}_0 \in L_0, \hat{w}_1 \in L_1$. However, since $S$ refines $T$ and the string-rewriting system $T$ is Church-Rosser there exists a unique normal form $u \in \mathrm{IRR}(T)$ such that $\hat{w}_0 \rightarrow^*_T u$ and $\hat{w}_1 \rightarrow^*_T u$. Hence it either violates the condition $L_0 \cap L_1 = \emptyset$ or contradicts the unique decryption of messages. □

**Theorem 1.** *If the public key is KBC-weak and if* $L_0, L_1$ *are finite sets, then a passive adversary can retrieve the plaintext of an encrypted message with significant probability in linear time.*

*Proof.* The adversary applies Lemma 2. In the case of a KBC-weak key (completion procedure terminates) the first condition is always fulfilled, because $[L_0]_{T'} \supseteq [\bar{L}'_0]_{T'}$ and $[L_1]_{T'} \supseteq [\bar{L}'_1]_{T'}$ are already disjoint (see Proposition 1). Moreover, the remaining conditions are often satisfied for short messages or sparse sets $\bar{L}'_0, \bar{L}'_1$ (including the trivial case of one-bit messages). As $(\bar{R}'_0 \cup \bar{R}'_1)$ is a code the adversary can easily determine the corresponding factorization of $\tilde{m}'$ and hence the plaintext. Further, if one of the necessary conditions does not hold, a passive adversary is probably still able to retrieve partial information about the corresponding plaintext $m$. Note that for a successful attack it may be sufficient to distinguish some subwords of $\tilde{m}'$ between $R'_0$ and $R'_1$. □

*Example 1 (Knuth-Bendix Completion Attack).*

$$T = \{(cb, c), (aa, a), (ab, a)\},\ S = \{(ab, aab), (cba, ca), (baa, ba)\}$$

$$R_0 = \{cacac\},\ R_1 = \{aca\},\ L_0 = \{caacbabc\},\ L_1 = \{abacbaab\}$$

On input of $S$ and $>_{\mathrm{llex}}$ (length-lexicographical ordering) the Knuth-Bendix completion terminates with the length-reducing Church-Rosser system

$$T' = \{(aab, ab), (cba, ca), (baa, ba), (caa, ca)\}.$$

By reducing $L_0, L_1$ modulo $T'$ we get $R'_0 = \{cacabc\}$ and $R'_1 = \{abacab\}$.

$$caacbabc \overset{(4)}{\rightarrow}_{T'} cacbabc \overset{(2)}{\rightarrow}_{T'} cacabc$$
$$abacbaab \overset{(2)}{\rightarrow}_{T'} abacaab \overset{(4)}{\rightarrow}_{T'} abacab$$

Now suppose that the cryptanalyst observes the ciphertext $c = cbaacabc$, which can be reduced in two steps to $\tilde{m}' = cacabc \in [R_0']_{T'}$.

$$c = cbaacabc \overset{(2)}{\rightarrow}_{T'} caacabc \overset{(4)}{\rightarrow}_{T'} cacabc = \tilde{m}' \in [R_0']_{T'}$$

Hence the corresponding plaintext is the single letter $x_0$.

As consequence we have to avoid the KBC-weakness during the key generation. Unfortunately, such a property is undecidable [5] in general. Hence the security of Oleshchuk's cryptosystem with respect to completion attacks remains undecidable. Furthermore, even the morphism strengthened variant does not really protect against this kind of attack, because it leaves the distinguish ability between $R_0'$ and $R_1'$ unchanged. In such a case, however, the effort of the adversary increases since he has to deal with the included dummy rules.

The straightforward idea to prevent our attack might be the usage of infinite regular languages $L_i$, e.g., represented by grammars in the public key. Nonetheless, the $\bar{L}_i'$'s are still finite, because the sender can only apply finitely many rewrite steps during the encryption. But the point is that hopefully these sets are large enough such that the computation of the distinguishing normal forms $\bar{R}_i'$ is infeasible for a bounded adversary.

We stress that completion attacks are applicable to other rewriting based schemes, e.g., the public key cryptosystem of Samuel et al. [19]. In their proposal (Church-Rosser) tree replacement systems [20, 21] are used for building the trapdoor. Thus it seems to be very likely to mount successful *tree completion procedures* and retrieve parts of secret messages, if KBC-weak keys are chosen as well.

## 4.2 Guessing $T$ by Pre- or Suffix Properties of $S$

Other properties of weak keys are exploitable: A pitfall stems from the fact that $S$ refines $T$. Of course, if $S = \{(x\ell y, xry) \mid (\ell, r) \in T,\ x, y \in \Delta^*\}$ the congruence $x\ell y \leftrightarrow_T^* xry$ holds for each rewrite rule in $S$.

A cryptanalyst can guess the secret Church-Rosser system $T$, if $S$ was simply chosen like above, i.e. for some $(\ell, r) \in T$ with $|\ell| > |r|$ there exists a prefix $z \in \Delta^*$ such that $(z\ell, zr) \in S$ or $(zr, z\ell) \in S$.

## 4.3 Further Ciphertext-Only Attacks

If the string-rewriting system $S$ and the sets $L_0, L_1$ are not carefully generated, then (analogously to [16]) additional information about the corresponding plaintext $m \in \mathcal{M}$ is leaked by a given ciphertext $c \in \mathcal{C}$.

For example, assume that a letter of the ciphertext alphabet $\Delta$ appears only in words either from $L_0$ or $L_1$. Assume further that this relation is preserved by the rules of $S$. Now the adversary counts the occurrences of such a letter in $c$ and hence obtains information about the number and positions of the $x_0$'s resp. $x_1$'s in the plaintext $m$. One can generalize this kind of attack to other "measures", e.g., if $S$ preserves some unique subword or a characteristic length. Again, the point is that a cryptanalyst only has to distinguish between $L_0$ and $L_1$.

## 5 Practical Issues

This section sketches some questions that arose during our implementation of Oleshchuk's cryptosystem. The programming was done as *proof of concept* in approximately 1 600 lines of C++ code. Thus features and documentation are very limited: The program constructs a random key pair $(K_{\text{pub}} = (S, L_0, L_1), K_{\text{sec}} = (T, R_0, R_1))$ and performs some simple encryption/ decryption operations on one-bit and larger messages. Finally, if possible, the completion attack (see Section 4.1) is mounted. We provide the source code [23] under the *GNU General Public License*, if the reader would like to verify the results of the next section or if he want to use parts of our work in further cryptanalysis.

*Choosing "Cryptographically Good" Parameter Settings.* This seems to be a serious question since many possible parameters may have influence on the security of the entire cryptosystem, e.g., the size of the ciphertext alphabet $\Delta$, the sizes of the string-rewriting systems $T$ and $S$, the sizes of the sets $R_i$, and, if finite sets $L_i$ are used, the number of applied rules during their generation. In the original paper there were only few hints how to choose these parameters appropriately.

Concerning the ciphertext alphabet $\Delta$ we can say that in the unary case the word problem becomes decidable for finite string-rewriting systems [3]. On the other hand, if we consider a bounded number of rewrite rules over an arbitrary finite alphabet, then there exists a string-rewriting system with only three rules which has an undecidable word problem [18]. It is a well-known open question [22] of whether or not this problem becomes decidable if we consider only one-rule rewriting systems. Hence $S$ should have at least three rules.

*Generating a String-rewriting System $S$ that Refines $T$.* Up to now we don't have any other method than to randomly guess $S$ and check whether $\ell \leftrightarrow_T^+ r$ holds for all rewrite rules $(\ell, r) \in S$. Each obvious strategy to perform this in a more efficient way will probably introduce new vulnerabilities, e.g., by the inherent structure of the derived public key space. Beside the other concerns, this issue is the most crucial problem since it makes the generation algorithm really impractical for reasonable key sizes.

*Encrypting Messages.* Here we have to ensure that a cryptanalyst cannot handle the word problem by a brute-force search in the Thue congruence. Therefore the number of nodes in the derivation tree of $c \leftrightarrow_S^* \hat{m}$ should grow exponentially in the number of performed rewrite steps during the encryption.

*Ciphertext Blow-up.* From a practical point of view the enormous ciphertext blow-up of the encryption function is undesirable. Of course, we can try to balance the application of length-increasing and length-reducing rewrite rules, but such a strategy could be another starting point for successful attacks.

## 6 Experimental Results

Now we want to estimate how likely the proposed completion attack works in practice. We stress again that in general it is undecidable whether a public key is

**Table 1.** Experimental results with the completion attack (three independent runs)

| | $\vert S\vert = 3$ | $\vert S\vert = 4$ | $\vert S\vert = 5$ |
|---|---|---|---|
| $\vert R_0 \cup R_1\vert = 5$ | $\underbrace{23:12, 24:20, 23:14}_{23.3\% \text{ vs. } 15.3\%}$ | $\underbrace{13:4, 8:7, 15:12}_{12\% \text{ vs. } 7.6\%}$ | $\underbrace{5:3, 4:1, 4:3}_{4.3\% \text{ vs. } 2.3\%}$ |
| $\vert R_0 \cup R_1\vert = 10$ | $\underbrace{17:13, 19:10, 23:11}_{19.6\% \text{ vs. } 11.3\%}$ | $\underbrace{13:8, 9:5, 11:2}_{11\% \text{ vs. } 5\%}$ | $\underbrace{4:2, 5:1, 3:2}_{4\% \text{ vs. } 1.6\%}$ |
| $\vert R_0 \cup R_1\vert = 25$ | $\underbrace{22:10, 23:13, 26:10}_{23.6\% \text{ vs. } 11\%}$ | $\underbrace{12:4, 15:8, 9:6}_{12\% \text{ vs. } 6\%}$ | $\underbrace{3:1, 4:1, 5:2}_{4\% \text{ vs. } 1.3\%}$ |

KBC-weak. Thus our experimental results can only give an very rough approximation. Further there are practical limitations, in particular the generation of $S$, while performing such an analysis. Hence we restrict our investigation to "small keys" which may have less cryptographic relevance. Table 1 shows the number of successful Knuth-Bendix completions on $S$ (using the length-lexicographical ordering) versus the number of successful completion attacks (Theorem 1) itself. The attack was performed on a fixed 112-bit message and we have count only a proper decryption as successful. The completion procedure was aborted either after three iterations or if more than 250 critical pairs occurred. For each single run our program [23] has generated (randomly and uniformly) one hundred instances of Oleshchuk's cryptosystem with $|\varDelta| = 3$, $|T| = 3$, and $6 \leq ||T|| \leq 12$. These parameters have been chosen to achieve a reasonable time-frame for the experiment. The above results show that our attack works well in the small parameter scenario, if the Knuth-Bendix completion itself is successful.

## 7 Conclusion

We have shown that the discussed cryptosystem is principally vulnerable to the presented completion attack. Even the strengthened variant does not hide the distinguish ability of the encoding languages properly. Thus, at least in our opinion, it does not offer an acceptable level of cryptographic security. On the other hand, the KBC-weakness of public keys is undecidable in general. Hence it is very hard to estimate how likely our attack works in practice. We leave it as an open question whether Oleshchuk's cryptosystem can be repaired to withstand the proposed attacks. However, the practical issues have shown clearly that all further effort can be only of theoretical interest.

*Acknowledgment.* The author thanks Friedrich Otto, Tomasz Jurdziński, and Andreas Conz for the interesting discussions regarding this work. Additionally, he thanks the anonymous referees for their helpful comments.

## References

1. Vladimir A. Oleshchuk. On Public-Key Cryptosystem Based on Church-Rosser String-Rewriting Systems. Proceedings of COCOON'95, Lecture Notes in Computer Science 959, pp. 264–269, 1995.
2. Vladimir A. Oleshchuk. Church-Rosser Codes. Proceedings of 5th IMA Conference, Lecture Notes in Computer Science 1025, pp. 199–204, 1996.
3. Ronald V. Book and Friedrich Otto. String-Rewriting Systems. Texts and Monographs in Computer Science, Springer, New-York, 1993.
4. Ronald V. Book. Confluent and other types of Thue systems. *Journal of the ACM* 29:171–183, 1982.
5. Colm O'Dunlaing. Undecidable questions related to Church-Rosser Thue systems. *Theoretical Computer Science* 23:339–345, 1983.
6. Günther Bauer and Friedrich Otto. Finite Complete Rewriting Systems and the Complexity of the Word Problem. *Acta Informatica* 21:521–540, 1984.
7. Deepak Kapur, Mukkai S. Krishnamoorthy, Robert McNaughton, and Paliath Narendran. An $O(|T|^3)$ algorithm for testing the Church-Rosser property of Thue systems. *Theoretical Computer Science* 35:109–114, 1985.
8. Robert McNaughton, Paliath Narendran, and Friedrich Otto. Church-Rosser Thue systems and formal languages. *Journal of the ACM* 35:324–344, 1988.
9. Valtteri Niemi. Cryptology: Language-Theoretic Aspects. In G. Rozenberg and A. Salomaa (Eds.): Handbook of Formal Languages, Springer, Berlin, 1997.
10. Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite Systems. In J. van Leeuwen (Ed.): Handbook of Theoretical Computer Science, Volume B, Elsevier Science Publishers, Amsterdam, 1990.
11. Neal R. Wagner and Marianne R. Magyarik. A Public-Key Cryptosystem Based on the Word Problem. In *Advances in Cryptology: Proceedings of CRYPTO'84*, Lecture Notes in Computer Science 196, pp. 19–36, 1985.
12. Arto Salomaa. On a public-key cryptosystem based on language theory. *Computers and Security* 7:83–87, 1988.
13. Rani Siromoney and Lisa Mathew. A Public Key Cryptosystem Based on Lyndon Words. *Information Processing Letters* 35:33–36, 1990.
14. Akihiro Yamamura. Public-Key Cryptosystems Using the Modular Group. 1st International Workshop on Practice and Theory in Public Key Cryptography (PKC'98), Lecture Notes in Computer Science 1431, pp. 203–216, 1998.
15. Maria I.G. Vasco and Rainer Steinwandt. Pitfalls in public key cryptosystems based on free partially commutative monoids and groups. Cryptology ePrint Archive: Report 2004/012, 2004.
16. David P. Garcia and Maria G. Vasco. Attacking a Public Key Cryptosystem Based on Tree Replacement. Cryptology ePrint Archive: Report 2004/098, 2004.
17. Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech (Ed.): Computational Problems in Abstract Algebra, pp. 263–297, Pergamon Press, New-York, 1970.
18. Yuri Matiyasevich and Geraud Sénizergues. Decision problems for semi-Thue systems with a few rules. Proceedings of the 11th IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, pp. 523–531, 1996.
19. S.C. Samuel, D.G. Thomas, P.J. Abisha, K.G. Subramanian. Tree Replacement and Public Key Cryptosystem. In A. Menezes, P. Sarkar (Eds.): INDOCRYPT 2002, Lecture Notes in Computer Science 2551, pp. 71–78, 2002.

20. Barry K. Rosen. Tree-Manipulating Systems and Church-Rosser Theorems. *Journal of the ACM* 20:160–187, 1973.
21. Jean H. Gallier and Ronald V. Book. Reductions in Tree Replacement Systems. *Theoretical Computer Science* 37:123–150, 1985.
22. Nachum Dershowitz and Ralf Treinen. The RTA list of open problems. `http://www.lsv.ens-cachan.fr/rtaloop/`
23. Heiko Stamer. Implementation of Oleshchuk's Public Key Cryptosystem. `http://www.theory.informatik.uni-kassel.de/~stamer/OlkPK2.tar.gz`

# An Optimal Subset Cover for Broadcast Encryption

Sarang Aravamuthan[1] and Sachin Lodha[2]

[1] Advanced Technology Center, Hyderabad, India
[2] Tata Research Development and Design Center, Pune, India*
{a.sarangarajan, sachin.lodha}@tcs.com

**Abstract.** In broadcast networks, it is often required to encrypt data so that only a privileged set of users with access to the session key can access the data. The standard technique of transferring the session key to each user individually does not scale with the number of users typically found on a network such as cable. This method is not only time-wise inefficient, but also incurs high communication cost. To counter this, a number of approaches have been proposed in the literature that include methods based on secret sharing schemes, construction of subset covers using combinatorial designs, *etc.*

In this paper, we propose and study two natural combinatorial optimization problems related to the subset cover framework for broadcast encryption. Here our objective is to minimize the communication cost given certain security and storage related constraints. We first derive lower bounds for the optimal communication cost for both problems. Then we propose the Partition-and-Power (PaP) subset cover scheme and show that it can provide a secure broadcast encryption with the communication costs matching those lower bounds. We illustrate the merits of the PaP scheme through a few examples and compare it with some of the prevailing subset cover schemes.

**Keywords:** Broadcast Networks, Encryption, Key Distribution, Subset Cover.

## 1 Introduction

In a broadcast network, a transmitter broadcasts signals that are intercepted by a set of receivers by tuning to the appropriate frequency. The medium of transmission may be air as in Radio or Television or dedicated transmission lines (Cable). The signals are usually sent in the clear so that any receiver within a certain radius of the transmitter or, in the case of cable, connected to the cable network can intercept the signal.

With the introduction of pay-per-view and video on demand, it is also required to encrypt some of the signals so that only authorized users can receive

* Both ATC and TRDDC are research units of Tata Consultancy Services Limited.

© Springer-Verlag Berlin Heidelberg 2005

these signals. This requires the secure transmission of the decryption key (also called the *session key*) to each authorized recipient allowing them to decipher further signals with that key.

This is possible provided each recipient has its own deciphering key for receiving secure transmissions (such as the session key) from the broadcaster. The deciphering keys can be distributed to the recipients through devices that have the key and decryption algorithm embedded in them.

Thus, to send an encrypted message such as a video stream to $k$ users, a broadcaster would first have to encrypt the session key with each user's enciphering key and transmit $k$ blocks of encrypted text. However, this scheme becomes impractical when the user base is large. A cable operator would typically be providing services to a million users. Just transferring a session key may involve encrypting and transmitting a few hundred thousand blocks of ciphertext. Moreover it is not a *true* broadcast scheme (see Berkovits [2]), since the transmitter is using the broadcast message to reach each recipient in *series.*

In *broadcast encryption*, each receiver is initially configured with a set of keys. The keys are not updated, i.e. the receivers are *stateless.* A session is a time interval when a message is broadcast to a privileged or non-revoked subset of receivers. A broadcast consists of a message encrypted with a session key along with a header that contains information for the non-revoked users to recover the session key. The efficacy of a broadcast encryption scheme is determined by three parameters

– the *processing time* or the number of operations at the receiver to recover the session key
– the *storage* at receiver
– the *transmission overhead* or the size of the message header.

The goal of broadcast encryption is to minimize the transmission overhead while keeping the storage size at the receiver and processing time as small as possible.

## 1.1 Related Work

Chiou and Chen were the first to propose a broadcast technique in [5] using a locking mechanism. This approach requires a large transmission overhead but less storage at the receivers.

In [2], a broadcast scheme was proposed by Berkovits that is based on the "*k out of n*" secret sharing scheme of Shamir [11]. The scheme uses polynomial interpolation and related vector formulation methods, but requires frequent update of the secret shares.

Fiat and Naor [7] consider $k$-resilient broadcast schemes where coalitions of $k$ users not in the privileged set cannot recover the session key. When $k$ is of the order of the number of receivers, then the transmission overhead as well as the processing time becomes large.

Stinson [12] discusses the general approach to fully resilient broadcast encryption using secret sharing schemes and key predistribution schemes. It is illustrated by using balanced incomplete block designs (BIBDs) together with threshold schemes to construct new broadcast encryption schemes. The resulting schemes again require a large transmission length.

As the reader can already sense from the description of different schemes, there is a trade-off between communication and storage. Blundo et al study it in information theoretic setting for unconditionally secure broadcast encryption schemes in [3, 4]. Luby and Staddon [9] study the inherent trade-off between the number of establishment keys held by each user and the number of transmissions needed to establish a new broadcast key. Unlike [4], their model is combinatorial in nature. They show essentially tight exponential lower bounds on the number of transmissions needed to establish a new broadcast key given an upper bound on the number of establishment keys held by each user.

In order to break away from such theoretical bounds [3, 4, 9], Abdalla, Shavitt and Wool consider a model in [1] that allows a controlled number of users outside the privileged set to occasionally receive the broadcast. They introduce $f$-redundant establishment key allocations which guarantee that the total number of recipients is no more than $f$ times the number of intended receivers.

In [10], Naor, Naor and Lotspiech introduce the *subset difference method* to cover the set of non-revoked users by a collection of disjoint subsets. Each non-revoked user uses the key corresponding to his subset to recover the session key. If $r$ is the number of revoked users and $n$ the total number of users, their scheme has a transmission overhead of $2r - 1$ messages, each receiver stores $O(\log^2 n)$ keys and the processing time at a receiver is $O(\log n)$ operations. These bounds are improved by Halevy and Shamir [8] using a *Layered Subset Difference* (LSD) scheme. Specifically, they reduce the storage size to $O(\log^{3/2} n)$ while the processing time remains the same. However, the transmission overhead increases to $4r$ messages.

## 1.2 Our Contribution

In this paper, we consider the traditional broadcast encryption problem as stated in the introduction under the subset cover framework. Like the authors of [4] and [9], our objective is also to study the communication-storage trade-off.

In [4], each receiver is given some secret information and the users use the information to compute common keys via a key pre-distribution scheme. The efficiency of the systems in [4] is measured by considering the amount of secret information held by each user as compared to the information content of the broadcast made to establish the session key; and the size of the broadcast as compared to its information content. In this paper, like [9], we assume that the users are actually given the keys (as in an integrated circuit (IC) card) rather than the information to compute them, and the communication is measured in terms of the number of keys needed to establish the session key. These are both important practical parameters. In an implementation of a broadcast encryption system, a receiver's keys may be contained in an IC card with only limited memory, and the broadcaster may want to limit the number of transmissions due to cost-efficiency concerns. Under these assumptions, we prove that for a given upper bound on the number of keys held by each user, there is an inherent lower bound on the transmission overhead needed to recover the session key. Because of the differences between our measurements of efficiency and those in [4], the optimal systems in [4] are not optimal in our model. For example, they present an optimal scheme using resolvable designs, with $\binom{n/2}{n/4} = \Omega(2^{n/2}/\sqrt{n})$ transmissions to broadcast to a privileged set of size $n/2$, out of universe of $n$ receivers, that requires each user to generate $\binom{n-1}{\frac{n}{4}-1}$ keys. In this paper, we present a system in which each receiver has $2^g$ keys and only $O(n/g)$ transmissions are needed where $g$ is a system determined constant dependent on the storage constraint and security consideration.

In [9], the authors consider the privileged sets of only a certain fixed size. In our model, any subset out of universe of $n$ receivers could be a privilged set. Thus our model is more general that that described in [9]. It is important to note that [9] addresses the issue of number of transmissions, but not the issue of the size of transmission. Thus, one of their three constructions requires $O(n)$ binary strings of the same size as the session key be sent with each transmission. Our construction is efficient in this respect, since it requires only that a binary string of the same size as the broadcast key be sent with each transmission. Another difference between [4, 9] and this paper is in the mathematical tools used to prove the lower bounds on the trade-off between communication and storage.

**Outline of Paper:** In the next section, we describe our subset cover framework. In Section 3, we study the communication-storage trade-off by posing it as a combinatorial optimization problem and derive lower bounds for the optimal transmission overhead cost given the storage constraint and security consideration. In Section 4, we describe our Partition-and-Power design in detail and show that it can provide a secure broadcast encryption with the communication costs matching these lower bounds. In Section 5, we compare our design with other subset-cover schemes.

## 2 The Subset Cover Framework

Let $\mathcal{R} = \{r_1, \ldots, r_n\}$ be a set of $n$ receivers capable of receiving transmissions from a broadcaster $\mathcal{B}$. Let $\mathcal{X} \subseteq 2^{\mathcal{R}}$ be a collection of subsets of $\mathcal{R}$ which forms a *cover* for $\mathcal{R}$, i.e. every subset of $\mathcal{R}$ can be expressed as a union of some elements from $\mathcal{X}$. Formally,

$$\forall R \subseteq \mathcal{R},\ \exists X_R \subseteq \mathcal{X} \text{ such that } \bigcup_{X \in X_R} X = R.$$

Let

$$f_{\mathcal{X}}(R) = \min_{\substack{X_R \subseteq \mathcal{X}, \\ \bigcup_{X \in X_R} X = R.}} |X_R|,$$

and let (with slight abuse of notation)

$$f_{\mathcal{X}} = \max_{R \subseteq \mathcal{R}} f_{\mathcal{X}}(R).$$

Thus $f_{\mathcal{X}}$ is the maximum number of sets from $\mathcal{X}$ required to cover a subset of $\mathcal{R}$.

In our setting, for each $X \in \mathcal{X}$, there exists an establishment key $K_X$ that is known only to the receivers in $X$. Moreover, the broadcaster $\mathcal{B}$ has a message $M_X$ that uniquely addresses $X$. Note that the broadcaster $\mathcal{B}$ has $|\mathcal{X}|$ such different messages (requiring $\log(|\mathcal{X}|)$[1] bits per message) to *address* every subset in the cover.

Now to distribute a session key $K$ to an arbitrary set of receivers, say $R$, $\mathcal{B}$ first finds the smallest cover for $R$ by the elements from $\mathcal{X}$. Suppose it is $R = X_1 \cup X_2 \cup \ldots X_f$. Then $\mathcal{B}$ sends the following broadcast

$$\langle [M_{X_1}, \ldots, M_{X_f}, E_{K_{X_1}}(K), \ldots, E_{K_{X_f}}(K)], E_K(M) \rangle, \tag{1}$$

where $E_{K_X}(K)$ is the encryption of $K$ under $K_X$ and $E_K(M)$ is the broadcast message $M$ encrypted under $K$.

Any receiver stores all the establishment keys $K_X$ corresponding to each $X \in \mathcal{X}$ it is a member of. Now consider a receiver $r \in R$ who is supposed to receive the above broadcast. The receiver $r$ then must belong to one of the $X_i$s since $X_i$s cover $R$. Suppose $r \in X_1$. When $r$ starts receiving the above broadcast, it uses the initial message part (here $M_{X_1}$) to figure out which of its establishment keys (here $K_{X_1}$) to use for recovering the session key. Since the message is well-structured, it knows that it can use that key ($K_{X_1}$) to decrypt $E_{K_{X_1}}(K)$ and recover $K$. Later it uses the session key $K$ to recover the actual message $M$ from $E_K(M)$.

The transmission overhead, the quantity within the square brackets in (1), is thus proportional to $f$. As noted earlier, $\log(|\mathcal{X}|)$ bits would be sufficient for $M_X$. Let's suppose that all keys (and their encrypted avatars) are $t$ bits in length. Since $f_{\mathcal{X}}$ is the maximum number of sets in $\mathcal{X}$ to cover any subset of $\mathcal{R}$, the number of bits transmitted by $\mathcal{B}$ as a transmission overhead is $O(f_{\mathcal{X}}(\log(|\mathcal{X}|) + t))$.

*Example 1.* Let $\mathcal{X} = \{\{r_1\}, \ldots, \{r_n\}\}$. Then $f_{\mathcal{X}} = n$. In this case, the transmission overhead is $O(n \log n + nt)$ bits. This is essentially the standard technique of transferring the session key to each user individually. □

## 3 Optimization Problems

Observe that the establishment key $K_X$ must be unique for every $X \in \mathcal{X}$. Then our goal is to minimize the following expression

$$\mathcal{T} = f_{\mathcal{X}}(\log(|\mathcal{X}|) + t) \tag{2}$$

over all $\mathcal{X}$ that can cover $\mathcal{R}$ subject to

$$|\mathcal{X}| < 2^t. \tag{3}$$

[1] All logs in this paper are in base 2.

**Lemma 1.** $|\mathcal{X}| \geq n$.

*Proof.* Since the privileged set could potentially include only an individual receiver, any cover $\mathcal{X}$ of $\mathcal{R}$ must have $\{r_i\}$ ( $1 \leq i \leq n$) as its members. □

**Lemma 2.** $f_{\mathcal{X}} \log(|\mathcal{X}|) \geq n$.

*Proof.* Suppose $f_{\mathcal{X}} = j$. Since every subset of $\mathcal{R}$ is covered by $j$ or less elements of $\mathcal{X}$, we have

$$2^n \leq \binom{|\mathcal{X}|}{1} + \cdots + \binom{|\mathcal{X}|}{j} \leq |\mathcal{X}|^j \tag{4}$$

so that $|\mathcal{X}| \geq 2^{n/j}$ and $f_{\mathcal{X}} \log(|\mathcal{X}|) \geq n$. □

If we choose $\mathcal{X}$ to be all subsets of $\mathcal{R}$, then $f_{\mathcal{X}} = 1$ and $|\mathcal{X}| = 2^n$. Thus, $f_{\mathcal{X}} \log(|\mathcal{X}|) = n$, and the number of bits transmitted is $n + t$. This is optimal in light of Lemma 2. By (3), this also implies that $t \geq n$.

In practice, $n$ is typically very large. This would make the length of the establishment key exorbitant, and therefore, very difficult to manage as well as use for encryption. Security and efficiency considerations would suggest a range for $t$. We incorporate it in our analysis in the next subsection.

## 3.1 Constraining the Key Size

In practice, the key-size is restricted. We fix $t$. In this case, the goal is to limit the transmission overhead given a limit on the size of the cover, i.e. $|\mathcal{X}| < 2^t$. Since, by Lemma 1, $|\mathcal{X}| \geq n$, this implies $n < 2^t$.

We first obtain a lower bound on $f_{\mathcal{X}}$. Again, assume $f_{\mathcal{X}} = j$. From inequalities (3) and (4), we have

$$2^n \leq (2^t)^j.$$

Thus $f_{\mathcal{X}} = j \geq n/t$.[2] This and Lemma 2 imply that the optimal transmission overhead cost

$$\begin{aligned} \mathcal{T} &= f_{\mathcal{X}}(\log(|\mathcal{X}|) + t) \\ &\geq n + f_{\mathcal{X}} t \\ &\geq 2n. \end{aligned} \tag{5}$$

**Choice of $t$:** Note that the above lower bound for $\mathcal{T}$ is independent of $t$. Then what is a good choice of $t$? We need to consider security aspects against brute-force attack of an eavesdropper. The eavesdropper succeeds if it can find the establishment key $K_X$ associated with some subset $X$ in the cover for it can start receiving the broadcast meant for $X$.

[2] This bound can be tightened by observing that inequality (4) can be improved to $2^n \leq (|\mathcal{X}|e/j)^j$. This yields $j \geq n/(t - \log j + \log e)$.

Assume that an establishment key needs to be valid for 1 year. Assume that a brute force attack to recover the establishment key can perform $10^6$ tests per second. Thus a receiver can perform a maximum of $10^6 \times 24 \times 365 \times 3600 \approx 3.2 \times 10^{13}$ such tests. The key-length is $t$. Equating $2^t$ with $3.2 \times 10^{13}$, we get $t \approx 45$.

*Example 2.* Suppose $n = 2^{20} \approx 10^6$, and $t = 45$. Thus there are $|\mathcal{X}| = O(2^{45})$ distinct establishment keys stored on $n$ receivers. This implies on an average $\Omega(2^{25})$ establishment keys per receiver. Since each key is of length $t = 45$ bits, this amounts to the minimum 180MB of average storage per receiver. In practice, $t$ could be larger and a receiver's keys may be contained in an IC card with very limited memory. We, therefore, need to do the analysis for the lower bound of $\mathcal{T}$ with this storage constraint factor into consideration. □

## 3.2 Constraining the Storage at a Receiver

A bound on the storage at a receiver directly translates to the number of subsets a receiver can belong to in a subset cover. Let's fix a cover $\mathcal{X}$ of $\mathcal{R}$ such that $|\mathcal{X}| < 2^t$. Let $\mathcal{X}_i = \{X \mid r_i \in X \in \mathcal{X}\}$. Let's add a constraint that

$$\forall i : \ |\mathcal{X}_i| \leq 2^m. \tag{6}$$

Our goal is to minimize $\mathcal{T}$ subject to the constraints (3) and (6).

**Lemma 3.** *Suppose $f_{\mathcal{X}} = j$. Consider any subset $R \subseteq \mathcal{R}$ and let*

$$X_R = \{X_1, X_2, \ldots, X_k\} \subset \mathcal{X}$$

*be its cover such that $k = f_{\mathcal{X}}(R) \leq j$. Then, there exist $k$ distinct receivers $r_{i_1}, r_{i_2}, \ldots, r_{i_k} \in R$ such that $r_{i_p} \in X_p$ for $1 \leq p \leq k$.*

*Proof.* Without loss of generality, let $R = \{r_1, r_2, \ldots, r_{|R|}\}$. Now consider an undirected bipartite graph $G$ on the vertex set $V(G) = V_1 \cup V_2$ where

$$\begin{aligned} V_1 &= \{v_1, v_2, \ldots, v_k\}, \\ V_2 &= \{w_1, w_2, \ldots, w_{|R|}\}, \text{ and} \\ E(G) &= \{(v_p, w_q) \mid r_q \in X_p\}. \end{aligned}$$

For any $S \subseteq V_1$, let its neighborhood in $V_2$ be

$$N(S) = \{w_q \mid \exists v_p \in S \text{ such that } (v_p, w_q) \in E(G)\}.$$

Note that $X_R$ is the best cover for $R$ in $\mathcal{X}$. Moreover, as discussed in the proof of Lemma 1, $\{r_i\} \in \mathcal{X}$ for $1 \leq i \leq n$. Therefore, it is clear that $|N(S)| \geq |S|$. Otherwise, the sets $X_p$s ($\in \mathcal{X}_R$) corresponding to the nodes in $S$ could be replaced by sets $\{r_q\} \in \mathcal{X}$ for each $w_q \in N(S)$. Since they are $|N(S)| < |S|$ in number, this gives a better cover for $R$!

Thus, the Hall's condition [6–Ch. 2] for the existence of a perfect $V_1$-matching is satisfied. Therefore, there exist $k$ distinct nodes $w_{i_1}, w_{i_2}, \ldots, w_{i_k} \in V_2$ such that $(v_p, w_{i_p}) \in E$ for $1 \leq p \leq k$. This implies that the corresponding receivers $r_{i_1}, r_{i_2}, \ldots, r_{i_k} \in R$ exist such that $r_{i_p} \in X_p$ for $1 \leq p \leq k$. □

**Corollary 1.**

$$2^n \leq \sum_{k=1}^{f_{\mathcal{X}}} \binom{n}{k} 2^{mk}. \tag{7}$$

*Proof.* Let $\mathcal{P} = 2^{\mathcal{R}}$ denote the power set of $\mathcal{R}$. Obviously, $|\mathcal{P}| = 2^n$. Let

$$\mathcal{P}'_k = \{X_1 \cup X_2 \cup \ldots \cup X_k \mid \exists 1 \leq i_1 < i_2 < \ldots < i_k \leq n : \mathcal{X}_{i_p} \ni X_p\},$$
$$\mathcal{P}' = \bigcup_{k=1}^{f_{\mathcal{X}}} \mathcal{P}'_k.$$

Consider any $R \subseteq \mathcal{R}$. Let its best cover be $X_R = \{X_1, X_2, \ldots, X_k\}$ in $\mathcal{X}$. By Lemma 3, $R$ must have some $k \leq f_{\mathcal{X}}$ elements, say $r_{i_1}, r_{i_2}, \ldots, r_{i_k}$ such that $r_{i_p} \in X_p$ for $1 \leq p \leq k$. Thus, by definition of $\mathcal{P}'_k$, $R \in \mathcal{P}'_k$. Since $R$ is any subset of $\mathcal{R}$, $\mathcal{P} = \mathcal{P}'$. By definition above,

$$2^n = |\mathcal{P}| = |\mathcal{P}'| \leq \sum_{k=1}^{f_{\mathcal{X}}} |\mathcal{P}'_k| \leq \sum_{k=1}^{f_{\mathcal{X}}} \binom{n}{k} 2^{mk}.$$

□

For $j \leq n/2$, it is easy to verify that

$$\sum_{i=1}^{j} \binom{n}{i} 2^{mi} \leq 2 \binom{n}{j} 2^{mj}.$$

Letting $j = f_{\mathcal{X}}$, using $\binom{n}{j} \leq (en/j)^j$ approximation, and applying above observation to the RHS of (7), we get

$$\begin{aligned} 2^n &\leq 2\binom{n}{j} 2^{mj} \\ &\leq 2(e2^m n/j)^j. \end{aligned} \tag{8}$$

Solving (8) for $j = f_{\mathcal{X}}$, we get $j \geq \frac{n-1}{m+\log en - \log j}$. This implies the following simplified lower bound:

$$f_{\mathcal{X}} \geq \frac{n-1}{m + 2 + \log(m + \log en)}. \tag{9}$$

The inequality (9) and Lemma 2 imply that the optimal transmission overhead cost

$$\begin{aligned} \mathcal{T} &= f_{\mathcal{X}}(\log(|\mathcal{X}|) + t) \\ &\geq n + f_{\mathcal{X}} t \\ &\geq n + \frac{(n-1)t}{\min(m + 2 + \log(m + \log en), t)}. \end{aligned} \tag{10}$$

*Example 3.* Suppose $n = 2^{20} \approx 10^6$, and $t = 45$. Let $m = 7$. Thus, a receiver can store upto $2^7 = 128$ establishment keys amounting to less than 1KB of memory. The lower bound of (10) tells us that the transmission overhead is at least 4.25Mbits. This is better than the lower bound of 2Mbits provided by (5). □

## 4 The Partition-and-Power Scheme

Let $g = \min(\lceil t - \log n \rceil, m+1)$. Let's partition $\mathcal{R}$ into groups of size $g$ each as follows: define the $j^{\text{th}}$ group of receivers as

$$G_j = \{r_{(j-1)g+1}, \ldots, r_{jg}\}$$

for $j = 1, \ldots, f = n/g$. Let $2^{G_j}$ denote the *Power Set* of $G_j$, *i.e.* the collection of all subsets of $G_j$. Then let

$$\mathcal{X}^{\text{PAP}} = 2^{G_1} \cup \cdots \cup 2^{G_f}.$$

Note that both (3) and (6) are satisfied by $\mathcal{X}^{\text{PAP}}$. Namely,

1. $|\mathcal{X}^{\text{PAP}}| = \frac{n}{g} 2^g \leq \frac{n}{t - \log n} \cdot \frac{2^t}{n} \leq 2^t$, and
2. $|\mathcal{X}_i^{\text{PAP}}| = 2^{g-1} \leq 2^m$.

To distribute a session key $K$ to a subset $R$ of receivers, $R$ is written uniquely as a disjoint union

$$R = X_1 \cup \cdots \cup X_f$$

with $X_j \subseteq G_j$. Thus, $f_{\mathcal{X}^{\text{PAP}}} = f = n/g$. Depending on $R$, some of these $X_i$s could be empty. Let $1 \leq i_1 < i_2 < \ldots < i_k \leq f$ be the indices of the non-empty $X$s. Now the overall broadcast is

$$\langle [M_{i_1}, \ldots, M_{i_k}, E_{K_{X_{i_1}}}(K), \ldots, E_{K_{X_{i_k}}}(K)], E_K(M) \rangle. \tag{11}$$

where $E_{K_X}(K)$ is the encryption of $K$ under the establishment key $K_X$ and $E_K(M)$ is the broadcast message $M$ encrypted under $K$. The message $M_{i_j}$ (for $1 \leq j \leq k$) has two components: the first component is of length $\log f = \log n - \log g$ bits and it indicates the group number $i_j$. The second component is of length $g$ bits and it addresses the subset $X_{i_j} \in 2^{G_{i_j}}$. Thus, the total length of the message $M_{i_j}$ is at most $\log n - \log g + t - \log n \leq t$ bits. Therefore the transmission overhead, the quantity in the square brackets in (11), is of length at most $2kt \leq 2 f_{\mathcal{X}^{\text{PAP}}} t = \frac{2nt}{g}$ bits. Thus, for the example 3, the PaP scheme will have transmission overhead of 11.25Mbits.

Note that each receiver stores $2^{g-1} \leq 2^m$ keys, one for each subset it belongs to in the group. Thus, the storage at each receiver is $t2^{g-1} = O(\frac{t2^t}{n})$ bits.

**Salient Features of the PaP Scheme:**

- The PaP scheme can handle privileged sets of any size.
- It is fully resilient in the sense that no collusion of non-receivers can break this scheme.
- In practice, $t$ is large for security reasons, while $m$ is small owing to memory limitations. Thus, $t > m + \log n$ is often the case. Then the transmission overhead of the PaP scheme is within a factor three of the optimal, *i.e.* $< 3\mathcal{T}$.
- Addition and removal of a receiver is very easy to handle in the PaP scheme. To remove a receiver $r_i$ from the broadcast network, it is marked as unused. To add a new receiver to the network, if there is any group containing an unused receiver, it is assigned to this new receiver. Otherwise, $t$ may have to be increased to accommodate the creation of a new group. The earlier receivers can be designed to truncate messages to $t$ bits and ignore the additional bits.

## 5 A Comparison with Other Subset Cover Schemes

In this section, we compare our lower bounds and the PaP scheme to other prevailing subset cover schemes. As already pointed out in the Subsection 1.2, we will not consider the schemes and the bounds from [4] in these comparisons.

In [9], the authors consider the privileged sets of only a certain fixed size. In out model, any subset out of universe of $n$ receivers could be a privilged set. Thus our model is more general that that described in [9]. Moreover, the lower bounds and the schemes in [9] are geared more towards small or large sized privileged sets. For example, if the number of revoked users is $r = n/2$, then the lower bound derived in (10) is better than the one derived from [9–Theorem 12].

In [10], the authors propose two subset cover schemes. For $r$ revoked users, the complete subtree method requires a transmission overhead of $r \log(n/r)$ and a storage of $\log n$ keys per receiver. The subset difference method requires $(2r-1)$ transmission overhead and $(\log^2 n)/2$ storage.

These bounds are improved in the Layered Subset-Difference (LSD) scheme of Halevy and Shamir [8]. In particular, their scheme uses $O(\log^{3/2} n)$ keys per receiver and a transmission overhead of $4r$ messages in the worst case and $2r$ on average.

Continuing with the example 3, let's suppose that the number of revoked users is $r = n/2$. Here the subset difference scheme requires storage of 200 keys per receiver and transmission overhead of $n$ messages. The LSD scheme requires storage of $\approx$ 100 keys per receiver and transmission overhead of $2n$ messages. The PaP scheme, with $g = m + 1 = 8$, requires 128 keys per receiver and a transmission overhead of $n/4$ messages. Thus the savings in the transmission overhead due to the PaP scheme are significant without any big strain on the memory.

## 6 Conclusion

In this paper, we have studied two natural combinatorial optimization problems related to the subset cover framework for broadcast encryption. Our objective was to minimize the transmission overhead cost given an upper bound on the number of keys per receiver and a lower bound on the key-length. These are important parameters to study because they measure quantities that affect the security, cost-effectiveness, and speed of a broadcast encryption system.

We derived lower bounds for the optimal communication cost for both problems and showed that these bounds are essentially tight by constructing the Partition-and-Power (PaP) subset cover scheme. The PaP scheme is fully resilient against any collusion of non-privileged receivers and it can handle privileged sets of any size. Moreover it can easily adapt to dynamic environments where receivers can join and/or leave.

## References

1. M. Abdalla, Y. Shavitt, and A. Wool. Towards making broadcast encryption practical. In M. Franklin, editor, *Proceedings of Financial Cryptography'99*, Lecture Notes in Computer Science 1648, Springer-Verlag, 1999, pp. 140–157.
2. S. Berkovits. How to Broadcast a Secret. *Advances in Cryptology*-Eurocrypt'91, Lecture Notes in Computer Science 547, Springer 1991, pp. 536–541.
3. C. Blundo and A. Cresti. Space Requirements for Broadcast Encryption. *Advances in Cryptology*-Eurocrypt'94, Lecture Notes in Computer Science 950, Springer-Verlag, 1995, pp. 287-298.
4. C. Blundo, L. A. Frota Mattos and D. R. Stinson. Tradeoffs Between Communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution. *Advances in Cryptology*-CRYPTO'96, Lecture Notes in Computer Science 1109, Springer-Verlag, 1996, pp. 387–400.
5. G.H. Chiou and W.T. Chen. Secure Broadcasting Using the Secure Lock. *IEEE Transactions on Software Engineering*, Vol SE-15(8), August 1989, pp. 929–934.
6. R. Diestel. *Graph Theory.* Springer-Verlag, 2005.
7. A. Fiat and M. Naor. Broadcast Encryption. *Advances in Cryptology*-CRYPTO'93, Lecture Notes in Computer Science 773, Springer-Verlag, 1993, pp. 480–491.
8. D. Halevy and A. Shamir. The LSD Broadcast Encryption Scheme. *Advances in Cryptology*-CRYPTO'02, Lecture Notes in Computer Science 2442, Springer-Verlag, 2002, pp. 47–60.
9. M. Luby and J. Staddon. Combinatorial Bounds for Broadcast Encryption. *Advances in Cryptology*-Eurocrypt'98, Lecture Notes in Computer Science 1403, Springer, 1998, pp. 512–526.
10. D. Naor, M. Naor and J. Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. *CRYPTO 2001*, pp. 41–62.
11. A. Shamir. How to Share a Secret. *Communications of the ACM*, Vol 22(11), November 1979, pp. 612–613.
12. D. R. Stinson. On some methods for unconditionally secure key distribution and broadcast encryption. *Designs, Codes and Cryptography*, Vol 12, 1997, pp. 215–243.

# MaTRU: A New NTRU-Based Cryptosystem

Michael Coglianese[1,⋆] and Bok-Min Goi[2,⋆⋆]

[1] Macgregor, 321 Summer Street, Boston, MA 02210, USA
mcoglian@comcast.net

[2] Centre for Cryptography and Information Security (CCIS),
Faculty of Engineering,
Multimedia University, 63100 Cyberjaya, Malaysia
bmgoi@mmu.edu.my

**Abstract.** In this paper, we propose a new variant of the NTRU public key cryptosystem − the MaTRU cryptosystem. MaTRU works under the same general principles as the NTRU cryptosystem, except that it operates in a different ring with a different linear transformation for encryption and decryption. In particular, it operates in the ring of $k$ by $k$ matrices of polynomials in $\mathbf{R} = \mathbb{Z}[X]/(X^n - 1)$, whereas NTRU operates in the ring $\mathbb{Z}[X]/(X^N - 1)$. Note that an instance of MaTRU has the same number of bits per message as an instance of NTRU when $nk^2 = N$. The improved efficiency of the linear transformation in MaTRU leads to respectable speed improvements by a factor of $O(k)$ over NTRU at the cost of a somewhat larger public key.

**Keywords:** Public key cryptosystems, NTRU, lattice based cryptography, lattice attacks, partial polynomial evaluation.

## 1 Introduction

Since the concept of public key cryptography was first introduced by Diffie and Hellman [4] in 1976, there has been steadily increasing interest in cryptographic studies; many public key cryptosystems have been proposed, i.e. RSA [22] based on integer factorization problem, the McEliece systems [15] based on algebraic coding theory, the ECC systems [12] based on the intractability of elliptic curve DLP and the variants of Matsumoto-Imai cryptosystems [14, 3] based on the systems of multivariable polynomials. Unfortunately, in practice many of these algorithms are costly in terms of computational and space complexity. These costs inhibit the ability of these algorithms to be substituted for symmetric key cryptosystems, and it prevents some of them (e.g. RSA) from running effectively on low power computing devices such as low-cost smart cards, RFID devices, and cell phones. As a result, cryptographers continue to look for new, fast public key cryptosystems, especially those which are based on different hard problems. Since 1996, researchers from NTRU Cryptosystems [21] have proposed a group

⋆ The first author initially performed this work at Brown University, USA.
⋆⋆ The second author acknowledges the Malaysia IRPA grant (04-99-01-00003-EAR).

© Springer-Verlag Berlin Heidelberg 2005

of fast public key cryptosystems based on partial evaluation of constrained polynomials over polynomial rings. These cryptosystems include the NTRU public key encryption algorithm [8] and the digital signature scheme NTRUSign [7].

Next, let us briefly describe one of these cryptosystems, NTRU. NTRU is a public key cryptosystem that operates in the ring $\mathbb{Z}[X]/(X^N - 1)$. Encryption and decryption of a message corresponds to applying a linear transformation to a ring element. Since this linear transformation performs the multiplication of two polynomials, the cost of applying it is $O(N^2)$ operations (assuming Fast Fourier Transforms are not used). In addition, these operations are on small integers, allowing for further speed optimizations. For these reasons, the speed of NTRU is one of its strongest features. NTRU operates considerably faster than both RSA and ECC at relatively the same security levels [13, 8]. However, the speed of NTRU can be further improved by choosing a different ring and applying a more efficient linear transformation [9, 10]. The hard problem underlying this cryptosystem is related to finding short vectors in a lattice due to the properties of short polynomials used in the system [2, 16, 20]. Since NTRU was proposed, it has been cryptanalyzed heavily by the cryptographic community, and some interesting results can be found in [5, 6, 11, 17, 19]. Meanwhile, some variants of NTRU encryption schemes have also been proposed, such as the generalized NTRU schemes [1].

The MaTRU cryptosystem, described in this paper, uses a more efficient linear transformation while providing a security level comparable to that of NTRU. MaTRU operates in the ring of $k$ by $k$ matrices of polynomials in $\mathbf{R} = \mathbb{Z}[X]/(X^n - 1)$. Note that an instance of MaTRU has the same number of bits per message as an instance of NTRU when $nk^2 = N$. While NTRU involves performing a one-sided multiplication for encryption and decryption [8], the linear transformation applied in MaTRU is a two-sided matrix multiplication. This means that the private key in MaTRU has two ring elements, as opposed to one ring element in NTRU. This is essential because multiplying on one side just gives a search space of size, say $S$, for the private key and the effect would be linear. Then, a lattice attack could be mounted very similar to the one on NTRU. However, multiplying on both sides will amplify the space of all linear transformations to $S^2$. The lattice attack will be extremely hard, due to the high dimension lattice matrix. Another difference between the two cryptosystems is that the ring in MaTRU is not commutative. This means that the matrices in the private key and the random matrices applied during encryption must specifically be constructed so that they commute with each other.

Since applying the linear transformation in MaTRU involves matrix multiplications, the encryption and decryption processes only require $O(n^2k^3)$ operations. This results in a speed increase by a factor of $O(k)$ over NTRU. In practice, this increase is approximately $\frac{k}{2}$ since MaTRU uses two matrix multiplications for every polynomial multiplication in NTRU. The private and public key lengths for MaTRU are $O(nk^2)$, making them comparable to key lengths in NTRU.

In the next section, we describe our proposed MaTRU cryptosystem in detail. Then, we discuss the parameter selection for MaTRU in Section 3. In Section 4, we present the details of the security analysis of the proposed scheme. We show its security strength based on certain parameter choices and compare it with standard NTRU in Section 5. Finally, we summarize our conclusions in Section 6.

## 2 The MaTRU Algorithm

### 2.1 Notation

The MaTRU cryptosystem operates in the ring $\mathbf{M}$ of $k$ by $k$ matrices of elements in the ring $\mathbf{R} = \mathbb{Z}[X]/(X^n - 1)$. The ring $\mathbf{R}$ consists of polynomials with degree at most $(n-1)$ having integer coefficients. Multiplication and addition of polynomials in $\mathbf{R}$ is done in the usual manner, but exponents of $X$ are reduced modulo $n$. Matrix multiplication in $\mathbf{M}$ is denoted using the $*$ symbol.

Besides $n$ and $k$, MaTRU also uses the parameters $p, q \in \mathbb{N}$. The numbers $p$ and $q$ may or may not be prime, but they must be relatively prime. In general, $p$ is much smaller than $q$; in this paper, for ease of explanation, we stick to $p = 2$ or $p = 3$ and $q$ in the range of $2^8$ to $2^{11}$. When we say we perform a matrix multiplication modulo $p$ (or $q$), we mean that we reduce the coefficients of the polynomials in the matrices modulo $p$ (or $q$).We define the *width* of an element $M \in \mathbf{M}$ to be $|M|_\infty = (\max_{\text{polys. } m \text{ in } M} \text{coeff. in } m) - (\min_{\text{polys. } m \text{ in } M} \text{coeff. in } m)$. The width of $M$ is the maximum coefficient in any of its $k^2$ polynomials minus the minimum coefficient in any of its polynomials. We say a matrix $M \in \mathbf{M}$ is *short* if $|M|_\infty \leq p$. When short matrices are multiplied together, we get a matrix which has a width which may be greater than $p$ but is still almost certainly smaller than $q$; we call this matrix *pretty short*. The definitions for width and shortness apply similarly to polynomials in $\mathbf{R}$. For $r \in \mathbf{R}$, $|r|_\infty = (\max\ \text{coeff. in r}) - (\min\ \text{coeff. in r})$. The polynomial $r$ is said to be short if $|r|_\infty \leq p$. We also define the *size* of an element $M \in \mathbf{M}$ to be $|M| = \sqrt{\sum_{\text{polys. } m \text{ in } M} \sum (\text{coeff. in } m)^2}$.

When defining some of the sets of short matrices below, we use the notation

$$\mathcal{L}(d) = \left\{ M \in \mathbf{M} \;\middle|\; \begin{array}{l} \text{for } i = \left\lceil -\frac{p-1}{2} \right\rceil \dots \left\lceil \frac{p-1}{2} \right\rceil, i \neq 0, \text{ each polynomial} \\ \text{in } M \text{ has on average } d \text{ coefficients equal to } i, \\ \text{with the rest of the coefficients equal to 0.} \end{array} \right\}.$$

For example, if $p = 3$ and $n = 5$, then $\mathcal{L}(2)$ consists of all matrices of polynomials where on average each polynomial has 2 coefficients equal to 1, 2 coefficients equal to $-1$, and 1 coefficient equal to zero. Or, if we had $p = 2$ and $n = 5$, then $\mathcal{L}(2)$ consists of all matrices of polynomials where on average each polynomial has 2 coefficients equal to 1 and 3 coefficients equal to zero.

The parameters for MaTRU consist of the four integers $(n, k, p, q)$ described above and the five sets of matrices $(\mathcal{L}_f, \mathcal{L}_\Phi, \mathcal{L}_A, \mathcal{L}_w, \mathcal{L}_m) \subset \mathbf{M}$. These sets have the following meanings and compositions:

| Set | Elements | Description | Composition |
|---|---|---|---|
| $\mathcal{L}_f$ | $f, g$ | Compose private key | Short; see (2) below |
| $\mathcal{L}_\Phi$ | $\Phi, \Psi$ | Random matrices applied for each encryption | Short; see (2) below |
| $\mathcal{L}_A$ | $A, B$ | Used to construct $f, g, \Phi, \Psi$ | Short; see (1) below |
| $\mathcal{L}_w$ | $w$ | Used to construct public key | Short |
| $\mathcal{L}_m$ | $m$ | Messages | Short; see (3) below |

1. $\mathcal{L}_A$ consists of all matrices $C \in \mathbf{M}$ such that $C^0, C^1, \ldots, C^{k-1}$ are linearly independent modulo $q$; and for short $c_0, \ldots, c_{k-1} \in \mathbf{R}$, $\sum_{i=0}^{k-1} c_i C^i$ is short. Section 3.2 describes the exact nature of $\mathcal{L}_A$ that satisfies these conditions.
2. $\mathcal{L}_f$ and $\mathcal{L}_\Phi$ consist of all matrices $D \in \mathbf{M}$ constructed such that, for $C \in \mathcal{L}_A$ and short $c_0, \ldots, c_{k-1} \in \mathbf{R}$, $D = \sum_{i=0}^{k-1} c_i C^i$. Additionally, matrices in $\mathcal{L}_f$ must satisfy the requirement that they have inverses modulo $p$ and modulo $q$.
3. The set of messages $\mathcal{L}_m$ consists of all matrices of polynomials with coefficients modulo $p$. We therefore express

$$\mathcal{L}_m = \left\{ M \in \mathbf{M} \,\middle|\, \begin{matrix} \text{polynomials in } M \text{ have coefficients} \\ \text{between } \lceil -\frac{p-1}{2} \rceil \text{ and } \lceil \frac{p-1}{2} \rceil \end{matrix} \right\} .$$

This means that each message contains $nk^2 \log_2 p$ bits of information.

## 2.2 Key Creation

To create a public/private key pair, Bob chooses two $k$ by $k$ matrices $A, B \in \mathcal{L}_A$. Next, Bob randomly selects short polynomials $\alpha_0, \alpha_1, \ldots \alpha_{k-1} \in \mathbf{R}$ and $\beta_0, \beta_1, \ldots \beta_{k-1} \in \mathbf{R}$. Bob then constructs the matrices $f, g \in \mathcal{L}_f$ by taking

$$f = \sum_{i=0}^{k-1} \alpha_i A^i \qquad \text{and} \qquad g = \sum_{i=0}^{k-1} \beta_i B^i .$$

As noted above in Section 2.1, the matrices $f$ and $g$ must have inverses modulo $p$ and modulo $q$. This will generally be the case, given suitable parameter choices. We denote the inverses as $F_p, F_q$ and $G_p, G_q$, where

$$F_q * f \equiv I (\text{mod } q) \qquad \text{and} \qquad F_p * f \equiv I (\text{mod } p);$$
$$G_q * g \equiv I (\text{mod } q) \qquad \text{and} \qquad G_p * g \equiv I (\text{mod } p).$$

Note that $I$ is a $k$ by $k$ identity matrix. Bob now has his private key, $(f, g)$, although in practice he will want to store the inverses $F_p$ and $G_p$ as well. Bob now selects a random matrix $w \in \mathcal{L}_w$, and constructs the matrix $h \in \mathbf{M}$ by taking

$$h \equiv F_q * w * G_q \qquad (\text{mod } q) .$$

Bob's public key consists of the three matrices, $(h, A, B)$.

## 2.3 Encryption

To encrypt a message to send to Bob, Alice randomly generates the short polynomials $\phi_0, \phi_1, \ldots \phi_{k-1} \in \mathbf{R}$ and $\psi_0, \psi_1, \ldots \psi_{k-1} \in \mathbf{R}$. Alice then constructs the matrices $\Phi, \Psi \in \mathcal{L}_\Phi$ by taking

$$\Phi = \sum_{i=0}^{k-1} \phi_i A^i \qquad \text{and} \qquad \Psi = \sum_{i=0}^{k-1} \psi_i B^i \ .$$

Alice then takes her message $m \in \mathcal{L}_m$, and computes the encrypted message

$$e \equiv p(\Phi * h * \Psi) + m \pmod q \ .$$

Alice then sends $e$ to Bob.

## 2.4 Decryption

To decrypt, Bob computes

$$a \equiv f * e * g \pmod q \ . \tag{1}$$

Bob translates the coefficients of the polynomials in the matrix $a$ to the range $-q/2$ to $q/2$ using the centering techniques as in the original NTRU paper [8]. Then, treating these coefficients as integers, Bob recovers the message by computing

$$d \equiv F_p * a * G_p \pmod p \ .$$

## 2.5 Why Decryption Works

In decryption, from Eq. [1] Bob has

$$\begin{aligned} a &\equiv f * (p(\Phi * h * \Psi) + m) * g && \pmod q \\ &\equiv p(f * \Phi * F_q * w * G_q * \Psi * g) + f * m * g && \pmod q \end{aligned}$$

Although matrix multiplication is not generally commutative, $f$ and $\Phi$ here do indeed commute:

$$\begin{aligned} f * \Phi &\equiv (\textstyle\sum_{i=0}^{k-1} \alpha_i A^i) * (\sum_{i=0}^{k-1} \phi_i A^i) && \pmod q \\ &\equiv \textstyle\sum_{i=0}^{k-1} \sum_{i \equiv j+\ell \ (\text{mod k})} \alpha_j A^j \phi_\ell A^\ell && \pmod q \\ &\equiv \textstyle\sum_{i=0}^{k-1} \sum_{i \equiv j+\ell \ (\text{mod k})} \phi_\ell A^{j+\ell} \alpha_j && \pmod q \\ &\equiv \textstyle\sum_{i=0}^{k-1} \sum_{i \equiv j+\ell \ (\text{mod k})} \phi_\ell A^\ell \alpha_j A^j && \pmod q \\ &\equiv (\textstyle\sum_{i=0}^{k-1} \phi_i A^i) * (\sum_{i=0}^{k-1} \alpha_i A^i) \equiv \Phi * f && \pmod q \end{aligned}$$

Similarly, $g * \Psi \equiv \Psi * g \pmod q$. So, Bob now has that

$$a \equiv p(\Phi * w * \Psi) + f * m * g \pmod q$$

For appropriate parameter choices, $|a|_\infty \leq q$. Then, treating the polynomials in this matrix as having coefficients in $\mathbb{Z}$, Bob can take those coefficients modulo $p$, leaving $f * m * g (\text{mod } p)$. The original message is then recovered by left-multiplying by $F_p$ and right-multiplying by $G_p$.

## 3 Parameter Selection

### 3.1 Selection of Pairs $(\boldsymbol{f}, \boldsymbol{g})$ and $(\boldsymbol{\Phi}, \boldsymbol{\Psi})$

We define $d_f$ and $d_\phi$ such that

$$\mathcal{L}_f = \mathcal{L}(d_f) \qquad \text{and} \qquad \mathcal{L}_\Phi = \mathcal{L}(d_\phi) \ .$$

Since the matrices $A$ and $B$ are public, the security of $f$, $g$, $\Phi$, and $\Psi$ necessarily depends on the difficulty of discovering the short polynomials $\alpha_i$, $\beta_i$, $\phi_i$, and $\psi_i$. For this reason, we want to maximize the number of possible choices for these polynomials. We therefore commonly select

$$d_f \approx \frac{n}{p} \qquad \text{and} \qquad d_\phi \approx \frac{n}{p} \ .$$

See section 4.1 for precise brute force security calculations.

*Remark 1.* A matrix $f$ in the ring $\mathbf{M}$ will be invertible modulo $p$ and $q$, only if the correspond matrix determinant $det_f$, which is in the ring $\mathbf{R}$, is also invertible modulo $p$ and $q$. In practice, this is impossible if $det_f(1) = 0$ (the sum of the coefficient values of the determinant polynomial is equal to 0). So we must re-select one or more of the polynomial elements in $f$ if this condition was not fulfilled.

### 3.2 Selection of $\boldsymbol{A}$ and $\boldsymbol{B}$

A main concern in generating the matrices $f$ and $\Phi$ (and likewise, $g$ and $\Psi$) is that they must not only commute, but they should also be short. Shorter matrices ensure that $|p(\Phi * w * \Psi) + f * m * g|_\infty$ will be smaller, which will allow us to reduce $q$ and valid ciphertexts will be decipherable.

To achieve this, we select $A$ and $B$ to be *permutation matrices.* A permutation matrix is a binary matrix (i.e. consisting of only the scalars 0 and 1) such that there is exactly one 1 in each row and column with all 0s elsewhere. Since $A$ and $B$ have the additional requirement that the sets $A^0, \ldots, A^{k-1}$ and $B^0, \ldots, B^{k-1}$ are both linearly independent, we have that

$$\sum_{i=0}^{k-1} A^i = \sum_{i=0}^{k-1} B^i = \begin{pmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \ldots & 1 \end{pmatrix} \ .$$

This implies that each row and column of $f$ will contain some permutation of $\alpha_0, \ldots, \alpha_{k-1}$, meaning that each $\alpha_i$ will appear $k$ times in $f$. An analogous situation exists for $g$, $\Phi$, and $\Psi$.

Using the common choice of $d_f \approx d_\phi \approx \frac{n}{p}$, we have that

$$|f| \approx \sqrt{k^2|\alpha_i|^2} \approx \sqrt{\frac{(p-1)nk^2}{p}} \approx |g| \approx |\Phi| \approx |\Psi| \ .$$

### 3.3 Selection of $w$

Like $f$ and $g$, $w$ should also be chosen to be short in order to keep $|p(\Phi * w * \Psi) + f * m * g|_\infty$ small. For security reasons, it is important that $w$ remain secret from an attacker. Therefore, in order to maximize the space of $w$ we make

$$\mathcal{L}_w = \mathcal{L}\left(\left\lfloor \frac{n}{p} \right\rfloor\right) .$$

The size of $w$ is then given by

$$|w| = \sqrt{\frac{(p-1)nk^2}{p}} .$$

*Remark 2.* Note that when $w$ is chosen in this manner, on average $|w| \approx |m|$. This means that $|\Phi * w * \Psi| \approx |f * m * g|$.

## 4 Security Analysis

### 4.1 Brute Force Attacks

To find a private key by brute force, an attacker must try all possible short pairs of matrices $(f, g)$ to find one such that $f * h * g$ is also short. Since the matrices $A$ and $B$ are public, $f$ and $g$ are determined by the $2k$ polynomials $\alpha_0, \ldots, \alpha_{k-1}, \beta_0, \ldots, \beta_{k-1}$. Each of these polynomials has degree $n-1$, so the number of possible $(f, g)$ pairs is

$$\text{(key security)} = \left(\frac{n!}{(n-(p-1)d_f)!d_f!^{(p-1)}}\right)^{2k} . \tag{2}$$

Similarly, the encryption of a particular message is determined by the $2k$ polynomials $\phi_0, \ldots, \phi_{k-1}, \psi_0, \ldots, \psi_{k-1}$, so we have the same message security as Eq. [2] with replacing $d_f$ by $d_\phi$. Using a meet-in-the-middle attack, such as the method due to Odlyzko [18] used on the standard NTRU algorithm, assuming sufficient memory storage, the key and message security would be equal to the square root of the above values. Note that for the standard NTRU algorithm with the suggested parameters, the meet-in-the-middle attack is the most effective known attack.

### 4.2 Lattice Attacks

**Key security.** Message decryption, which left-multiplies the encrypted message by $f$ and right-multiplies it by $g$, amounts to the application of a linear transformation $T : \mathbf{M} \to \mathbf{M}$ such that both $T$ and $T(h)$ are short. If this was the case, then it is likely that either $T$ is the transformation corresponding to the one given by the actual private key, or that $T$ will work as a substitute for the private key in decrypting messages.

Let $T_{f,g} : \mathbf{M} \to \mathbf{M}$ be the linear transformation corresponding to decryption with the actual private key. Then $T_{f,g}$ is defined by $T_{f,g}(J) : J \to f * J * g$ . What does the transformation $T_{f,g}$ look like? To see this, we look at where $T_{f,g}$ takes the basis matrices for the space of all possible matrices $J$. The basis consists of the $k^2$ matrices $\delta_{i,j}$, where $\delta_{i,j}$ has a 1 in position $(i, j)$ and 0s elsewhere. We then have that $T_{f,g} = \begin{pmatrix} f\delta_{0,0}g & f\delta_{0,1}g & \dots & f\delta_{k-1,k-1}g \end{pmatrix}$. This describes how $T_{f,g}$ maps the basis matrices for the space of possible $J$'s: $\delta_{0,0} \to f\delta_{0,0}g$, $\delta_{0,1} \to f\delta_{0,1}g$, and so on.

Since $f = \sum_{i=0}^{k-1} \alpha_i A^i$ and $g = \sum_{i=0}^{k-1} \beta_i B^i$, we can express $T_{f,g}$ as a combination of the $k^2$ linear transformations $T_{A^i,B^j}$, where $T_{A^i,B^j} = \begin{pmatrix} A^i\delta_{0,0}B^j & A^i\delta_{0,1}B^j & \dots & A^i\delta_{k-1,k-1}B^j \end{pmatrix}$. We then have that $T_{f,g}(J) = \sum_{i,j} \gamma_{i,j} T_{A^i,B^j}(J)$ . In this formula, each polynomial $\gamma_{i,j}$ is the multiple of $T_{A^i,B^j}$ needed to produce the particular transformation $T_{f,g}$. Therefore, we have precisely that $\gamma_{i,j} = \alpha_i \beta_j$.

Now, since the $\alpha_i$'s and $\beta_j$'s are short, the polynomials $\gamma_{i,j}$ will be pretty short. In addition, $T_{f,g}(h) = w$ is short. So, the linear transformation $T_{f,g}$ corresponds to a short target vector $(\gamma_{0,0}, \gamma_{0,1}, \dots, \gamma_{k-1,k-1}, w)$ in the lattice $L = \{(T, T(h))\}$. This lattice $L$ is generated by the rows of the following $2nk^2$ by $2nk^2$ matrix composed of four $nk^2$ by $nk^2$ blocks:

$$\left(\begin{array}{cccc|cccc}
1 & 0 & \dots & 0 & h_{0,0} & h_{0,1} & \dots & h_{k-1,k-1} \\
0 & 1 & \dots & 0 & h_{k-1,k-1} & h_{0,0} & \dots & h_{k-1,k-2} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 1 & h_{0,1} & h_{0,2} & \dots & h_{0,0} \\
\hline
0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\
0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \dots & 0 & 0 & 0 & \dots & q
\end{array}\right)$$

In the above matrix, the $n$ by $n$ matrix $h_{i,j}$ represents the $n$ coefficients of the polynomial at position $(i, j)$ in $h$. Note that $\det L = q^{nk^2}$ and $\dim L = 2nk^2$.

As noted earlier, each $\gamma_{i,j} = \alpha_i \beta_j$, so $|\gamma_{i,j}| = |\alpha_i \beta_j| \approx |\alpha_i||\beta_j| \approx (p-1)d_f$. There are $k^2$ $\gamma_{i,j}$ polynomials, so the size of the target vector $(\gamma_{0,0}, \gamma_{0,1}, \dots, \gamma_{k-1,k-1}, w)$ is given by

$$|\text{target}| \approx \sqrt{((p-1)d_f)^2 k^2 + |w|^2}.$$

Using the suggested $d_f \approx \frac{n}{p}$ and $|w| = \sqrt{\frac{(p-1)nk^2}{p}}$ yields

$$|\text{target}| \approx \sqrt{\frac{(p-1)nk^2((p-1)n+p)}{p^2}}.$$

By the Gaussian heuristic, the expected shortest vector in $L$ is

$$|\text{exp. shortest}| = \sqrt{\frac{\dim L}{2\pi e}}(\det L)^{\frac{1}{\dim L}} = \sqrt{\frac{qnk^2}{\pi e}}.$$

Let $c_h$ equal the ratio of the target vector to the expected shortest vector. If $c_h$ is near 0, the target vector will likely be much smaller than any other vectors in the lattice, and will therefore be easier to find. If $c_h$ is near 1, then there will likely be many vectors near the size of the target, making the target difficult to find. In our case,

$$c_h \approx \frac{|\text{target}|}{|\text{exp. shortest}|} \approx \sqrt{\frac{\pi e(p-1)((p-1)n+p)}{p^2 q}}.$$

For example, the MaTRU parameters suggested in section 5 give values for $c_h$ around 0.2. This means that if the LLL algorithm finds a vector in the lattice $L$ around two tenth the size of the expected shortest vector, then the algorithm has most likely found $T_{f,g}$ or another suitable linear transformation.

**Message security.** A lattice attack can also be used to try to discover a particular message. The way this is done is very similar to the lattice attack on a key. Since we selected the parameters $d_f \approx d_\phi$ and $|w| \approx |m|$, we have that $|\Phi * w * \Psi| \approx |f * m * g|$. So the lattice security of a message will be the same as that of the key, meaning $c_m \approx c_h$. The constant $c_m$ indicates how difficult it will be to discover a particular message. Finding the message will be more difficult when $c_m$ is close to 1 and easier when $c_m$ is close to 0.

*Remark 3.* The above lattice matrix for MaTRU can be further optimized by multiplying the top-left $nk^2$ by $nk^2$ identity submatrix by a scaling factor, $\alpha$, as in [8]. Also, by using zero-forcing technique [16], we can reduce the dimension of the lattice matrix and increase the performance of lattice attacks. These considerations will be taken into account in a future revision of the parameter choices for MaTRU.

## 5 Discussion

### 5.1 Parameter Choices

Table 1 shows some possible parameter choices for MaTRU along with their brute force and lattice security levels. Key and message securities listed below are for a meet-in-the-middle attack; these values should be squared for a regular brute force attack.

**Table 1.** Possible parameter choices for MaTRU

| $n$ | $k$ | $p$ | $q$ | $d_f$ | $d_\phi$ | key security | msg. security | $c_h$ | $\dim L$ |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 15 | 3 | 2048 | 2 | 2 | $2^{97.4}$ | $2^{97.4}$ | 0.118 | 2700 |
| 8 | 9 | 3 | 1024 | 2 | 2 | $2^{78.4}$ | $2^{78.4}$ | 0.188 | 1296 |
| 11 | 6 | 3 | 1024 | 3 | 3 | $2^{79.0}$ | $2^{79.0}$ | 0.215 | 792 |
| 16 | 8 | 2 | 379 | 8 | 8 | $2^{109.2}$ | $2^{109.2}$ | 0.318 | 2048 |
| 18 | 5 | 2 | 251 | 9 | 9 | $2^{77.8}$ | $2^{77.8}$ | 0.412 | 880 |

## 5.2 Comparison with Standard NTRU

Here we compare the theoretical operating characteristics of MaTRU with those of NTRU, as shown in Table 2. The properties are listed in terms of the parameters $(N, p, q)$ for NTRU and the parameters $(n, k, p, q)$ for MaTRU. These should be compared by setting $N = nk^2$, since this equates to plain text message blocks of the same size.

**Table 2.** Comparison between MaTRU with NTRU

| Characteristic | NTRU [8] | MaTRU [this paper] |
|---|---|---|
| Plain Text Block | $N \log_2 p$ bits | $nk^2 \log_2 p$ bits |
| Encrypted Text Block | $N \log_2 q$ bits | $nk^2 \log_2 q$ bits |
| Encryption Speed | $O(N^2)$ operations | $O(n^2k^3)$ operations[1] |
| Decryption Speed | $O(N^2)$ operations | $O(n^2k^3)$ operations[1] |
| Message Expansion | $\log_p q$-to-1 | $\log_p q$-to-1 |
| Private Key Length | $2N \log_2 p$ bits | $2nk^2 \log_2 p$ bits[2] |
| Public Key Length | $N \log_2 q$ bits | $3nk^2 \log_2 q$ bits[3] |
| Key Security (Message Security)[4] | $\frac{N!}{d_g!^2(N-2d_g)!}$ | $\left(\frac{n!}{((n-2)d_f)!d_f!^2}\right)^{2k}$ |
| Lattice Security, $c_h$ $(c_m)$[5] | $2\left(\frac{\pi^2 de^2}{3Nq^2}\right)^{\frac{1}{4}}$ | $\frac{1}{3}\sqrt{\frac{2\pi e(2n+3)}{q}}$ |

[1] Since MaTRU performs two-sided multiplications, the constant factor will be about twice that of standard NTRU.
[2] A key length of $2nk \log_2 p + 2k^2 \log_2 k$ bits can be achieved by storing $f$ and $g$ not as matrices but as the $2k$ polynomials found in the matrices along with their positions in the matrices.
[3]A key length of $nk^2 \log_2 q + 2k \log_2 k$ bits can be achieved by storing $A$ and $B$ not as matrices but as the positions of each of the $k$ 1s in the two matrices.
[4]For message security, $d_g$ is replaced by $d$ for NTRU whereas $d_f$ is replaced by $d_\phi$ for MaTRU. For ease of comparison, we fix $p = 3$. We refer the readers to [8] for the definition of $d_g$ and $d$ used in NTRU.
[5]Note that $c_h \approx c_m$, so that we have equivalent security level at key and message. For ease of comparison, we fix $p = 3$.

As indicated by the table, the total time for encryption and decryption is $\frac{k}{2}$ times faster for MaTRU than for NTRU. MaTRU has a larger public key length as a result of needing to store the matrices $A$ and $B$, but a smaller private key length due to the particular nature of the private keys $f$ and $g$.

For example, compare the NTRU "high" security level of $(N, p, q) = (263, 3, 128)$ with the MaTRU parameter choices of $(n, k, p, q) = (18, 5, 2, 251)$. NTRU in this case would have a plain text block size of 417 bits, a private key length of 834 bits, and a public key length of 1841 bits. MaTRU would have a plain text block size of 450 bits, a private key length of 297 bits, and a public key length of 3611 bits. MaTRU would theoretically be 2.5 times faster at encryption/decryption than the instance of NTRU in this case.

## 6 Conclusion

We have presented the MaTRU cryptosystem in detail, and we have shown that its security level is comparable to NTRU with respect to several well-known attacks, including brute force attacks, lattice attacks and meet-in-the-middle attacks. However, the security analysis of MaTRU is heuristic because there may be a better attack on it than on the original NTRU. Further research on the lattice attack on MaTRU may yield new techniques (e.g. subdividing the lattice) that are more effective. We have also suggested several parameter choices for MaTRU that provide a significant speed improvement over NTRU with relatively similar security levels. Future work to obtain precise running times and lattice attack times will allow for further refinements to the list of suggested MaTRU parameters in Table 1. Additionally, the introduction of the commutative family (using permutation matrices) has been given a reasonable scrutiny but would benefit from further analysis. Finally, we believe that the continued study of optimization, improvement and cryptanalysis of MaTRU based on the previously proposed techniques used with the original NTRU, especially the impact of imperfect decryption [17], presents interesting challenges to explore.

## Acknowledgement

We would like to thank Jeffrey Hoffstein for his advice and suggestions, Wee-Keong Lim for the discussion in matrix theory and anonymous referees for their constructive and detailed comments that have greatly improved this paper.

## References

1. William D. Banks and Igor E. Shparlinski. A variant of NTRU with non-invertible polynomials. In *Proceeding of Indocrypt '02*, LNCS, vol. 2551, Springer-Verlag, pp.62-70, 2002.
2. D. Coppersmith and A. Shamir. Lattice attacks on NTRU. In *Proceeding of Eurocrypt '97*, LNCS, vol. 1233, Springer-Verlag, pp.52-61, 1997.
3. J. Ding. A new variant of the Matsumoto-Imai cryptosystem through perturbation. In *Proceeding of PKC '04*, LNCS, vol. 2947, Springer-Verlag, pp.305-318, 2004.
4. W. Diffie and M.E. Hellman. New directions in cryptography. In *IEEE Trans. On Information Theory*, vol. 22, pp.644-654, 1976.
5. C. Gentry. Key recovery and message attacks on NTRU-composite. In *Proceeding of Eurocrypt '01*, LNCS, vol. 2045, Springer-Verlag, pp.182-194, 2001.
6. Daewan Han, Jin Hong, Jae Woo Han and Daesung Kwon. Key recovery attacks on NTRU without ciphertext validation routine. In *Proceeding of ACISP '03*, LNCS, vol. 2727, Springer-Verlag, pp.274-284, 2003.
7. J. Hoffstein, N. Howgrave-Graham, J. Pipher, J.H. Silverman and W. Whyte. NTRUSign: Digital Signatures Using the NTRU Lattice. In *Proceeding of CT-RSA '03*, LNCS, vol. 2612, Springer-Verlag, pp.122-140, 2003.
8. J. Hoffstein, J. Pipher and J.H. Silverman. NTRU: A Ring-Based Public Key Cryptosystem. In *Proceeding of ANTS III*, LNCS, vol. 1423, Springer-Verlag, pp. 267-288, 1998.

9. J. Hoffstein and J.H. Silverman. Optimizations for NTRU. In *Public-key Cryptography and Computational Number Theory*, DeGruyter, 2000. Available at [21].
10. J. Hoffstein and J.H. Silverman. Random small hamming weight products with applications to cryptography. Discrete Applied Mathematics, vol. 130, Issue 1 - special issue on the 2000 com2MaC workshop on cryptography, pp. 37 - 49, 2003. Available at [21].
11. E. Jaulmes and A. Joux. A Chosen Ciphertext Attack on NTRU. In *Proceeding of CRYPTO '00*, LNCS, vol. 1880, Springer-Verlag, pp. 20-35, 2000.
12. N. Koblitz. Elliptic curves cryptosystems. Math of Comp. vol. 48, pp. 203-209, 1987.
13. P. Karu and J. Loikkanen. Practical comparison of fast public-key cryptosystems. Seminar on Network Security, Telecommunications Software and Multimedia Laboratory, Kelsinki University of Technology. Available at http://www.tml.hut.fi/Opinnot/Tik-110.501/2000/papers.html.
14. T. Matsumoto and H. Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Proceeding of Eurocrypt '88*, LNCS, vol. 330, Springer-Verlag, pp.419-453, 1988.
15. R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. JPL DSN Progress Report 42-44, pp. 114-116, 1978.
16. A. May and J.H. Silverman. Dimension Reduction Methods for Convolution Modular Lattices. In *Proceeding of CaCL '01*, LNCS, vol. 2146, Springer-Verlag, pp. 110-125, 2001.
17. N. Howgrave-Graham, P.Q. Nguyen, D. Pointcheval, J. Proos, J.H. Silverman, A. Singer and W. Whyte. The Impact of Decryption Failures on the Security of NTRU Encryption. In *Proceeding of CRYPTO '03*, LNCS, vol. 2729, Springer-Verlag, pp. 226-246, 2003.
18. N. Howgrave-Graham, J.H. Silverman, W. Whyte, *NTRU Cryptosystems Technical Report #004, Version 2: A Meet-In-The-Middle Attack on an NTRU Private Key*, `www.ntru.com`.
19. P.Q. Nguyen and D. Pointcheval. Analysis and Improvements of NTRU Encryption Paddings. In *Proceeding of CRYPTO '02*, LNCS, vol. 2442, Springer-Verlag, pp. 210-225, 2002.
20. P.Q. Nguyen and J. Stern. The Two Faces of Lattices in Cryptology. In *Proceeding of CaCL '01*, LNCS, vol. 2146, Springer-Verlag, pp. 148-180, 2001.
21. NTRU Cryptosystems. Technical reports available at http://www.ntru.com/cryptolab/tech_notes.htm.
22. R.L. Rivest, A. Shamir, L. Adleman. A method for obtaining digital signatures and public key cryptosystem. *Communications of the ACM*, vol. 21, pp. 120-126, 1978.

# Anonymous Password-Based Authenticated Key Exchange

Duong Quang Viet, Akihiro Yamamura, and Hidema Tanaka

Department of Information and Networks Systems,
National Institute of Information and Communications Technology,
4-2-1 Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan
{viet, aki, hidema}@nict.go.jp

**Abstract.** We propose and discuss an *anonymous password-based authenticated key exchange scheme* that allows a user in a group to establish a session key with a server in an anonymous way. In our scheme, each user in a legitimate group and the server share a human-memorable password, and they can authenticate each other. The scheme is secure against the dictionary attack. Furthermore, we extend this to the scheme that allows any subgroup of at least $k$-out-of-$n$ users of the group to establish a session key with the server in an anonymous way.

**Keywords:** Password-based authenticated key exchange, Anonymous group authentication, Oblivious transfer.

## 1 Introduction

### 1.1 Password-Based Authenticated Key Exchange

The Diffie-Hellman protocol provides a nice way to establish a common key between two parties, however, it is vulnerable to the man-in-the-middle attack. Under the realistic circumstance, an adversary can control all communications between legitimate parties. In fact, an adversary can freely copy, resend, modify or delete messages transmitted between the legitimate parties. Therefore, it is essential to provide some mean to authenticate the entities and the key established. An *authenticated key exchange scheme* (AKE) is an essential technique to solve this problem. It allows two parties holding a long-term secret to authenticate the agreed session key that will be used for their secure communication.

There are two types of AKE. The first is the scheme where two parties already hold a long and random secret key. There are numerous schemes in this category. See [7] for examples. The second is a *password-based authenticated key exchange scheme* (PAKE), where two parties share only a human-memorable password, which is relatively short and easily guessed. It authenticates the parties even under the circumstance that the users have restricted computing and memory devices. The password-based authenticated key exchange scheme secure against the dictionary attack was proposed by Bellovin and Merrit[2]. Since then, such schemes have been extensively studied. We here mention some of the attempts;

© Springer-Verlag Berlin Heidelberg 2005

the following are not exhaustive at all. Bellare et al. [3] and Boyko et al. [8] proposed a formal model and security goals for password-based authenticated key exchange schemes. Katz et al. [13] and Gennaro et al. [12] showed a PAKE schemes that has provable security in the standard model. Recently, Bresson et al. constructed an efficient PAKE with provable security in the random oracle model [5]; Abdalla et al. showed a simple PAKE schemes that are relying little on random oracles [1].

In both approaches above, the communicating parties have to share secret information before communication starts. The difference between these two approaches is the quality of the secret keys. On one hand, keys in the first scheme are long and randomly chosen from the uniformly distributed binary sequences of a fixed length. On the other hand, the keys in the second have low entropy, that is, sequences that probably lie in a dictionary of feasible size for exhaustive search by a computer and easily guessed by an adversary. From the attacker's standpoint, the password-based scheme is vulnerable unless sufficient consideration is taken because it is quite easy to guess the password and check whether or not the guessed password is correct by data transmitted. However, it has an advantage that users do not have to hold any additional store device, whereas the users have to carry the store device containing the long random key for the first scheme.

## 1.2 Anonymous Authentication Scheme

An *anonymous authentication scheme* (AA) [6, 14, 10] (or anonymous group identification scheme) is a protocol that allows a member called a prover of a group $\Gamma$ to convince a verifier that she is a member of $\Gamma$ without revealing any information about her. So this is an interactive proof; the protocol consists of two parties, the prover and the verifier, and the prover convinces the verifier that she knows a secret without revealing itself. We should note that no outside adversary is considered in the security of this scheme. The security model for this is different from that of (password-based) authenticated key exchange schemes in which the typical adversary is not a protocol participant. Thus it is not trivial to establish and authenticate a common session key using an anonymous authentication scheme. We also note that secret keys are long and random in the all previous attempts [6, 14, 10] along this research. So it seems interesting to construct a password-based anonymous authentication scheme, where the secret kept by the users has low entropy such as a short word in a dictionary. Such scheme has not been proposed so far as far as we know.

## 1.3 Our Contribution

All previous PAKE schemes allow two parties to authenticate each other, but their identities turn to be revealed. This should be inconvenient in the case that users desire to hide their identities although a secure authenticated session key with a server is established. In this context, the authentication means to verify that the entity whom the server communicates belongs to a predetermined legitimate group of users. In this paper, we propose a new model of PAKE called an *anonymous password-based authenticated key exchange* (APAKE). We require APAKE to be secure against the *offline dictionary attack* as it is required to PAKE. The key idea in the construction of APAKE is to embed an oblivious transfer protocol (OT) into a two-party PAKE scheme. We follow the security model in [3, 5] and analyze the security of our proposed schemes in random oracle model.

We also extend this to a scheme that allows any subgroup of at least $k$-out-of-$n$ users of a group $\Gamma$ to generate a session key with the server in password-based and anonymous way.

Our scheme attains the anonymity under the assumption that the server follows the protocol. As a matter of fact, there exists an active attack by the server to detect the user identity. However, the server has to take a risk of revealing his dishonest behavior. Such attack can distinguish whether the user belong to some subgroup $\Gamma' \subset \Gamma$ or not with success probability $\frac{k}{n}$ while the probability of disclosure of the server's dishonest probability is $\frac{n-k}{n}$, where $n$ and $k$ are the number of the users in $\Gamma$ and $\Gamma'$ respectively.

**Table 1.** Compare with related schemes

| Schemes | Example | Type of secret data | Protocol goal | Security attained/ adversary |
|---|---|---|---|---|
| AKE | [4, 11] | Random bit sequence | KE | Confidential/outsider Authentication/outsider |
| PAKE | [1, 3, 5, 8] [12, 13] | Password (low entropy bit sequence) | KE | Confidential/outsider Authentication/outsider |
| AA | [6, 14, 10] | Random bit sequence | ID | Authentication/outsider Anonymity/server |
| APAKE | Proposal in this paper | Password (low entropy bit sequence) | KE | Confidential/outsider Authentication/outsider Anonymity/server |

Here, KE and ID stand for key exchange and identification (entity authentication), respectively.

## 1.4 Application

To justify our proposal of APAKE, we show a plausible application. Suppose that a business company has the set of employee $\{C_1, \ldots, C_n\}$. Every $C_i$ has a secret password to access to the company's network. In the company, everyone is allowed to report her or his personal opinion on any issues on their activities to the president of the company in order to improve their performance or working environment. Probably, such a report should be confidential occasionally. On the other hand, one wishes to hide their identity when reporting something that the president feels unpleasant. In such a case, APAKE can be used to establish a session key for later use of encrypted communication in a password-based and anonymous way. Establishing a session key, the employee $C_i$ and the president can have encrypted communication even though the president is sure that he is communicating with one of $\{C_1, \ldots, C_n\}$ but does not know whom the president is communicating. In this model, we do not have to construct any extra anonymous transmission channel other than providing a password to each employee. In this application, we need a password-based key exchange scheme and the anonymity property and APAKE provides both of the properties.

## 2 Model and Definitions

In this section, we formalize the protocol goals, the security of the protocol and attacker's ability.

### 2.1 Model and Protocol Goals

**Participants.** APAKE protocol involves a server $S$ and a group $\Gamma$ of $n$ users $C_1, \ldots, C_n$, that is, $\Gamma = \{C_1, \ldots, C_n\}$. We call $\{S, \Gamma\}$ a set of participants of APAKE.

**Protocol goal.** When user $C_i$ wish to have communication with $S$, $C_i$ first establish a session key with $S$ for later encrypted communication. Moreover, $C_i$ wants to keep his privacy from $S$ and anybody else. APAKE provides such a technique; it allows $C_i$ to establish a session key with $S$ without giving any information on $C_i$'s identity to $S$ and anybody else. After the protocol, a user in the legitimate group can establish a session key with the server and nobody other than the user knows with whom the server agrees the session key. We also consider a protocol where at least $k$ users in the group can establish a session key with the server but less than $k$ users cannot.

**Long-lived key.** Each user $C_i \in \Gamma$ holds a low entropy password $pw_i$ that is picked independently each other from a dictionary $\mathcal{D}$. Server $S$ holds a list $pw_S = \{pw_i\}_{C_i \in \Gamma}$. We call $pw_i$ and $pw_S$ the *long-lived keys* of user $C_i$ and the server $S$, respectively. The typical size of a dictionary $\mathcal{D}$ is about $2^{30}$ and so a password may be short and guessed easily.

**Adversary.** In our scheme, the security is twofold; confidentiality of the session key and authenticity of user and server, and anonymity and unlinkability of the user. The adversary varies corresponding to the security considered. When considering the confidentiality of the session key and authenticity of user and server, the adversary is somebody who is not a legal participant of the protocol. On the other hand, the adversary to break the anonymity and unlinkability may be a server even though the server is a legal participant.

### 2.2 Security Notions

Our scheme should satisfy both security properties of a password-based authenticated key exchange and the anonymity/unlinkability of an anonymous authentication scheme. Basically our scheme attains the same security level as these two schemes except for that the anonymity for our scheme is guaranteed only against the passive attacks of a server; we assume the server follows the protocol and does not deviate it.

First, we denote the adversary against confidentiality of the session key and authenticity of user and server as $\mathcal{A}$. We shall explain the act of $\mathcal{A}$ below.

**Executing the protocol.** Let $C_i \in \varGamma$ and $S$ be two participants that participate in APAKE protocol $P$. Each of them may have several instances called oracles involved in distinct, possibly concurrent, executions of $P$. Generally, we denote the instance $\rho$ of participant $U$ by $\varPi_U^\rho$.

An adversary $\mathcal{A}$ is a probabilistic algorithm with a distinguished query tape. The capability of $\mathcal{A}$ is modeled by queries he can ask to the oracles, representing the actions of $\mathcal{A}$ over the interaction among the protocol participants. The oracle queries made by $\mathcal{A}$ are formally defined in [3]. See [3] for more details.

- $\mathsf{Execute}(C_i, \rho, S, \delta)$-This query models passive attack, where the adversary gets access to honest executions of $P$ between the instances $\varPi_{C_i}^\rho$ and $\varPi_S^\delta$ by eavesdropping.
- $\mathsf{Reveal}(U, \rho)$-This query models the misuse of the session key by instance $\varPi_U^\rho$ (known-key attacks). The query is only available to $\mathcal{A}$ if the attacked instance actually "holds" a session key and it releases the latter to $\mathcal{A}$.
- $\mathsf{Send}(U, \rho, m)$-This query enables to consider active attacks by having $\mathcal{A}$ sending a message to instance $\varPi_U^\rho$. The adversary $\mathcal{A}$ gets back the response $\varPi_U^\rho$ generates in processing the message $m$ according to the protocol $P$. A query $\mathsf{Send}(C_i, \rho, \mathsf{Start})$ initializes $P$, and thus the adversary receives the initial flow the user instance $\varPi_{C_i}^\rho$ should send out to the server $S$.

**AKE security.** The confidentiality (semantic security) of the session key is evaluated by how much information on the key is leaked to the adversary. If no information is leaked, or information leaked is negligible in polynomial time computation, the adversary cannot distinguish the session key and a random bit sequence by polynomial time computation. On the other hand, if information is leaked, then the probability that an adversary can determine whether a given bit sequence is a session key processed by the APAKE protocol or a randomly and uniformly chosen bit sequence is substantially larger than $\frac{1}{2}$.

An adversary $\mathcal{A}$ is allowed to call oracles Execute and Send during an execution of the APAKE protocol $P$. Eventually, $\mathcal{A}$ calls $\mathsf{Test}(U, \rho)$-query only one time, for some instance $\varPi_U^\rho$. The $\mathsf{Test}(U, \rho)$-query is only available to $\mathcal{A}$ if attacked instance $\varPi_U^\rho$ is **Fresh** (which roughly means that the session key that $\varPi_U^\rho$ hold is not "obviously" known to the adversary.) The Test oracle tosses a coin and obtains a bit $b \in \{0, 1\}$. If $b = 0$, then Test gives a random bit sequence, and if $b = 1$, then Test gives a session key (the output of $\mathsf{Reveal}(U, \rho)$). Receiving the output of Test, an adversary $\mathcal{A}$ is a probabilistic algorithm and outputs $b'$, which represents $\mathcal{A}$'s guess of $b$. An *ake advantage* is the double of the probability that $\mathcal{A}$ correctly guesses the value of $b$ minus 1, that is,

$$\mathsf{Adv}_{P,\mathcal{D}}^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1,$$

where the probability is taken over all the random coins of the adversary and all the oracles and the passwords are picked from a dictionary $\mathcal{D}$. The protocol $P$ is said to be $(t, \epsilon)$-*AKE-secure* if $\mathcal{A}$'s advantage is smaller than $\epsilon$ for any adversary $\mathcal{A}$ running in time $t$.

**Authentication.** Another goal of the adversary $\mathcal{A}$ is to impersonate a user or the server. We denote the probability that $\mathcal{A}$ successfully impersonates a sever instance (resp. user instance) in an execution of $P$ by $\mathsf{Succ}_P^{\mathsf{S-auth}}(\mathcal{A})$ (resp. $\mathsf{Succ}_P^{\mathsf{C-auth}}(\mathcal{A})$). Impersonation succeeds when a user (resp. the server) accepts a session key which is shared with no instance of the server (resp. the user in $\Gamma$), that is,
$\mathsf{Succ}_P^{\mathsf{S-auth}}(\mathcal{A}) = \Pr[C \text{ accept a key with no instance of } S]$
$\mathsf{Succ}_P^{\mathsf{C-auth}}(\mathcal{A}) = \Pr[S \text{ accept a key with no instance of } C_i \in \Gamma]$

In this paper we consider unilateral authentication of the server ($\mathsf{S\text{-}auth}$). The protocol $P$ is said to be $(t, \epsilon)$-*S-auth-secure* if $\mathcal{A}$'s success probability for breaking $\mathsf{S\text{-}auth}$ is smaller than $\epsilon$ for any adversary $\mathcal{A}$ running in time $t$.

Here, we explain the anonymity and unlinkability of the user against $S$.

**Anonymity.** A protocol is *anonymous* if no information about the user's identity is revealed, whereas the user can establish a session key with the server. We shall show that any user $C_i$ gives no information to $S$ except that he is a member of $\Gamma$. We consider only a passive attack by the server $S$, that is, we assume that $S$ does not deviate the protocol. Let $P(C_i, S)$ be a transaction of $\mathsf{APAKE}$ run between user $C_i$ in $\Gamma$ and $S$. Let $\Pi_i$ a probability space of $P(C_i, S)$. We say that our $\mathsf{APAKE}$ achieves anonymity if for any two user $C_i, C_j \in \Gamma$, we have

$$\Pi_i = \Pi_j.$$

This is different from the setting in [10], where the verifier can send any challenge as long as it is allowed. If the verifier sends an unlawful challenge, then the prover can detect it in the scheme [10]. On the other hand, the server, who can be regarded as a verifier, has to follow the protocol to make a challenge in our scheme. As we discuss in Section 5, there is an active attack to identify $C_i$ at the risk of losing the server's qualification. Such attack can distinguish whether $C_i$ belong to some subgroup $\Gamma' \subset \Gamma$ or not with success probability $\frac{k}{n}$ while the probability of disclosure of the server's dishonest behaviors is $\frac{n-k}{n}$, where $n$ and $k$ are the number of the users in $\Gamma$ and $\Gamma'$ respectively.

**Unlinkability.** A protocol achieves *unlinkability* if $S$ cannot determine whether two key agreement transactions are made by a single user.

## 3 APAKE Scheme

### 3.1 Preparation

**Computational Diffie-Hellman assumption.** Suppose that $\mathbb{G} = \langle g \rangle$ is a cyclic group of prime order $p$. A $(t, \epsilon)$-$\mathsf{CDH}_{g,\mathbb{G}}$ attacker in $\mathbb{G}$ is a probabilistic algorithm $\varDelta$ running in time $t$ to compute $g^{xy}$ for given $g^x$ and $g^y$ such that the success probability $\mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}}(\varDelta)$ is greater than $\epsilon$. We denote the maximal success probability over all adversaries running in time $t$ by $\mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}}(t)$ . The $\mathsf{CDH}$-assumption is the assumption that $\mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}}(t) \leq \epsilon$.

**Oblivious transfer.** A 1-out-of-$n$ OT is a protocol where a sender $S$ has $n$ secret strings and a chooser $C$ can obtain only one of them without revealing his choice of index. In [15], Tzeng gives an efficient 1-out-of-$n$ OT that is secure in the random oracle model. We briefly explain Tzeng's protocol. Suppose that $S$ has $n$ secret strings $\alpha_1, \alpha_2, \ldots, \alpha_n$ of size $l_1$ and $C$ wants to obtain the string of index $i$. Let $g$ and $h$ be generators of a cyclic group $\mathbb{G}$, and $\mathcal{H}$ be a hash function from $\{0,1\}^*$ to $\{0,1\}^{l_1}$ which is considered as a random oracle.

1. $C$ chooses uniformly and randomly $r \in \mathbb{Z}_p$ and computes $Q(i) = g^r h^i$, and sends query $Q(i)$ to $S$.
2. For every $1 \leq j \leq n$, $S$ chooses uniformly and randomly $k_j \in \mathbb{Z}_p$ and computes $\beta_j = \mathcal{H}((Q(i)(h^j)^{-1})^{k_j}, j) \oplus \alpha_j$ and sends the answer $A(Q(i)) = \{(\beta_1, g^{k_1}), \ldots, (\beta_n, g^{k_n})\}$ to $C$ ($\oplus$ denotes the bitwise exclusive-or).
3. $C$ extracts $(\beta_i, g^{k_i})$ from $A(Q(i))$ and obtains $\alpha_i$ as $\alpha_i = \beta_i \oplus \mathcal{H}((g^{k_i})^r, i)$.

## 3.2 Proposed Scheme

Let $\mathbb{G} = \langle g \rangle$ be a finite cyclic group whose order is an $l$-bit prime number $p$ and every element of $\mathbb{G}$ is represented by a bit string of size $l_1$. The operation in $\mathbb{G}$ is written multiplicatively. Let $g, h$ are generators of $\mathbb{G}$; assume that the discrete logarithm $\log_g h$ is unknown to all. Let $\mathcal{F}, \mathcal{G}, \mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2$ be a random hash functions from $\{0,1\}^*$ to $\{0,1\}^{l_1}$. So we assume these are random oracles. Let $\varGamma$ be a user-group (or simply group) of $n$ users $\{C_1, \ldots, C_n\}$. Each user $C_i$ in $\varGamma$ is initially provided a distinct low entropy password $pw_i$, while $S$ holds a list of these passwords. We set $\mathsf{PWF}_i = \mathcal{F}(pw_i)$ and $\mathsf{PWG}_i = \mathcal{G}(i, pw_i)$.

All $pw_i$ are independently picked from the dictionary $\mathcal{D}$ according to the distribution $\mathcal{D}_{pw}$. We use the notation $\mathcal{D}_{pw}(\tau)$ for the probability to be in the most probable set of $\tau$ passwords:

$$\mathcal{D}_{pw}(\tau) = \max_{\mathcal{P} \subseteq \mathcal{D}} \left\{ \Pr_{pw \in \mathcal{D}_{pw}} [pw \in \mathcal{P} | \sharp\mathcal{P} \leq \tau] \right\}$$

Note that if we denote by $\mathcal{U}_N$ the uniform distribution among $N$ passwords, $\mathcal{U}_N(\tau) = \tau/N$. We first provide a high-level description of our protocol:

### Outline of the scheme

1. $C_i$ chooses uniformly and randomly a secret exponent $x$ and sends the corresponding Diffie-Hellman public key $g^x$ to $S$.
2. $S$ chooses uniformly and randomly a secret exponent $y$. Next, $S$ prepares $n$ elements $g^y g^{\mathcal{F}(pw_i)}$ for $1 \leq i \leq n$, where $\mathcal{F}$ is a random oracle.
3. $C_i$ makes an (1-out-of-$n$) OT query in order to get $g^y g^{\mathcal{F}(pw_i)}$ from $S$.

4. $S$ answers the query in a manner that $C_i$ will retrieve only $g^y g^{\mathcal{F}(pw_i)}$ and then $g^y$.
5. $C_i$ and $S$ compute the common Diffie-Hellman secret value $g^{xy}$. $C_i$ and $S$ will use this $g^{xy}$ to generate the common section key and authenticate each other.

**Detailed description.** As mentioned in Sec.1.3, we construct APAKE by embed an OT protocol into a two-party PAKE scheme. In fact, we can use any OT to our protocol, but as we design a PAKE that works on a finite cyclic group $\mathbb{G}$ so we also apply an OT that works on $\mathbb{G}$ for convenient. Here, we will use the OT of Tzeng in [15] which is simple and designed for elements in $\mathbb{G}$. Our protocol is run between user $C_i$ and the server $S$. The pair $(Q(i), A(Q(i)))$ in our protocol is an OT run by $C_i$ and $S$. See Figure 1 for the protocol flow.

**Phase 1.**

1. $C_i$ chooses randomly and uniformly $x, r \in \mathbb{Z}_p$ and computes $X = g^x$. Next, $C_i$ generates a query $Q(i)$ for the $i$-th data in OT protocol as $Q(i) = g^r h^{\mathcal{G}(i,pw_i)} = g^r h^{\mathsf{PWG}_i}$.
2. $C_i$ sends $(\Gamma, X, Q(i))$ to $S$.

**Phase 2.**

1. $S$ chooses randomly and uniformly $y, k_1, \ldots, k_n \in \mathbb{Z}_p$ and computes $Y = g^y$ and $\alpha_i, \beta_j$ for $1 \le j \le n$ as follows
$\alpha_j = Y g^{\mathcal{F}(pw_j)} = Y g^{\mathsf{PWF}_j}, \quad \beta_j = \mathcal{H}_0\Big( \big(Q(i)(h^{\mathsf{PWG}_j})^{-1}\big)^{k_j}, j\Big) \oplus \alpha_j.$
2. Let $A(Q(i)) = (\beta_1, \ldots, \beta_n, g^{k_1}, \ldots, g^{k_n})$, and let $K_S = X^y$.

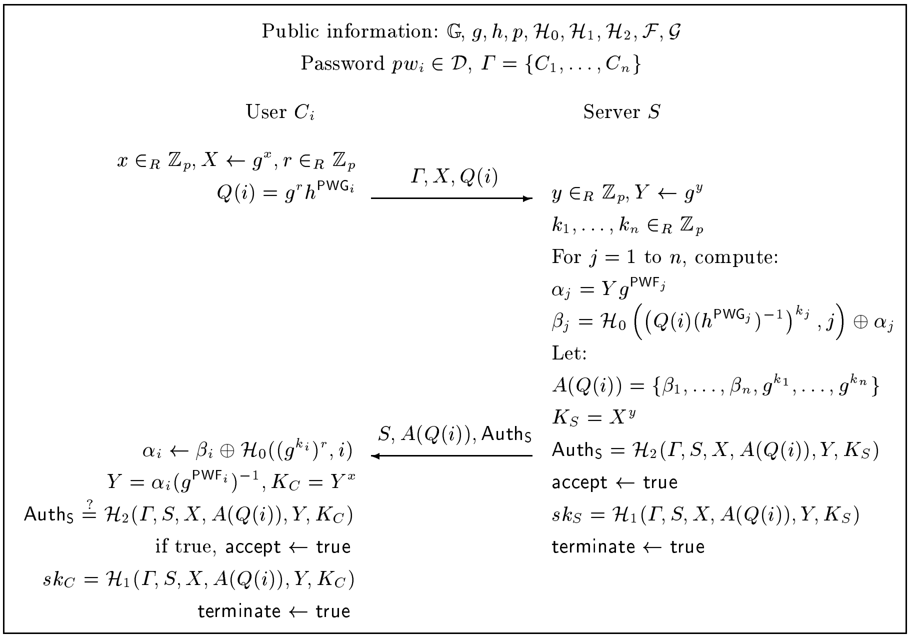

Public information: $\mathbb{G}, g, h, p, \mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2, \mathcal{F}, \mathcal{G}$
Password $pw_i \in \mathcal{D}$, $\Gamma = \{C_1, \ldots, C_n\}$
User $C_i$
Server $S$
$x \in_R \mathbb{Z}_p, X \leftarrow g^x, r \in_R \mathbb{Z}_p$
$Q(i) = g^r h^{\mathsf{PWG}_i}$
$\xrightarrow{\Gamma, X, Q(i)}$
$y \in_R \mathbb{Z}_p, Y \leftarrow g^y$
$k_1, \ldots, k_n \in_R \mathbb{Z}_p$
For $j = 1$ to $n$, compute:
$\alpha_j = Y g^{\mathsf{PWF}_j}$
$\beta_j = \mathcal{H}_0\Big(\big(Q(i)(h^{\mathsf{PWG}_j})^{-1}\big)^{k_j}, j\Big) \oplus \alpha_j$
Let:
$A(Q(i)) = \{\beta_1, \ldots, \beta_n, g^{k_1}, \ldots, g^{k_n}\}$
$K_S = X^y$
$\mathsf{Auth}_\mathsf{S} = \mathcal{H}_2(\Gamma, S, X, A(Q(i)), Y, K_S)$
accept $\leftarrow$ true
$sk_S = \mathcal{H}_1(\Gamma, S, X, A(Q(i)), Y, K_S)$
terminate $\leftarrow$ true
$\xleftarrow{S, A(Q(i)), \mathsf{Auth}_\mathsf{S}}$
$\alpha_i \leftarrow \beta_i \oplus \mathcal{H}_0((g^{k_i})^r, i)$
$Y = \alpha_i (g^{\mathsf{PWF}_i})^{-1}, K_C = Y^x$
$\mathsf{Auth}_\mathsf{S} \overset{?}{=} \mathcal{H}_2(\Gamma, S, X, A(Q(i)), Y, K_C)$
if true, accept $\leftarrow$ true
$sk_C = \mathcal{H}_1(\Gamma, S, X, A(Q(i)), Y, K_C)$
terminate $\leftarrow$ true

**Fig. 1.** Our APAKE with server-authentication

3. $S$ computes the authenticator $\mathsf{Auth_S}$ and the session key $sk_S$ as follows $\mathsf{Auth_S} = \mathcal{H}_2(\Gamma, S, X, A(Q(i)), Y, K_S)$ and $sk_S = \mathcal{H}_1(\Gamma, S, X, A(Q(i)), Y, K_S)$.
4. $S$ sends $(S, A(Q(i)), \mathsf{Auth_S})$ to $C_i$.

**Phase 3.**

1. $C_i$ extracts $\alpha_i$ from $A(Q(i))$ as $\alpha_i = \beta_i \oplus \mathcal{H}_0((g^{k_i})^r, i)$.
2. $C_i$ computes $Y = \alpha_i (g^{\mathsf{PWF}_i})^{-1}$, $K_C = Y^x$.
3. $C_i$ computes $\mathsf{Auth_C} = \mathcal{H}_2(\Gamma, S, X, A(Q(i)), Y, K_C)$ and checks whether $\mathsf{Auth_S} \overset{?}{=} \mathsf{Auth_C}$
4. If $\mathsf{Auth_S}$ is valid, $C_i$ accepts and computes the session-key $sk_C$ as $sk_C = \mathcal{H}_1(\Gamma, S, X, A(Q(i)), Y, K_C)$. If $\mathsf{Auth_S}$ is invalid then $C_i$ aborts the protocol.

*Remark:* In Phase 1 above, only person who knows $pw_i$ can make a correct query for $g^y g^{\mathcal{F}(pw_i)}$. Thus, an attacker trying to make an on-line attack cannot succeed in guessing the correct password $pw_i$.

## 3.3 Security

**AKE security/Authentication.** We provide a theorem claiming that the $\mathsf{APAKE}$ protocol satisfies AKE security and the unilateral authentication of the server $S$.

**Theorem 1. (AKE/S-Auth Security).** *Suppose that* $\mathsf{APAKE}$ *protocol is run employing a group of a prime order* $p$ *and a dictionary* $\mathcal{D}$ *equipped with the distribution* $\mathcal{D}_{pw}$. *For any adversary* $\mathcal{A}$ *with a time bound* $t$, *with less than* $q_{se}$ $\mathsf{Send}$*-queries,* $q_{ex}$ $\mathsf{Execute}$*-queries, and* $q_g$ *and* $q_h$ *hash queries to* $\{\mathcal{G}, \mathcal{F}\}$ *and* $\{\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2\}$, *respectively, we have*

$$\mathsf{Adv}^{\mathsf{ake}}_{\mathsf{apake}}(\mathcal{A}) \le \frac{2q_{se}}{2^{l_1}} + 12 \times \mathcal{D}_{pw}(q_{se}) + 12q_h^2 \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}}(t + 3\tau_e) + \frac{4T^2}{p},$$

$$\mathsf{Succ}^{\mathsf{S-auth}}_{\mathsf{apake}}(\mathcal{A}) \le \frac{q_{se}}{2^{l_1}} + 3 \times \mathcal{D}_{pw}(q_{se}) + 3q_h^2 \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}}(t + 3\tau_e) + \frac{T^2}{p},$$

*where* $T = q_{se} + q_q + q_g$, *and* $\tau_e$ *is the computational time for an exponentiation in* $\mathbb{G}$.

The proof of Theorem 1 is similar to the one in [5] and we give it in full version of this paper due to lack of space. We here give only the outline. Before the outline of the proof, we note that this theorem shows that the protocol is secure against dictionary attacks since the advantage of the adversary essentially grows with the ratio of interactions (number of $\mathsf{Send}$-queries) to the number of passwords.

*Outline of Proof.* Here, we show an intuitive idea behind the proof. As we mentioned, the proof is similar to the one in [5], however, there is a clear difference between the protocol of Bresson et al.[5] and ours. We use an oblivious transfer transaction $(Q(i), A(Q(i)))$ between $C_i$ and $S$ in our protocol. This additional information just securely transmits $Y$ to $C_i$ while it does not reveal any information that may be used to expose the user identity, password or session key:

- We see that $Q(i)$ is uniformly distributed because $r$ is chosen uniformly and randomly. Therefore, $Q(i)$ does not reveal any information about $C_i$'s identity and $pw_i$.
- Due to privacy of $S$ guaranteed by the OT protocol, the user $C_i$ can learn only $\alpha_i (1 \leq i \leq n)$ from the transaction. As we embedded the password $pw_i$ in $Q(i)$ and $\beta_j (1 \leq j \leq n)$ as

  $$Q(i) = g^r h^{\mathsf{PWG}_i} = g^r h^{\mathcal{G}(i, pw_i)} \quad \text{and} \quad \beta_j = \mathcal{H}_0 \left( \left( Q(i)(h^{\mathsf{PWG}_j})^{-1} \right)^{k_j}, j \right) \oplus \alpha_j,$$

  only $C_i$, who knows $pw_i$, can learn $\alpha_i$. In other words, nobody other than $C_i$ obtains any meaningful information from $A(Q(i))$.
- Moreover, $C_i$ does not obtain any information on any $\alpha_j (i \neq j)$ and so he cannot mount the dictionary attack to obtain the other users' password.

As a result, the information that $S$ and $C_i$ can use is almost similar to that in the protocol [5]. Therefore, we obtain a similar security result to [5]. Q.E.D.

**Anonymity/Unlinkability**

**Theorem 2.** *The anonymity and unlinkability of* APAKE *protocol is unconditional.*

*Proof.* We see that the triple $(\Gamma, X = g^x, Q(i))$ in any transaction of APAKE protocol made by any user $C_i$ is uniformly distributed over the set $\{\Gamma\} \times \mathbb{G} \times \mathbb{G}$ because $x$ and $r$ are chosen uniformly and randomly from $\mathbb{Z}_p$. Therefore, nobody distinguishes a transaction of $C_i$ and a transaction of $C_j$ $(i \neq j)$. This implies that APAKE unconditionally satisfies the anonymity, that is, the anonymity does not depend on any computational assumption. Similarly, we can show the unlinkability. Q.E.D.

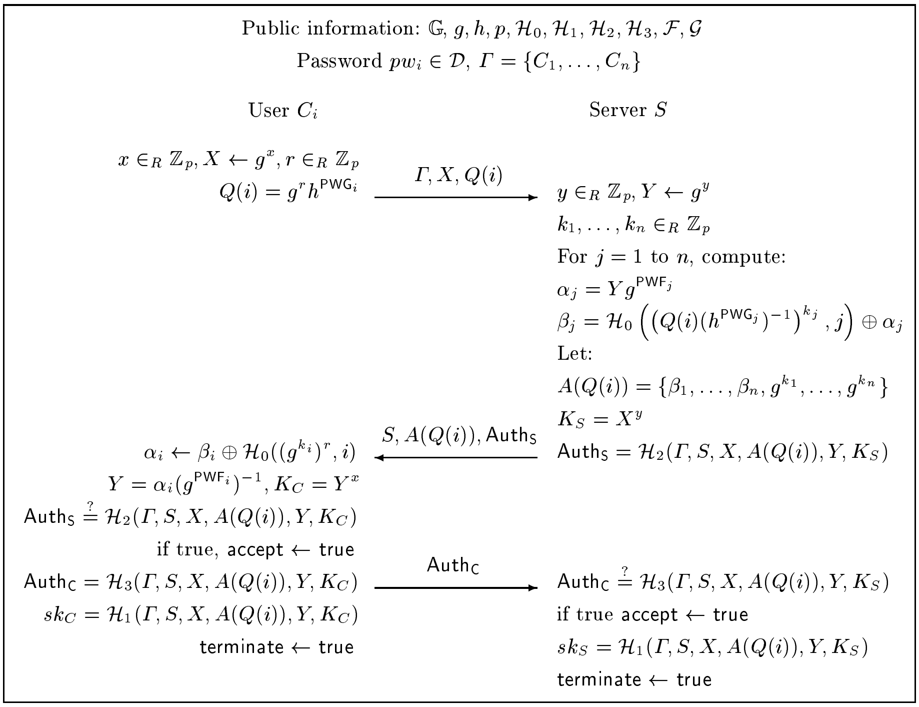
## 3.4 Mutually Authentication

In addition to **Phase 1**, **Phase 2** and **Phase 3**, we run **Phase 4** in order to authenticate the users without revealing the user's identity. In this mode, we assume $S$ behave honestly.

**Phase 4.**

1. $C_i$ computes $\mathsf{Auth_C} = \mathcal{H}_3(\Gamma, S, X, A(Q(i)), Y, K_C)$, where $\mathcal{H}_3$ is a random hash function of $\{0,1\}^*$ to $\{0,1\}^{l_1}$. $C_i$ sends $\mathsf{Auth}_C$ to $S$.
2. $S$ checks whether $\mathsf{Auth_C} = \mathcal{H}_3(\Gamma, S, X, A(Q(i)), Y, K_S)$. If $\mathsf{Auth_C}$ is valid then $S$ accepts and computes the session key as $sk_S{=}\mathcal{H}_1(\Gamma, S, X, A(Q(i)), Y, K_S)$.

The APAKE with mutual authentication is illustrated in Fig.2.



Public information: $\mathbb{G}, g, h, p, \mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{F}, \mathcal{G}$

Password $pw_i \in \mathcal{D}$, $\Gamma = \{C_1, \ldots, C_n\}$

| User $C_i$ | | Server $S$ |
|---|---|---|
| $x \in_R \mathbb{Z}_p, X \leftarrow g^x, r \in_R \mathbb{Z}_p$<br>$Q(i) = g^r h^{\mathsf{PWG}_i}$ | $\xrightarrow{\Gamma, X, Q(i)}$ | $y \in_R \mathbb{Z}_p, Y \leftarrow g^y$<br>$k_1, \ldots, k_n \in_R \mathbb{Z}_p$<br>For $j = 1$ to $n$, compute:<br>$\alpha_j = Y g^{\mathsf{PWF}_j}$<br>$\beta_j = \mathcal{H}_0\left(\left(Q(i)(h^{\mathsf{PWG}_j})^{-1}\right)^{k_j}, j\right) \oplus \alpha_j$<br>Let:<br>$A(Q(i)) = \{\beta_1, \ldots, \beta_n, g^{k_1}, \ldots, g^{k_n}\}$<br>$K_S = X^y$ |
| $\alpha_i \leftarrow \beta_i \oplus \mathcal{H}_0((g^{k_i})^r, i)$<br>$Y = \alpha_i (g^{\mathsf{PWF}_i})^{-1}, K_C = Y^x$<br>$\mathsf{Auth_S} \stackrel{?}{=} \mathcal{H}_2(\Gamma, S, X, A(Q(i)), Y, K_C)$<br>if true, accept $\leftarrow$ true | $\xleftarrow{S, A(Q(i)), \mathsf{Auth_S}}$ | $\mathsf{Auth_S} = \mathcal{H}_2(\Gamma, S, X, A(Q(i)), Y, K_S)$ |
| $\mathsf{Auth_C} = \mathcal{H}_3(\Gamma, S, X, A(Q(i)), Y, K_C)$<br>$sk_C = \mathcal{H}_1(\Gamma, S, X, A(Q(i)), Y, K_C)$<br>terminate $\leftarrow$ true | $\xrightarrow{\mathsf{Auth_C}}$ | $\mathsf{Auth_C} \stackrel{?}{=} \mathcal{H}_3(\Gamma, S, X, A(Q(i)), Y, K_S)$<br>if true accept $\leftarrow$ true<br>$sk_S = \mathcal{H}_1(\Gamma, S, X, A(Q(i)), Y, K_S)$<br>terminate $\leftarrow$ true |

**Fig. 2.** APAKE with mutual-authentication

## 4 *k*-out-of-*n* APAKE Scheme

### 4.1 Description

We shall extend the APAKE scheme in the previous section to the scheme that allows any subgroup of $\Gamma$ consisting of at least $k$ users ($k \leq n$) to generate a common session key with $S$ in password-based authenticated and anonymous way. Moreover, a subgroup consisting of less than $k$ users is unable to generate a common session key with $S$. We employ a $k$-out-of-$n$ OT in [9] but we omit the description of the scheme for lack of space. See Figure 3 for the protocol flow. Let $C$ be a subgroup of $\Gamma$ consisting of $k$ users. We may assume $C = \{C_1, \ldots, C_k\}$ without loss of generality.

**Phase 1.**

1. $C$ chooses randomly and uniformly $x \in \mathbb{Z}_p$ and computes $X = g^x$. Next, each $C_i$ ($1 \leq i \leq k$) chooses randomly and uniformly $\gamma_i, a_i \in \mathbb{Z}_p$. These $\gamma_i$ and $a_i$ are keep secret by $C_i$. Then, each $C_i (1 \leq i \leq k)$ computes $w_i = h^{\gamma_i \mathcal{G}(i, pw_i)}$ and $A_i = w_i g^{a_i}$. Let $Q = (A_1, \ldots, A_k)$.
2. $C$ sends $(\Gamma, X, Q)$ to $S$.

**Phase 2.**

1. $S$ chooses randomly and uniformly $y \in \mathbb{Z}_p$ and computes $Y = g^y$.
2. $S$ shares $y$ using Shamir's secret sharing scheme by choosing randomly and uniformly $b_1, \ldots, b_k \in \mathbb{Z}_p$, and sets $f(x) = y + b_1 x + \cdots + b_k x^k$ and

$y_i = f(i)$ for $1 \leq i \leq k$. Note that we have $y = \lambda_1 f(1) + \cdots + \lambda_k f(k)$, where $\lambda_i = \Pi_{1 \leq j \leq k, j \neq i} \frac{j}{j-i}$.
3. $S$ computes $Y_i = g^{y_i}$ for $1 \leq i \leq n$.
4. $S$ chooses randomly and uniformly $z \in \mathbb{Z}_p$ and computes $Z = g^z$.
5. $S$ computes $D_i = A_i^z$ for $1 \leq i \leq k$ and $\alpha_i = Y_i g^{\mathcal{F}(pw_i)} \oplus \mathcal{H}_0(w_i^z)$ for $1 \leq i \leq n$. Let $A(Q) = (Z, D_1, \ldots, D_k, \alpha_1, \ldots, \alpha_n),\ K_S = X^y$
6. $S$ computes the authenticator $\mathsf{Auth_S}$ and the session key $sk_S$ as follows $\mathsf{Auth_S} = \mathcal{H}_2(\Gamma, S, X, A(Q), Y, K_S)$ and $sk_S = \mathcal{H}_1(\Gamma, S, X, A(Q), Y, K_S)$.
7. $S$ sends $(S, A(Q(i)), \mathsf{Auth_S})$ to $C_i$.

**Phase 3.**
1. For $1 \leq i \leq k$, $C$ extracts $Y_i$ from $A(Q)$ as $Y_i = [\alpha_i \oplus \mathcal{H}_0(D_i(Z^{a_i})^{-1})]\ (g^{\mathcal{F}(pw_i)})^{-1}$
2. $C$ computes $Y = Y_1^{\lambda_1} \cdots Y_k^{\lambda_k}$ and $K_C = Y^x$.
3. $C$ checks whether the authenticator of $S$ is valid or not by $\mathsf{Auth_S} \overset{?}{=} \mathsf{Auth_C} = \mathcal{H}_2(\Gamma, S, X, A(Q), Y, K_C)$
4. If $\mathsf{Auth_S}$ is valid, $C$ accepts and computes the section key $sk_C$ as $sk_C = \mathcal{H}_1(\Gamma, S, X, A(Q), Y, K_C)$.

### 4.2 Security

Any subgroup of at least $k$ users in the legitimate group can authenticate their qualification that they comprise at least $k$ users and establish a session key with the server, whereas any subset of less than $k$ users cannot establish or authenticate a session key with the server. In addition, we have to consider an adversary $\mathcal{A}$ that colludes with at most $k-1$ users in $C$. Security properties of $k$-out-$n$ APAKE is similar with scheme in Sec.3. Detailed explanations will be presented in the full version of the paper.

## 5 Discussion

We discuss active attack by $S$ in the scheme of Sec. 3.2. In the case that $S$ behaves honestly, we can extend our scheme to a mutually authenticated key exchange scheme by adding one more authenticate flow from $C_i$ to $S$ as illustrated Fig.2. Unfortunately, if $S$ is malicious and attempts to expose identity of $C_i$, then $S$ can perform an active attack as follows. After receiving $(\Gamma, X, Q(i))$ from a user, $S$ chooses some subset $\Gamma' = \{C_{i_1}, \ldots, C_{i_k}\} \subset \Gamma$ that he guest $C_i$ is belong to $(1 \leq k \leq n-1)$. $S$ then chooses two distinct random numbers $y_1, y_2 \in \mathbb{Z}_p$, computes $Y_1 = g^{y_1}, Y_2 = g^{y_2}$ and modify $\alpha_i$ as $\alpha_i = Y_1 g^{\mathsf{PWF}_i}$ for $C_i \in \Gamma'$ and $\alpha_j = Y_2 g^{\mathsf{PWF}_j}$ for $C_j \in \Gamma \setminus \Gamma'$. $S$ then set $K_S = X^{y_1}$ and computes $\mathsf{Auth_S} = H_2(\Gamma, S, X, A(Q(i)), Y_1, K_S)$. $S$ then sends $A(Q(i))$ and $\mathsf{Auth_S}$ to the user.

If $S$'s guess is correct, that is, the user $C_i$ is belong to $\Gamma'$, then $C_i$ accepts and sends back $\mathsf{Auth_C}$ to $S$. Otherwise, the user rejects $\mathsf{Auth_S}$. So, we can see that $S$ will succeed in with the probability $k/n$, while his malicious action is exposed with probability $(n-k)/n$. In the about attack, to earn more information about identity of $C_i$, $S$ should choose a smaller $k$, but in that case, his malicious action

Public information: $\mathbb{G}, g, h, p, \mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2, \mathcal{G}, \mathcal{F}$

Password $pw_i \in \mathcal{D}$, $\Gamma = \{C_1, \dots, C_n\}$

| Group $C = \{C_1, \dots, C_k\}$ | | Server $S$ |
|---|---|---|
| $x \in_R \mathbb{Z}_p, X \leftarrow g^x$ | | |
| for $i = 1$ to $n$, each $C_i$ chooses: | | |
| $\gamma_i \in_R \mathbb{Z}_p$ and $a_i \in_R \mathbb{Z}_p$ | | |
| $w_1 \leftarrow h^{\gamma_1 \mathcal{G}(1, pw_1)}, \dots, w_k \leftarrow h^{\gamma_k \mathcal{G}(k, pw_k)}$ | | |
| $A_1 \leftarrow w_1 g^{a_1}, \dots, A_k \leftarrow w_k g^{a_k}$ | | |
| $Q = A_1, \dots, A_k$ | $\xrightarrow{\Gamma, X, Q}$ | $y \in_R \mathbb{Z}_p, Y \leftarrow g^y$ |
| | | $b_1, \dots, b_k \in_R \mathbb{Z}_p$ |
| | | $f(x) = y + b_1 x + \cdots + b_k x^k$ |
| | | $y_1 \leftarrow f(1), \dots, y_n \leftarrow f(n)$ |
| | | $Y_1 = g^{y_1}, \dots, Y_n = g^{y_n}$ |
| | | $z \in_R \mathbb{Z}_p, Z = g^z$ |
| | | $D_1 = A_1^z, \dots, D_k = A_k^z$ |
| | | for $i = 1$ to $n$, computes: |
| | | $\alpha_i = Y_i g^{\mathcal{F}(pw_i)} \oplus \mathcal{H}_0(w_i^z)$ |
| | | $A(Q) = \{Z, D_1, \dots, D_k, \alpha_1, \dots, \alpha_n\}$ |
| | | $K_S = X^y$ |
| for $1 \leq i \leq k$, compute: | $\xleftarrow{S, A(Q), \mathsf{Auth_S}}$ | $\mathsf{Auth_S} = \mathcal{H}_2(\Gamma, S, X, A(Q), Y, K_S)$ |
| $Y_i \leftarrow [\alpha_i \oplus \mathcal{H}_0(D_i(Z^{a_i})^{-1})](g^{\mathcal{F}(pw_i)})^{-1}$ | | $\mathsf{accept} \leftarrow \mathsf{true}$ |
| $Y = Y_1^{\lambda_1} \cdots Y_k^{\lambda_k},\ K_C = Y^x$ | | $sk_S = \mathcal{H}_1(\Gamma, S, X, A(Q), Y, K_S)$ |
| where $\lambda_i = \Pi_{1 \leq j \leq k, j \neq i} \frac{j}{j-i}$ | | $\mathsf{terminate} \leftarrow \mathsf{true}$ |
| $\mathsf{Auth_S} \overset{?}{=} \mathcal{H}_2(\Gamma, S, X, A(Q), Y, K_C)$ | | |
| if true, $\mathsf{accept} \leftarrow \mathsf{true}$ | | |
| $sk_C = \mathcal{H}_1(\Gamma, S, X, A(Q), Y, K_C)$ | | |
| $\mathsf{terminate} \leftarrow \mathsf{true}$ | | |

**Fig. 3.** Our $k$-out-$n$ APAKE with server-authentication

will be exposed with larger probability. In a realistic scenario, the server would be reluctant to take a risk to disclose his dishonest behavior. To correct this problem will be our future work.

## References

1. M. Abdalla, and D. Pointcheval. "Simple Password-Based Encrypted Key Exchange Protocols". In *Proc. of CT-RSA '05*, LNCS 3376, pages 191–208. Springer Verlag, 2005.
2. S. M. Bellovin, and M. Merritt. "Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks". In *Proc. of the Symposium on Security and Privacy*, IEEE, pages.72–84, 1992.
3. M. Bellare, D. Pointcheval, and P. Rogaway. "Authenticated Key Exchange Secure Against Dictionary Attacks". In *Proc. of Eurocrypt '00*, LNCS 1807, pages 139–155. Springer Verlag, 2000.

4. M. Bellare, Ran. Canetti, and H. Krawczyk. "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols". In *STOC '98*, pages. 419–428. 1998.
5. E. Bresson, O. Chevassut, and D. Pointcheval. "New Security Results on Encrypted Key Exchange". In *Proc. of PKC '04*, LNCS 2947, pages 145–158. Springer Verlag, 2004.
6. D. Boneh and M. Franklin. "Anonymous authentication with subset queries". In *ACM CCS' 99*, pages 113–119. 1999.
7. C. Boyd and A. Mathuria. *Protocols for authentication and key establishment*, Springer-Verlag, 2003.
8. V. Boyko, P. MacKenzie, and S. Patel. "Provable secure password-authenticated key exchange using Diffie-Hellman". In *Proc. of Eurocrypt '00*, LNCS 1807, pages 156–171. Springer Verlag, 2000.
9. C. K. Chu and W. G. Tzeng. "Efficient $k$-out-of-$n$ Oblivious Transfer Schemes with Adaptive and Non-Adaptive Queries". In *Proc. of PKC '05*, LNCS 3386, pages 172–183. Springer Verlag, 2005.
10. A. De Santis, G. Di Crescenzo and G. Persiano. "Communication-efficient anonymous group identification". In *ACM CCS '98*, pages 73–82. 1998.
11. W. Diffie, P. C. van Oorschot, and M. J. Wiener. "New Authentication and Authenticated Key Exchanges". In *Designs, Cosdes and Cryptography*, Vol.2, No.2, pages 107–125. 1992.
12. R. Gennaro and Y. Lindell. "A framework for password-based authenticated key exchange". In *Proc. of Eurocrypt '03*, LNCS 2656, pages 524–543. Springer Verlag, 2003.
13. J. Katz, R. Ostrovsky, and M. Yung. "Efficient password-authenticated key exchange using human-memorable passwords". In *Proc. of Eurocrypt '01*, LNCS 2045, pages 475–494. Springer Verlag, 2001.
14. C. H. Lee, X. Deng, and H. Zhu. "Design and Security Analysis of Anonymous Group Identification Protocols". In *Proc. of PKC '02*, LNCS 2274, pages 188–198. Springer Verlag, 2002.
15. W. G. Tzeng. "Efficient 1-Out-$n$ Oblivious Transfer Schemes". In *Proc. of PKC '02*, LNCS 2274, pages 159–171. Springer-Verlag, 2002.

# Faster Pairings Using an Elliptic Curve with an Efficient Endomorphism

Michael Scott

School of Computing,
Dublin City University,
Ballymun, Dublin 9, Ireland
mike@computing.dcu.ie

**Abstract.** The most significant pairing-based cryptographic protocol to be proposed so far is undoubtedly the Identity-Based Encryption (IBE) protocol of Boneh and Franklin. In their paper [6] they give details of how their scheme might be implemented in practice on certain supersingular elliptic curves of prime characteristic. They also point out that the scheme could as easily be implemented on certain special non-supersingular curves for the same level of security. An obvious question to be answered is – which is most efficient? Motivated by the work of Gallant, Lambert and Vanstone [14] we demonstrate that, perhaps counter to intuition, certain ordinary curves closely related to the supersingular curves originally recommended by Boneh and Franklin, provide better performance. We illustrate our technique by implementing the fastest pairing algorithm to date (on elliptic curves over fields of prime characteristic) for contemporary levels of security, albeit on a rather particular class of curves. We also point out that many of the non-supersingular families of curves recently discovered and proposed for use in pairing-based cryptography can also benefit (to an extent) from the same technique.

**Keywords:** Tate pairing implementation, pairing-based cryptosystems.

## 1 Introduction

If it is to be successful in the long term, pairing-based cryptography needs efficient algorithms for the calculation of the Weil or Tate pairing. In his early text book Menezes [17] mentions an implementation of the Weil pairing which "reported running times of just a few minutes" on a SUN-2 SPARC-station. However this is more than a little unfair – at the time there was no real incentive to try and optimise the standard technique, based on Miller's algorithm [18], and for the cryptanalytic purpose for which the Weil pairing was being used, a few minutes was more than adequate to make the point. In practice the Tate pairing has some advantages over the Weil pairing in most contexts, as first pointed out by Frey, Müller and Rück [12].

However the development of protocols that require fast pairings has produced a series of improvements and tricks which have drastically reduced this running time down to just a few milliseconds.

© Springer-Verlag Berlin Heidelberg 2005

One target might be that the pairing calculation should take as long as an RSA decryption, for the same level of security, and as pointed out by Scott [21], this target has already almost been reached. However more improvements may be possible, and it is the purpose of this paper to illustrate a new method which can either produce a further speed-up of up to 20%, or half the amount of storage required, depending on the context in which the pairing is to be calculated.

The development of fast pairings has advanced on two fronts. The first has concentrated on optimising algorithms for the Tate pairing on elliptic curves of prime characteristic, both supersingular and ordinary. The second has focused on algorithms for supersingular curves of small characteristic, typically of characteristic 2 and 3. The former approach is epitomised by the work of Barreto, Kim, Lynn and Scott [2] and Galbraith, Harris and Soldera [13]. For an easy-to-read description of the so-called BKLS-GHS algorithm, with timings, see [21]. While the BKLS-GHS algorithm is also suitable for use over small characteristic curves, the work of Duursma and Lee [11] made it clear that a more efficient algorithm was possible in this context. This approach culminated in the work of Barreto, Galbraith, O'hEigeartaigh and Scott [1], which introduced the primitive $\eta_T$ pairing, and showed how the Tate pairing could be calculated from it using an iterative loop only half the size of that required by Duursma and Lee. They also raise the possibility that pairings over characteristic 2 supersingular hyperelliptic curves may also be competitive.

However comparing the two types of fast pairings is difficult, as it amounts to comparing the difficulty of the discrete logarithm problem in fields of prime characteristic, with that in fields of small characteristic (although we do know that the latter is easier than the former for the same size of field [9]). In our view this comparison has not been adequately experimentally investigated. However the most authoritative comparison that we have found is due to Lenstra [16]. From this, and using the timings from [1], it would appear that the $\eta_T$ approach may in fact be the fastest. However since there is still some concern that the discrete logarithm problem in fields of low characteristic may be even easier than we currently think, in this paper we will concentrate exclusively on the prime characteristic case. See [20] for a nice discussion of these issues.

It has been suggested that pairings might be speeded up by using a prime modulus $p$ of low Hamming weight [15]. This idea can be used with both supersingular and non-supersingular curves. However this also raises legitimate concerns about a possible lowering of discrete-logarithm security. We will not consider the use of a prime modulus of low Hamming weight here.

A critical parameter of any pairing implementation is the embedding degree, or "security multiplier", denoted $k$. For reasons of efficiency it is usually recommended that $k$ be even [2] . The security multiplier relates the size of the base field over which points on the elliptic curve are manipulated, with the discrete-logarithm security of the pairing. For example on a particular supersingular hyperelliptic curve of characteristic 2, a value of $k = 12$ is possible [1], and so a hyperelliptic curve over the field $\mathbb{F}_{2^{113}}$ would result in a pairing with the discrete logarithm security of a 12*113=1356 bit binary extension field, which by reference to [16] might be considered to be adequately secure. So a large security multiplier implies that we can work on an elliptic or hyperelliptic curves over a smaller base field, with efficiency advantages. However the advantage of a large security multiplier is perhaps not as great as one might think, as the major part of the pairing calculation involves manipulations over the extension field (of size 1356 bits in our example), rather than over the smaller base field.

In the case of prime characteristic fields, the use of a security multiplier of $k = 2$ has proven to be surprisingly efficient for contemporary levels of security [21]. Note that for supersingular elliptic curves over fields of prime characteristic, $k = 2$ is the maximum possible. The issue of how to scale security in pairing based protocols has been considered by both Koblitz and Menezes [15], and by Scott [22]. The consenus is that the appropriate way to scale security is to increase the security multiplier rather than increase the size of the prime modulus.

Note that by "contemporary levels of security", we mean a 1024-bit prime extension field size, and a group size of 160-bits. This implies roughly the same security as 1024-bit RSA [22].

Here we are concerned with the calculation of the Tate pairing, denoted $e(P, Q)$, which evaluates as an element of order $r$ in $\mathbb{F}_{p^k}$ where $P$ is a point of order $r$ on $E(\mathbb{F}_p)$ and $Q$ is a point on $E(\mathbb{F}_{p^k})$.

The rest of this papers is structured as follows: First we recall the supersingular curves originally recommended for use with IBE. Then by making a small change to the definition of these curves we draw attention to a class of non-supersingular curves with a useful property. Next we demonstrate that such curves are quite easy to generate. In our main contribution we describe some new algorithms which exploit the special property of these curves to speed up the calculation of the Tate pairing. We conclude with some results and by suggesting some extensions of the basic idea.

## 2 Supersingular Curves

In their original paper [6], Boneh and Franklin recommend the use of either of these supersingular curves over $\mathbb{F}_p$

$$y^2 = x^3 + Ax, \text{ where } p \equiv 3 \bmod 4 \tag{1}$$

$$y^2 = x^3 + B, \text{ where } p \equiv 2 \bmod 3 \tag{2}$$

On supersingular curves the modified pairing is calculated as $\hat{e}(P, Q) = e(P, \psi(Q))$, where $e(P, Q)$ denotes the Tate pairing, and $\psi(.)$ denotes the distortion map. For the first curve an appropriate distortion map is defined as $\psi_1 : (x, y) \rightarrow (-x, \alpha y)$ where $\alpha = \sqrt{-1}$, and for second curve the distortion map is $\psi_2 : (x, y) \rightarrow (\beta x, y)$ where $\beta$ is a non-trivial cube root of unity. Note that both $\alpha$ and $\beta$ are elements of the extension field $\mathbb{F}_{p^2}$, corresponding to the security multiplier value of $k = 2$.

In the Boneh and Franklin IBE scheme there is a necessity to hash identities to curve points. For the second curve this can be done by hashing the identity string to the $y$ coordinate, and then solving the modular cubic equation for $x$. Since $p \equiv 2 \bmod 3$, this will always be possible. For the first curve one could hash the identity to $x$ and test if $x^3 + ax$ is a quadratic residue. If it is not then negate $x$. Then solve the modular quadratic for $y$, and choose one of the two solutions according to some convention. This will always work as $p \equiv 3 \bmod 4$ implies that -1 is a quadratic non-residue.

## 3 Not Supersingular Curves

Consider now these non-supersingular curves over $\mathbb{F}_p$

$$y^2 = x^3 + Ax, \text{ where } p \equiv 1 \bmod 4 \tag{3}$$

$$y^2 = x^3 + B, \text{ where } p \equiv 1 \bmod 3 \tag{4}$$

Recall that IBE can equally well be implemented on these curves, using the Tate pairing $e(P, Q)$ directly. Suitable curves can be found with $k = 2$, but of course we are no longer restricted to this value alone – larger values of $k$ are also possible (see below).

Note that all that has been changed is the congruence conditions applying to $p$. For our convenience here we will describe these curves as the not-supersingular (NSS) curves to distinguish them from the generality of ordinary curves. Under these circumstances what becomes of the distortion maps? Well of course they are no longer distortion maps, as now $\alpha, \beta \in F_p$. However these mappings continue to be useful, as we will see, not as distortion maps, but rather as efficient *endomorphisms.*

What about hashing identities to curve points on these curves? One interesting feature of $k = 2$ curves is that both the curve and its quadratic twist have the same embedding degree of $k = 2$. This arises from the condition that the group order $r$ divides both $p+1$ and $p+1-t$ (the number of points on the curve), where $t$ is the trace of the Frobenius for the particular curve [2]. Therefore it follows that $r$ also divides $p + 1 + t$, the number of points on the quadratic twist of the curve, and either curve is a suitable vehicle for IBE.

To be concrete we will from this point on concentrate our attention to the curve of equation (4), although all our results apply equally to the other curve. For simplicity choose $p = 7 \bmod 12$, so that -1 is a quadratic non-residue, and a quadratic twist of the original curve is given by $y^2 = x^3 - B$. Then if an identity is hashed to a value $x$, if $x^3 + B$ is not a quadratic residue, then $(-x)^3 - B$ will be. So the trick is to hash to either the original curve or to the twisted curve, depending on the identity. IBE public parameters for both the curve and its twist must be maintained, but other than that the IBE implementation will proceed smoothly with negligible additional overhead.

## 4 Curve Generation

While generating suitable parameters for supersingular curves is easy, it is much more challenging to generate suitable non-supersingular curves. However a lot of progress has been made. Just as pairing-based cryptography was getting started, Miyaji, Nakabayashi and Takano [19] described a method for generating non-supersingular curves with embedding degrees of 3, 4 and 6. Barreto, Lynn and Scott [3] were the first to provide simple formulae for generating whole families of curves with useful embedding degrees. Their results were extended by Brezing and Weng [7], and later by Barreto and Naehrig [4] who came up with formulae which allow the generation of $k = 12$ curves with many nice properties. We will return to these curves later, but for now will just make the observation that the majority of such curves are of the not-supersingular form.

By far the most general method for generating pairing-friendly non-supersingular curves in that due to Cocks and Pinch [5], and this is the method that we shall use to generate $k = 2$ not-supersingular curves suitable as replacements for the standard supersingular curves described above, for use with Boneh and Franklin IBE. Crucially (for us) the Cocks-Pinch algorithm allows a free choice of the group order $r$. It is well known that a choice of a low Hamming weight $r$ speeds up the Tate pairing calculation [2], [13], [21]. Alternatively it suffices if a small multiple of $r$ has a low Hamming weight.

While these methods provide formulae for determining the prime modulus $p$ and the trace of the Frobenius $t$ of the desired curve, they do not generate the curve parameters directly. For this the method of Complex Multiplication must be used [10]. Note that the not-supersingular curves are associated with a CM discriminant of $D = -4$ and $D = -3$ respectively. In these cases determining the curve parameters is particularly simple, as for example demonstrated in [4].

For our NSS curve (4), the Cocks-Pinch algorithm can be described very simply: Select a suitable $r$ of low Hamming weight (or a small multiple of which has low hamming weight). Calculate $v = \sqrt{-4/3} \bmod r$. Set $t = \omega r$. Keep adding $r$ to $v$ until $p = (3v^2 + t^2)/4$ is prime. (Choose $\omega$ so that $p$ is 512 bits). Note that $r \mid p+1-t$. Finally use the CM method to find the curve parameter $B$ associated with the curve of order $p + 1 - t$. (There are 6 possible group orders generated by the CM method in this case [10], so care must be taken to choose the right one). This truncated description of the method is adequate for our immediate purposes; for a more detailed description the reader is encouraged to refer to [5].

## 5 Efficient Pairings on NSS Curves

In their paper Gallant, Lambert and Vanstone [14] describe an efficient method for point multiplication, that applies to NSS curves, and indeed this paper is the main source of inspiration for the current work, although we exploit the endomorphism in a completely different way. Another motivation comes from consideration of the $\eta_T$ pairing as described in [1].

In the course of calculating the Tate pairing $e(P, Q)$ the first parameter, the point $P$, is multiplied by its order. Of course this results in the point-at-infinity.

In the course of this process the intermediate values of this point, as it typically follows a simple double-and-add trajectory to its pre-ordained destination, along with the second parameter $Q$, are used to accumulate the pairing value. In some contexts the value of $P$ may be fixed and known in advance (for example it may be a public parameter, or a private key). In this case these intermediate values can be precalculated and stored, with some performance benefit [21].

Choosing as a parameter a group order $r$ of low Hamming weight drastically reduces the number of expensive add steps, and hence speeds the algorithm. However does this process take full advantage of our ability to choose $r$? The intuition that led to the discovery of the $\eta_T$ pairing was that the multiplication by the group order could perhaps be divided into two parts. At the "half-way" stage, the point $P$ might be "close" to where it started. Therefore the second half of the iteration would hopefully be a simple function of the first part, and so only half the number of iterations might be required. Here we demonstrate that something similar can be achieved for NSS curves.

Gallant, Lambert and Vanstone [14] have pointed out the following useful facts about NSS curves, and the efficient endomorphisms that they support.

For the curve (3) let $P$ be a point of prime order $r$, with coordinates $(x, y)$, and take $\lambda$ such that $\lambda^2 + 1 = 0 \bmod r$. Then the point $\lambda P$ has coordinates $(-x, \alpha y)$, where $\alpha$ is a square root of -1 mod $p$. Note that there are two possibilities for $\alpha$, depending on the two possible solutions of the quadratic equation for $\lambda$. The endomorphism in this case is defined as $\phi_1 : (x, y) \rightarrow (-x, \alpha y)$.

For the curve (4) let $P$ be a point of prime order $r$, with coordinates $(x, y)$, and take $\lambda$ such that $\lambda^2 + \lambda + 1 = 0 \bmod r$. Then the point $\lambda P$ has coordinates $(\beta x, y)$, where $\beta$ is a non-trivial cube root of unity mod $p$. Note that there are two possibilities for $\beta$, depending on the two possible solutions of the quadratic equation for $\lambda$. The endomorphism is this case is defined as $\phi_2 : (x, y) \rightarrow (\beta x, y)$.

This implies that given a point $P$ on one of these curves, one can immediately determine a fixed multiple of the point, with a single field multiplication. In [14] this is exploited to develope a fast point multiplication algorithm.

Focusing on the latter curve, our idea is that $\lambda$ should be chosen in advance, of low Hamming weight. For example we might choose $\lambda = 2^n$. Then select as $r$ a large prime divisor of $\lambda^2 + \lambda + 1$. For example $n = 87$, and $r = (2^{174} + 2^{87} + 1)/73$ (a 168-bit prime) would seem to be a suitable choice. Using a double and add algorithm for the calculation of the Tate pairing would require the calculation of $2^n(2^n P + P) + P$ [1]. However using the endomorphism we can immediately know the value of $2^n P + P$. And this implies that we know the sequence of points that will occur during the final $2^n$ doublings, without having to explicitly calculate them (exploiting the well-known commutativity of the endomorphism with point multiplication: $a\phi(P) = \phi(aP)$). This results in significant computational savings.

Now we explain our technique in a little more detail. Given the point $P$ with coordinates $(x, y)$, and using the endomorphism, it is easy to calculate $2^n P + P$ as

[1] As pointed out by Duursma and Lee [11] the final point addition can be omitted without changing the value of the pairing.

the point $(-(\beta+1)x, -y)$. Clearly if the values of the initial $2^n$ point doublings were stored, the values of the final $2^n$ doublings can be found at the cost of a single field multiplication, resulting in a faster algorithm. Alternatively if the value of $P$ is fixed, only half the storage would be required. Either way the result is a more efficient algorithm. However it is possible to do a little better than this.

For the first $n$ iterations of Miller's algorithm the contribution to the pairing value is $(y_Q - y_i) - m_i(x_Q - x_i)$, where $(x_i, y_i)$ is the point $2^iP$, $m_i$ is the line slope resulting from the current point doubling, and $(x_Q, y_Q)$ is the point $Q$. This value can be multiplied at will by any element of $\mathbb{F}_p$, as the effect of any such multiplication will be wiped out by the final exponentiation. For the final $n$ iterations the contribution will be $(y_Q + y_i) + (\beta^2+1)m_i(x_Q + (\beta+1)x_i)$ – note the adjusted value of the slope. But since $\beta$ is a non-trivial cube root of unity, we know that $\beta^2+\beta+1 = 0 \bmod p$. Substituting and simplifying we have $(y_Q+y_i)+(\beta^2+1)m_i(x_Q+(\beta+1)x_i) = (y_Q+y_i)+m_i(\beta x_Q - x_i)$. Now multiply by -1 to obtain the equivalent contribution of $(-y_Q - y_i) - m_i(\beta x_Q - x_i)$.

Observe therefore that we can obtain the same values by switching the point $Q$ to $\bar{Q}$ for the final $n$ iterations, where $\bar{Q} = (\beta x_Q, -y_Q)$, and using exactly the same sequence of $(x_i, y_i)$ as we did for the first $n$ iterations.

## 5.1 A Basic Algorithm

We are now ready to bring these ideas together and describe our modified BKLS-GHS algorithm in detail. First we equip ourselves with a library which can add or double points on an elliptic curves by means of a function $A.add(B)$ which adds $B$ to $A$, and returns the line slope $m$.

Next we need a function $g(.)$ to calculate the contribution of the current iteration to the pairing value. The returned value is used for the first $n$ iterations, and the values for the final $n$ iterations are calculated at the same time (at very little extra cost) and stored for later use.

**Algorithm 1.** Function $g(.)$

```
INPUT: $A, B, Q, i$
 1: $x_i, y_i \leftarrow A$
 2: $x_Q, y_Q \leftarrow Q$
 3: $m_i = A.add(B)$
 4: store $-y_Q - y_i - m_i(\beta x_Q - x_i)$ in an array element $s[i]$
 5: return $y_Q - y_i - m_i(x_Q - x_i)$
```

In practice the function $A.add(B)$ will be faster if the point $A$ is represented in projective coordinates, which makes for a somewhat more complex $g(.)$ function. We omit the details, except to point out that much of the calculation is shared between the stored point and the returned point, with further savings.

For optimal performance the point $Q$ is deliberately placed into the trace-zero subgroup, which means that only the variable $y_Q$ is in $\mathbb{F}_{p^2}$. The variables $x_i, y_i, x_Q, m_i$ are all in $\mathbb{F}_p$. See [21] for details. The returned and stored values are in $\mathbb{F}_{p^2}$. Sometimes we pass the dummy parameter – for $i$. As the notation suggests, in this case the storage step is omitted.

Care must be taken to ensure that correct non-trival cube root of unity for $\beta$ is chosen, as there are two possibilities associated with the two solutions for $\lambda^2 + \lambda + 1 = 0 \bmod r$. The right value can easily be found by trial and error, and the value of $\beta x_Q$ can then be precalculated and stored.

For the particular case $n = 87$ the full Tate pairing algorithm is given in Algorithm 2. This algorithm will also work for any choice of $\lambda = 2^n$ which leads to a near-prime value of $r = \lambda^2 + \lambda + 1$. However in practice, and in the range of useful values, good values for $n$ are hard to find.

**Algorithm 2.** Computation of $e(P, Q)$ on NSS curve (4), $k = 2$, $\lambda = 2^{87}$, $r = (\lambda^2 + \lambda + 1)/73$

```
INPUT: $P, Q$
OUTPUT: $e(P, Q)$
1: $A \leftarrow P$, $f \leftarrow 1$
2: for $i \leftarrow 1$ to 87 do
3:     $f \leftarrow f^2.g(A, A, Q, i)$
4: end for
5: $f \leftarrow f.g(A, P, Q, -)$
6: for $i \leftarrow 1$ to 87 do
7:     $f \leftarrow f^2.s[i]$
8: end for
9: return $f^{(p-1)(p+1)/r}$
```

### 5.2 A Better Algorithm

Consider now the slightly more complicated choice of $\lambda = 2^a + 2^b$. In this case we have much greater control of $r$, and it is much easier to find a prime value which still has a very low Hamming weight. For example choosing $\lambda = 2^{80} + 2^{16}$ gives a prime $r = \lambda^2 + \lambda + 1$ of 161 bits. A slight complication arises in this case, as the multiplication of the point $P$ by $r$ no longer follows a purely double-and-add algorithm, and so Miller's algorithm needs to be slightly modified to accomodate this. In the following algorithm 3, the variable $h$ is used to handle this modification.

The extra storage requirement for the array $s$ is not very large (2x512x81 bits), but in some circumstances it may become an issue. An alternative version of the algorithm requires no storage, for a little extra work. See algorithm 4.

## 6 Results

An important first step in an implementation is to use the Cocks and Pinch algorithm to generate a suitable 512-bit, k=2 NSS elliptic curve. The curve $y^2 = x^3 + 5 \bmod p$, for the 512-bit prime $p$, where

$$p = 1145747568399549380635317418620582531453546123676759744111553372850507052782315453265765699123447398664170319394034355982362866887873432690950208939349364\!3$$

**Algorithm 3.** Computation of $e(P, Q)$ on NSS curve (4), $k = 2$, $\lambda = 2^a + 2^b$, $a > b$, $r = \lambda^2 + \lambda + 1$. Requires an array $s$ of length $a$.

```
INPUT: $P, Q$
OUTPUT: $e(P, Q)$
 1: $A \leftarrow P$, $f \leftarrow 1$, $j \leftarrow 1$
 2: for $i \leftarrow 1$ to $a - b$ do
 3:     $f \leftarrow f^2.g(A, A, Q, j{++})$
 4: end for
 5: $f \leftarrow f.g(A, P, Q, j{++})$
 6: for $i \leftarrow 1$ to $b$ do
 7:     $f \leftarrow f^2.g(A, A, Q, j{++})$
 8: end for
 9: $f \leftarrow f.g(A, P, Q, -)$
10: $h \leftarrow f$, $j \leftarrow 1$
11: for $i \leftarrow 1$ to $a - b$ do
12:     $f \leftarrow f^2.s[j{++}]$
13: end for
14: $f \leftarrow f.s[j{++}]$
15: $f \leftarrow f.h$
16: for $i \leftarrow 1$ to $b$ do
17:     $f \leftarrow f^2.s[j{++}]$
18: end for
19: return $f^{(p-1)(p+1)/r}$
```

**Algorithm 4.** Computation of $e(P, Q)$ on NSS curve (4), $k = 2$, $\lambda = 2^a + 2^b$, $a > b$, $r = \lambda^2 + \lambda + 1$. No extra storage requirement.

```
INPUT: $P, Q$
OUTPUT: $e(P, Q)$
 1: $A \leftarrow P$, $f_1 \leftarrow 1$, $f_2 \leftarrow 1$, $j \leftarrow 1$
 2: for $i \leftarrow 1$ to $a - b$ do
 3:     $f_1 \leftarrow f_1^2.g(A, A, Q, 0)$
 4:     $f_2 \leftarrow f_2^2.s[0]$
 5: end for
 6: $f_1 \leftarrow f_1.g(A, P, Q, 0)$
 7: $f_2 \leftarrow f_2.s[0]$
 8: for $i \leftarrow 1$ to $b$ do
 9:     $f_1 \leftarrow f_1^2.g(A, A, Q, 0)$
10:     $f_2 \leftarrow f_2^2.s[0]$
11: end for
12: $f_1 \leftarrow f_1.g(A, P, Q, -)$
13: $f \leftarrow f_1^{\lambda}.f_2$
14: return $f^{(p-1)(p+1)/r}$
```

was quickly found. The Tate pairing was calculated on this curve using algorithms 3 and 4 with $a = 80$ and $b = 16$, and compared with the pairing calculated on the original Boneh and Franklin supersingular curve using the method described in [21]. As anticipated, the NSS curve pairings are significantly faster.

**Table 1.** NSS vs Supersingular Tate Pairing – 3GHz Intel PIV

| Curve type | $\mathbb{F}_p$ muls | Time (ms) |
|---|---|---|
| 512-bit, $k = 2$ Supersingular curve | 4070 | 8.9 |
| 512-bit, $k = 2$ NSS curve, with storage | 3163 | 7.2 |
| 512-bit, $k = 2$ NSS curve, no storage | 3329 | 7.5 |

We provide both timings and a count of the total number of $\mathbb{F}_p$ modular multiplications and squarings required. In the implementation we used a final exponentiation based on the calculation of a Lucas sequence, which allows easy times-two compression of the output, as described in [21] and [23].

## 7 Extensions

The basic idea can be extended in a few directions. Firstly the same basic technique will also work for the "other" NSS curve of equation (3). The idea could also be applied to the non-supersingular curves with CM discriminants of $D = -7$ and $D = -8$, as described in [14], although in these cases the savings would be much less significant as the endomorphisms are much more complex to calculate. In fact it would be more correct to refer to the class of exploitable curves as "small discriminant CM curves".

If the first parameter $P$ is fixed, then no savings will be realised in terms of computation using this method as all the multiples of $P$ can be precalculated and stored. However using NSS curves only half the storage will be required, which leads to greater efficiency, and which might be significant in a constrained environment.

The method has been described in the context of a security multiplier of $k = 2$, but it also applies immediately to higher values of $k$. In these cases the computational time savings will be smaller, as the implicit point multiplication of $P$ by $r$ becomes a less significant part of the overall calculation [22].

We note in passing that the method of Gallant, Lambert and Vanstone [14] also has direct application to pairing-based protocols which require point multiplication, such as the Boneh and Franklin IBE, if they are implemented on NSS curves. With our choice of group order the deployment of this scheme becomes particularly simple: Calculate $kP$ as $k_1P + k_2\phi(P)$, where $k_2 = k/\lambda, k_1 = k \bmod \lambda$. See [14] for details.

Finally we point out that many curves that have been suggested as suitable for use in pairing-based cryptography are in fact already of the NSS form [3], [7], and furthermore have a group order of the required form. For example the $k = 12$ curve suggested in Appendix A of [3] is of this form, as is the nice $k = 8$ curve suggested by Brezing and Weng [7] and implemented in [22]. This is facilitated by the group order of these curves being derived from a cyclotomic polynomial which, as luck would have it, is of the same form as $r = \lambda^2 + \lambda + 1$.

Are NSS curves any less secure than supersingular curves or indeed general pairing-friendly non-supersingular curves? There is no reason to think so. In standard elliptic curve cryptography some classes of curves are considered as weaker than others, although sometimes the reasons are not well supported by any hard evidence. For example it is considered in certain quarters (for example the German National Security Agency), that curves with smaller CM discriminants are in some sense weaker than others [8]. Whatever the merits of such judgements, they do not apply in the pairing context, where it is already known that there is an index calculus attack on the extension field $\mathbb{F}_{p^k}$, which arises as a direct consequence of having a small embedding degree $k$. Of course by choosing $p$ and $k$ wisely this index calculus attack becomes infeasible. It remains an interesting open question whether or not there are any weak classes of pairing-suitable curves, supersingular or non-supersingular.

## 8 Conclusions

We have described a method of calculating pairings on certain non-supersingular curves, which is faster than using the equivalent supersingular curve, as originally recommended by Boneh and Franklin [6] for use in Identity Based Encryption. The proposed method is more efficient in terms of time (or of space) than any other method so far proposed for elliptic curves over fields of prime characteristic and at contemporary levels of security.

## Acknowledgement

Thanks to Steven Galbraith for his comments on an early draft of this paper.

## References

1. P.S.L.M. Barreto, S. Galbraith, C. O hEigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. Cryptology ePrint Archive, Report 2004/375, 2004. `http://eprint.iacr.org/2004/375`.
2. P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology – Crypto'2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–68. Springer-Verlag, 2002.
3. P.S.L.M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in Communication Networks – SCN'2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 263–273. Springer-Verlag, 2002.
4. P.S.L.M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. Cryptology ePrint Archive, Report 2005/133, 2005. `http://eprint.iacr.org/2005/133`.
5. I. F. Blake, G. Seroussi, and N. P. Smart, editors. *Advances in Elliptic Curve Cryptography, Volume 2.* Cambridge University Press, 2005.
6. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

7. F. Brezing and A. Weng. Elliptic curves suitable for pairing based cryptography. Cryptology ePrint Archive, Report 2003/143, 2003. Available from `http://eprint.iacr.org/2003/143`.
8. J. Buchmann and H. Baier. Efficient construction of cryptographically strong elliptic curves. In *INDOCRYPT*, pages 191–202, 2000.
9. D. Coppersmith. Fast evaluation of logarithms in fields of characteristics two. In *IEEE Transactions on Information Theory*, volume 30, pages 587–594, 1984.
10. R. Crandall and C. Pomerance. *Prime Numbers: a Computational Perspective.* Springer-Verlag, Berlin, 2001.
11. I. Duursma and H. S. Lee. Tate-pairing implementations for tripartite key agreement. In *Advances in Cryptology – Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 111–123. Springer-Verlag, 2003.
12. G. Frey, M. Müller, and H. Rück. The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717–1719, 1999.
13. S. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithm Number Theory Symposium – ANTS V*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer-Verlag, 2002.
14. R.P. Gallant, R.J. Lambert, and S.A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In *Advances in Cryptology – Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer-Verlag, 2001.
15. N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. Cryptology ePrint Archive, Report 2005/076, 2005. `http://eprint.iacr.org/2005/076`.
16. A. K. Lenstra. Unbelievable security. Matching AES security using public key systems. In *Advances in Cryptology – Asiacrypt 2001*, volume 2248, pages 67–86. Springer-Verlag, 2001.
17. A. Menezes. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993.
18. V. Miller. Short programs for functions on curves. unpublished manuscript, 1986.
19. A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Transactions on Fundamentals*, E84-A(5):1234–1243, 2001.
20. D. Page, N.P. Smart, and F. Vercauteren. A comparison of MNT curves and supersingular curves. Cryptology ePrint Archive, Report 2004/165, 2004. `http://eprint.iacr.org/2004/165`.
21. M. Scott. Computing the Tate pairing. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 293–304. Springer-Verlag, 2005.
22. M. Scott. Scaling security in pairing-based protocols. Cryptology ePrint Archive, Report 2005/139, 2005. `http://eprint.iacr.org/2005/139`.
23. M. Scott and P. Barreto. Compressed pairings. In *Advances in Cryptology – Crypto' 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 140–156. Springer-Verlag, 2004. Also available from `http://eprint.iacr.org/2004/032/`.

# Reconsideration on the Security of the Boneh-Franklin Identity-Based Encryption Scheme

Mototsugu Nishioka

HITACHI Systems Lab, Kawasaki, Japan
nishioka@sdl.hitachi.co.jp
http://www.sdl.hitachi.co.jp/

**Abstract.** The Boneh-Franklin identity-based encryption (BF-IBE) scheme [6] is well-known as a fully functional identity-based encryption (IBE) scheme. Recently, Galindo [13] pointed out a flaw in the original proof of the security of the BF-IBE scheme. He claims that its security can be fixed without changing both the scheme and the underlying assumption if the efficiency of the security reduction is sacrificed. This result would be bad news for many cryptographic schemes [1, 7, 10, 15] that are based on the BF-IBE scheme because an inefficient security reduction would imply either the lower security level or the use of larger key sizes to attain a given security level. In this paper, we give a new proof of the security of the BF-IBE scheme, showing that it has a tighter security reduction than had been previously believed.

## 1 Introduction

Identity-based cryptography utilizing pairing has become a fashionable topic since the fully functional identity-based encryption (IBE) scheme was proposed in 2001 by Boneh and Franklin [6]. Historically, identity-based cryptography was proposed by Shamir [18] in 1984 to simplify certificate management in e-mail systems. In this approach, a user can choose an arbitrary string (e.g., an e-mail address and telephone number) as his/her own public key. The user receives his/her private key from a "Private Key Generator" (PKG) after authenticating himself/herself and registering the public key in the PKG. This paradigm bypasses the trust problems that arise in traditional certificate-based public key infrastructures (PKIs).

Although various identity-based signature (IBS) and authentication (IBA) schemes have been proposed, the fully functional identity-based encryption (IBE) scheme was not found until 2001 by Boneh and Franklin [6] and independently by Cocks [11]. Boneh and Franklin construct the IBE scheme (called the "BF-IBE scheme") by utilizing the Weil paring. Cocks describes another construction using quadratic residues modulo a composite, which is less practical than the BF-IBE scheme. The BF-IBE scheme is very practical for use and has had a great influence on later designs of cryptographic protocols, even though its security is given in the heuristic model (i.e., the random oracle model).

© Springer-Verlag Berlin Heidelberg 2005

Recently, Galindo [13] pointed out a flaw in the original proof of the (IND-ID-CCA sense) security of the BF-IBE scheme. This flaw has consequences for many other schemes [1, 7, 10, 14, 15, 16, 17] that are based on the BF-IBE scheme. He claims that the security of the BF-IBE scheme can be fixed without changing both the scheme and the underlying assumption if the efficiency of the security reduction is sacrificed. The security reduction is a measure indicating how the strength of the underlying intractable problem, which is used as a cryptographic assumption, is preserved in the underlying scheme's security. An inefficient security reduction would imply either the lower security level or the use of larger key sizes to attain a given security level. Therefore it is important to find a proof of the security that gives a tight security reduction. In this paper, we give a new proof of the security of the BF-IBE scheme, showing that it has a tighter security reduction than had been previously believed.

## 2 Preliminaries

Notations and conventions. We use standard notations and conventions for writing probabilistic algorithms and experiments. If $A$ is a probabilistic algorithm, then $A(x_1, x_2, \dots; r)$ is the result of running $A$ on inputs $x_1, x_2, \dots$ and coin tosses $r$. We let $y \leftarrow A(x_1, x_2, \dots; r)$ denote the experiment of picking $r$ at random and letting $y$ be $A(x_1, x_2, \dots; r)$. When $S$ is a probability space, $x \leftarrow S$ denotes the operation of picking an element with its distribution from $S$. If $S$ is a set, $x \leftarrow S$ is the operation of picking an element uniformly from $S$. We use $x, y \leftarrow S$ as shorthand for $x \leftarrow S;\ y \leftarrow S$. For probability spaces or sets $S_1, S_2, \dots$, the notation $\Pr[\,x_1 \leftarrow S_1;\ x_2 \leftarrow S_2;\ \dots\ :\ p(x_1, x_2, \dots)\,]$ denotes the probability that the predicate $p(x_1, x_2, \dots)$ is true after the ordered execution of the algorithms $x_1 \leftarrow S_1,\ x_2 \leftarrow S_2$, etc. If $\alpha$ is neither an algorithm, a probabilistic space, nor a set then $x \leftarrow \alpha$ is a simple assignment statement. We say that a function $\mu : \mathbb{N} \rightarrow \mathbb{R}$ is *negligible* if for every constant $c > 0$ there exists an integer $k_c$ such that $\mu(k) \leq k^{-c}$ for all $k \geq k_c$. For a set $A$, $\#A$ denotes the number of elements in $A$.

### 2.1 Bilinear Maps and the Bilinear Diffie-Hellman Assumption

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order $q$ for some prime $q$. Then a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is said to be an *admissible bilinear map* if it satisfies the following properties:

1. Bilinear: We say that a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is *bilinear* if $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}$.
2. Non-degenerate: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$. Observe that $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of prime order this implies that if $P$ is a generator of $\mathbb{G}_1$ then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_2$.
3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

It is known that an admissible bilinear map can be constructed by using the Weil pairing on elliptic curves (see [6] for details).

We say that a randomized algorithm $\mathcal{G}$ is a *BDH parameter generator* if (1) $\mathcal{G}$ takes a security parameter $k \in \mathbb{N}$, (2) $\mathcal{G}$ runs in polynomial time in $k$, and (3) $\mathcal{G}$ outputs a prime number $q$, the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ of order $q$, and the description of an admissible bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. We denote the output of $\mathcal{G}$ by $\mathcal{G}(1^k) = \langle\, q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}\,\rangle$.

**Bilinear Diffie-Hellman (BDH) Assumption.** Let $\mathcal{G}$ be a BDH parameter generator. We define the advantage of an algorithm $A$ in solving the BDH problem for $\mathcal{G}$ as

$$\mathsf{Adv}^{\mathrm{bdh}}_{\mathcal{G},A}(k) = \Pr\Big[\langle\, q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}\,\rangle \leftarrow \mathcal{G}(1^k)\ ;\ P \leftarrow \mathbb{G}_1^*\ ;\ a, b, c \leftarrow \mathbb{Z}_q^*\ : \\ A(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP, cP) = \hat{e}(P,P)^{abc}\Big].$$

We say that $\mathcal{G}$ *satisfies the BDH assumption* if $\mathsf{Adv}^{\mathrm{bdh}}_{\mathcal{G},A}(k)$ is negligible for any probabilistic polynomial-time (PPT) algorithm $A$.

### 2.2 Chosen Ciphertext Security for Identity-Based Encryption

We say that an IBE scheme $\Pi$ is polynomial time indistinguishable (or semantically secure) against an adaptive chosen ciphertext attack (IND-ID-CCA) if no PPT algorithm $A$ has a non-negligible advantage against the Challenger in the following IND-ID-CCA game:

**Setup:** The challenger takes a security parameter $k \in \mathbb{N}$ and runs the Setup algorithm. It gives the adversary the resulting system parameters params. It keeps the master-key to itself.

**Phase 1:** The adversary issues queries $q_1, \ldots, q_m$ where $q_i$ is one of:
- Extraction query $\langle\, \mathsf{ID}_i\,\rangle$. The challenger responds by running algorithm Extract to generate the private key $d_i$ corresponding to the public key $\mathsf{ID}_i$. It sends $d_i$ to the adversary.
- Decryption query $\langle\, \mathsf{ID}_i, C_i\,\rangle$. The challenger responds by running algorithm Extract to generate the private key $d_i$ corresponding to $\mathsf{ID}_i$. It then runs algorithm Decrypt to decrypt the ciphertext $C_i$ using $d_i$. It sends the resulting plaintext to the adversary.

These queries may be asked adaptively, that is, each query $q_i$ may depend on the replies to $q_1, \ldots, q_{i-1}$.

**Challenge:** Once the adversary decides that Phase 1 is over it outputs two equal length messages $M_0, M_1 \in \mathcal{M}$ and an identity ID on which it wishes to be challenged. The only constraint is that ID did not appear in any private key extraction query in Phase 1. The challenger picks a random bit $b \in \{0,1\}$ and sets $C = \mathsf{Encrypt}(\mathsf{params}, \mathsf{ID}, M_b)$. It sends $C$ as the challenge to the adversary.

**Phase 2:** The adversary issues more queries $q_{m+1}, \ldots, q_n$ where $q_i$ is one of:
- Extraction query $\langle\, \mathsf{ID}_i\,\rangle$ where $\mathsf{ID}_i \neq \mathsf{ID}$. Challenger responds as in Phase 1.
- Decryption query $\langle\, \mathsf{ID}_i, C_i\,\rangle \neq \langle\, \mathsf{ID}, C\,\rangle$. Challenger responds as in Phase 1.

These queries may be asked adaptively as in Phase 1.

**Guess:** Finally, the adversary outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$.

We refer to such an adversary $A$ as an IND-ID-CCA adversary. We define $A$'s advantage in attacking the scheme $\Pi$ as $\mathsf{Adv}^{\text{ind-id-cca}}_{\Pi,A}(k) = \left|\Pr[b = b'] - \frac{1}{2}\right|$. The probability is over the random bits used by the challenger and the adversary.

**Definition 1.** *We say that the IBE scheme $\Pi$ is IND-ID-CCA secure if $\mathsf{Adv}^{\text{ind-id-cca}}_{\Pi,A}(k)$ is negligible for any PPT algorithm $A$.*

### 2.3 The Boneh-Franklin IBE Scheme

The BF-IBE scheme, which is called $\mathsf{Full\text{-}Ident}$ in [6], is given as follows. We let $\mathcal{G}$ be some BDH parameter generator.

**Setup:** Given a security parameter $k \in \mathbb{N}$, do the following: (1) $\langle\, q, \mathbb{G}_1, \mathbb{G}_2, \hat{e} \,\rangle \leftarrow \mathcal{G}(1^k)$, (2) $P \leftarrow \mathbb{G}_1$ ; $s \leftarrow \mathbb{Z}_q^*$ ; $P_{pub} \leftarrow sP$, and (3) choose cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$, $H_2 : \mathbb{G}_2 \to \{0,1\}^m$, $H_3 : \{0,1\}^m \times \{0,1\}^n \to \mathbb{Z}_q^*$, and $H_4 : \{0,1\}^m \to \{0,1\}^n$. (In the security analysis, we view the $H_i$ $(1 \le i \le 4)$ as random oracles.) The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^m \times \{0,1\}^n$. The system parameters are $\mathsf{params} = \langle\, q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, m, n, P, P_{pub} \,\rangle$, and the $\mathsf{master\text{-}key}$ is $s \in \mathbb{Z}_q^*$. The following algorithms, $\mathsf{Extract}$, $\mathsf{Encrypt}$, and $\mathsf{Decrypt}$, can make queries to every hash function $H_i$ $(1 \le i \le 4)$: For the $H_i$-query $x$, we denote its response from $H_i$ as $H_i(x)$.

**Extract:** For a given string $\mathsf{ID} \in \{0,1\}^*$, the algorithm sets the private key $d_{\mathsf{ID}}$ to be $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$, where $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$.

**Encrypt:** To encrypt $M \in \mathcal{M}$ under the public key $\mathsf{ID}$, do the following: (1) $Q_{\mathsf{ID}} \leftarrow H_1(\mathsf{ID})$ ; $\sigma \leftarrow \{0,1\}^m$ ; $r \leftarrow H_3(\sigma, M)$, and (2) set the ciphertext to be $C = \langle\, rP, \sigma \oplus H_2(g_{\mathsf{ID}}{}^r), M \oplus H_4(\sigma) \,\rangle$, where $g_{\mathsf{ID}} = \hat{e}(Q_{\mathsf{ID}}, P_{pub})$.

**Decrypt:** Let $C = \langle\, U, V, W \,\rangle$ be a ciphertext encrypted using the public key $\mathsf{ID}$. If $C \notin \mathcal{C}$, reject the ciphertext. To decrypt $C$ by using the private key $d_{\mathsf{ID}}$, do the following: (1) $\sigma \leftarrow V \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U))$ ; $M \leftarrow W \oplus H_4(\sigma)$ ; $r \leftarrow H_3(\sigma, M)$, and (2) if it holds that $U = rP$, output $M$ as the decryption of $C$; otherwise, reject the ciphertext.

## 3 Security of the Boneh-Franklin IBE Scheme

### 3.1 Background

To prove the security of the Boneh-Franklin IBE scheme (called $\mathsf{Full\text{-}Ident}$), the authors define an IBE scheme (called $\mathsf{BasicIdent}$) and two public-key encryption (PKE) schemes (called $\mathsf{BasicPub}$ and $\mathsf{BasicPub}^{hy}$) in [6]. $\mathsf{BasicIdent}$ is a simplified version of $\mathsf{Full\text{-}Ident}$, $\mathsf{BasicPub}$ is a PKE scheme derived from $\mathsf{BasicIdent}$, and $\mathsf{BasicPub}^{hy}$ is a PKE scheme obtained by applying the Fujisaki-Okamoto conversion [12] to $\mathsf{BasicPub}$. Given these definitions, the following four claims are presented:

**Claim 1.** BasicPub is IND-CPA secure under the BDH assumption. Concretely, suppose there is an IND-CPA adversary $A$ that has advantage $\epsilon(k)$ against BasicPub and $A$ runs in time at most $t(k)$. Suppose $A$ makes at most $q_{H_2}$ queries to the random oracle $H_2$. Then there is an algorithm $B$ that solves the BDH problem for $\mathcal{G}$ with advantage at least $\frac{2\epsilon(k)}{q_{H_2}}$ and running time $t(k) + O((1 + {q_{H_2}}^2)l)$.

**Claim 2.** Suppose there is an IND-ID-CPA adversary $A$ that has advantage $\epsilon(k)$ against BasicIdent and $A$ runs in time at most $t(k)$. Suppose $A$ makes at most $q_E$ private key extraction queries, and at most $q_{H_2}$ queries to the random oracle $H_2$. Then there is an IND-CPA adversary $B$ against BasicPub with advantage at least $\frac{\epsilon(k)}{e(1+q_E)}$ and running time $t(k) + (q_{H_1} + q_E)t_{\mathbb{G}_1^*} + O((1 + (q_{H_1} + q_E + q_D)^2)l)$.

**Claim 3 (Fujisaki-Okamoto [12]).** Suppose there is an IND-CCA adversary $A$ that has advantage $\epsilon_A(k)$ against BasicPub$^{hy}$ and $A$ runs in time at most $t_A(k)$. Suppose $A$ makes at most $q_D$ decryption queries, and at most $q_{H_3}, q_{H_4}$ queries to the random oracles $H_3, H_4$ respectively. Then there exists an IND-CPA adversary $B$ against BasicPub with advantage $\epsilon_B(k)$ and running time $t_B(k)$ where:

$$\epsilon_B(k) \geq FO_{adv}^{\text{ow-cpa}}(\epsilon_A(k), q_{H_3}, q_{H_4}, q_D) = \frac{(\epsilon_A(k) + 1)(1 - 2/q)^{q_D} - 1}{2(q_{H_3} + q_{H_4})} \qquad \text{and}$$

$$t_B(k) \leq FO_{time}^{\text{ow-cpa}}(t_A(k), q_{H_3}, q_{H_4}, q_D) = t_A(k) + q_{H_3}t_{enc} + O((1 + {q_{H_3}}^2 + {q_{H_4}}^2 + q_D q_{H_3}(1 + q_{H_4}))l).$$

**Claim 4.** Suppose there is an IND-ID-CCA adversary $A$ that has advantage $\epsilon(k)$ against Full-Ident and $A$ runs in time at most $t(k)$. Suppose $A$ makes at most $q_E$ private key extraction queries, at most $q_D$ decryption queries, and at most $q_{H_1}$ queries to the random oracle $H_1$. Then there exists an IND-CCA adversary $B$ against BasicPub$^{hy}$ with advantage at least $\frac{\epsilon(k)}{e(1+q_E+q_D)}$ and running time $t(k) + (q_{H_1} + q_E + q_D)t_{\mathbb{G}_1^*} + q_D t_{dec} + O((1 + (q_{H_1} + q_E + q_D)^2)l)$.

Here, $l$ denotes the length of the input to algorithm $A$, $e \approx 2.718$ denotes the base of the natural logarithm, $t_{\mathbb{G}_1^*}$ denotes the time required for computing a scalar multiplication on $\mathbb{G}_1^*$, $t_{enc}$ denotes the time required for encryption in Full-Ident, and $t_{dec}$ denotes the time required for decryption in Full-Ident. (From here on we use this notation.)

Assuming that these claims are true, Claims 1 and 2 induce that BasicIdent is IND-ID-CPA secure under the BDH assumption, and Claims 1, 3 and 4 induce that Full-Ident is IND-ID-CCA secure under the BDH assumption. Namely, the following result is then obtained.

**Boneh and Franklin's Result [6].** Let the hash functions $H_1, H_2, H_3, H_4$ be random oracles. Then Full-Ident is IND-ID-CCA secure assuming BDH problem is hard in groups generated by $\mathcal{G}$. Concretely, suppose there is an IND-ID-CCA adversary $A$ that has advantage $\epsilon_A(k)$ against Full-Ident and $A$ runs in time at most $t_A(k)$. Suppose $A$ makes at most $q_E$ extraction queries, at most $q_D$

decryption queries, and at most $q_{H_i}$ queries to $H_i$ $(1 \leq i \leq 4)$ respectively. Then there is an algorithm $B$ that solves BDH problem in groups generated by $\mathcal{G}$ with running time $t_B(k)$ where:

$$\begin{aligned}\mathsf{Adv}^{\text{bdh}}_{\mathcal{G},B}(k) &\geq 2FO^{\text{ow-cpa}}_{adv}\left(\frac{\epsilon_A(k)}{e(1+q_E+q_D)}, q_{H_3}, q_{H_4}, q_D\right) {q_{H_2}}^{-1},\\ t_B(k) &\leq FO^{\text{ow-cpa}}_{time}(t_A(k) + (q_{H_1}+q_E+q_D)t_{\mathbb{G}_1^*} + q_D t_{dec}\\ &\quad + O((1+(q_{H_1}+q_E+q_D)^2)l), q_{H_3}, q_{H_4}, q_D).\end{aligned}$$

Recently, Galindo [13] pointed out a flaw in the original proof showing that Claim 4 is true. Although the proofs of Claims 2 and 4 are similar, the flaw exists only in the proof of Claim 4. And he claims that it is possible to fix the security of the BF-IBE scheme without changing both the scheme and the underlying assumption if the efficiency of the security reduction is sacrificed [13]. In his revised proof of the security of the BF-IBE scheme, Claim 5 is given instead of Claim 4.

**Claim 5.** Suppose there is an IND-ID-CCA adversary $A$ that has advantage $\epsilon(k)$ against $\mathsf{Full\text{-}Ident}$ and $A$ runs in time at most $t(k)$. Suppose $A$ makes at most $q_E$ private key extraction queries, at most $q_D$ decryption queries, and at most $q_{H_1}$ queries to the random oracle $H_1$. Then there exists an IND-CCA adversary $B_1$ against $\mathsf{BasicPub}^{hy}$ with advantage at least $\frac{\epsilon(k)}{q_{H_1}}\left(1-\frac{q_E}{q_{H_1}}\right)$ and running time $t(k) + (q_E+q_D)t_{\mathbb{G}_1^*} + q_D t_{dec} + O((1+(q_{H_1}+q_E+q_D)^2)l)$.

Assuming that this claim is true, Claims 1, 3 and 5 induce that $\mathsf{Full\text{-}Ident}$ is IND-ID-CCA secure under the BDH assumption. Namely, the following result is then obtained.

**Galindo's Result [13].** Let the hash functions $H_1, H_2, H_3, H_4$ be random oracles. Then $\mathsf{Full\text{-}Ident}$ is IND-ID-CCA secure assuming BDH problem is hard in groups generated by $\mathcal{G}$. Concretely, suppose there is an IND-ID-CCA adversary $A$ that has advantage $\epsilon_A(k)$ against $\mathsf{Full\text{-}Ident}$ and $A$ runs in time at most $t_A(k)$. Suppose $A$ makes at most $q_E$ extraction queries, at most $q_D$ decryption queries, and at most $q_{H_i}$ queries to $H_i$ $(1 \leq i \leq 4)$ respectively. Then there is an algorithm $B$ that solves BDH problem in groups generated by $\mathcal{G}$ with running time $t_B(k)$ where:

$$\begin{aligned}\mathsf{Adv}^{\text{bdh}}_{\mathcal{G},B}(k) &\geq 2FO^{\text{ow-cpa}}_{adv}\left(\frac{\epsilon_A(k)}{q_{H_1}}\left(1-\frac{q_E}{q_{H_1}}\right), q_{H_3}, q_{H_4}, q_D\right) {q_{H_2}}^{-1},\\ t_B(k) &\leq FO^{\text{ow-cpa}}_{time}(t_A(k) + (q_E+q_D)t_{\mathbb{G}_1^*} + q_D t_{dec}\\ &\quad + O((1+(q_{H_1}+q_E+q_D)^2)l), q_{H_3}, q_{H_4}, q_D).\end{aligned}$$

Galindo supposes that $q_{H_1} \gg q_E, q_D$ in his proof of Claim 5. He then claims that the efficiency of his security reduction of the BF-IBE scheme is a bit worse than the previous one. However, there are problems in his proof of Claim 5, and the claim is also unreliable. The details of the problems are given in the proof of Claim 7 in Section 3.2.

## 3.2 Our Proof of the Security

In this section, we give a new proof of the security of the BF-IBE scheme, showing that it has a tighter security reduction than had been previously believed. We give Claim 6 instead of Claim 3, and Claim 7 instead of Claims 4 and 5. The security reduction in Claim 3 corresponds to the case where the underlying public-key encryption (PKE) scheme in the hybrid encryption scheme is secure in the sense of one-way (i.e., OW-CPA). Claim 6 says that the security reduction is drastically improved if the security of the underlying PKE scheme is strengthened to IND-CPA.

**Claim 6.** Suppose there is an IND-CCA adversary $A$ that has advantage $\epsilon_A(k)$ against $\mathsf{BasicPub}^{hy}$ and $A$ runs in time at most $t_A(k)$. Suppose $A$ makes at most $q_D$ decryption queries, and makes at most $q_{H_3}, q_{H_4}$ queries to the random oracles $H_3, H_4$ respectively. Then there exists an IND-CPA adversary $B$ against $\mathsf{BasicPub}^{hy}$ with advantage $\epsilon_B(k)$ and running time $t_B(k)$ where:

$$\begin{aligned}\epsilon_B(k) &\geq FO_{adv}^{\text{ind-cpa}}(\epsilon_A(k), q_{H_3}, q_{H_4}, q_D)\\ &= \left(\left(1-\frac{3}{q}-\frac{1}{2^n}\right)^{q_D} + \frac{q_{H_3}+q_{H_4}}{2^m}\right)\cdot \epsilon_A(k) + \frac{1}{2}\left(1-\frac{3}{q}-\frac{1}{2^n}\right)^{q_D}\\ &\qquad + \frac{q_{H_3}+q_{H_4}}{2^{m+1}} - \frac{1}{2} \qquad \text{and}\\ t_B(k) &\leq FO_{time}^{\text{ind-cpa}}(t_A(k), q_{H_3}, q_{H_4}, q_D) = t_A(k) + q_{H_3}t_{enc}\\ &\qquad + O((1+{q_{H_3}}^2+{q_{H_4}}^2+q_D q_{H_3}(1+q_{H_4}))l).\end{aligned}$$

The proof of Claim 6 is given in the full version of this paper. Claim 7 is the revision of Claim 5. We will describe the problems of Claim 5 in the proof of Claim 7.

**Claim 7.** Suppose there is an IND-ID-CCA adversary $A$ that has advantage $\epsilon(k)$ against $\mathsf{Full\text{-}Ident}$ and $A$ runs in time at most $t(k)$. Suppose $A$ makes at most $q_E$ private key extraction queries, at most $q_D$ decryption queries, and at most $q_{H_1}$ queries to the random oracle $H_1$. Then there exists an IND-CCA adversary $B$ against $\mathsf{BasicPub}^{hy}$ with advantage at least $\frac{\epsilon(k)}{1+q_{H_1}+q_D}$ and running time $t(k)+(q_E+q_D)t_{\mathbb{G}_1^*}+q_D t_{dec}+O((1+(q_{H_1}+q_E+q_D)^2)l)$.

*Proof.* The proof is given by solving the problems of the original proof of Claim 5. We show how to construct an IND-CCA adversary $B$ that uses $A$ to gain advantage $\frac{\epsilon(k)}{1+q_{H_1}+q_D}$ against $\mathsf{BasicPub}^{hy}$. Algorithm $B$ works by interacting with $A$ in an IND-ID-CCA game as follows ($B$ simulates the challenger for $A$): The game between algorithm $B$ and its challenger starts with the challenger generating a random public key $K_{pub}$ by running a key generation algorithm of $\mathsf{BasicPub}^{hy}$. Algorithm $B$ receives a public key $K_{pub} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, m, n, P, P_{pub}, Q_{\mathsf{ID}} \rangle$ from its challenger. Algorithm $B$ can make queries to the random oracles $H_2, H_3, H_4$.

**Setup:** Algorithm $B$ gives $A$ the system parameters $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, m, n, P, P_{pub} \rangle$. Algorithm $A$ makes queries to the random oracles $H_1, H_2, H_3, H_4$, where $H_1$ is a random oracle controlled by $B$ as described below.

$H_1$**-queries:** To respond to $A$'s queries, algorithm $B$ maintains two lists, $H_1^{list1}$ and $H_1^{list2}$, of tuples $\langle i, \mathsf{ID}, Q, b \rangle$ as explained below, where $i \in \mathbb{N}$, $\mathsf{ID} \in \{0,1\}^*$, $Q \in \mathbb{G}_1$, and $b \in \mathbb{Z}_q^* \cup \{*\}$. These lists are initially empty. We let $L_1 = \# H_1^{list1}$ and $L_2 = \# H_1^{list2}$ (cf. Remark 1). Algorithm $B$ first selects a random $j \in \{1, 2, \ldots, 1+q_{H_1}+q_D\}$. When $A$ queries the oracle $H_1$ at a point $\mathsf{ID}_i$, algorithm $B$ responds as follows:

1. If there is a tuple $\langle \lambda, \mathsf{ID}_\lambda, Q_\lambda, b_\lambda \rangle$ on the $H_1^{list1}$ or $H_1^{list2}$ such that $\mathsf{ID}_i = \mathsf{ID}_\lambda$ then algorithm $B$ responds with $H_1(\mathsf{ID}_i) = Q_\lambda$ $(\in \mathbb{G}_1^*)$.
2. Otherwise:
   (a) if $L_1 \neq j-1$, algorithm $B$ picks a random $b_i \in \mathbb{Z}_q^*$, sets $Q_i = b_i P$, adds $\langle L_1 + 1, \mathsf{ID}_i, Q_i, b_i \rangle$ to the $H_1^{list1}$, and responds to $A$ with $H_1(\mathsf{ID}_i) = Q_i$.
   (b) if $L_1 = j - 1$, algorithm $B$ sets $Q = Q_{\mathsf{ID}}$, adds $\langle j, \mathsf{ID}_i, Q, * \rangle$ to the $H_1^{list1}$, and responds to $A$ with $H_1(\mathsf{ID}_i) = Q$, where $*$ denotes a special symbol.

**Phase 1: Private key queries.** Let $\mathsf{ID}_i$ be a private key extraction query issued by algorithm $A$. Algorithm $B$ responds to this query as follows:

1. If there is a tuple $\langle \lambda, \mathsf{ID}_\lambda, Q_\lambda, b_\lambda \rangle$ on the $H_1^{list1}$ or $H_1^{list2}$ such that $\mathsf{ID}_i = \mathsf{ID}_\lambda$ then algorithm $B$ obtains a $Q_\lambda \in \mathbb{G}_1^*$ such that $H_1(\mathsf{ID}_i) = Q_\lambda$.
2. Otherwise, algorithm $B$ picks a random $b_i \in \mathbb{Z}_q^*$, sets $Q_i = b_i P$, and adds $\langle L_2 + 1, \mathsf{ID}_i, Q_i, b_i \rangle$ to the $H_1^{list2}$. Algorithm $B$ then obtains a $Q_i \in \mathbb{G}_1^*$ such that $H_1(\mathsf{ID}_i) = Q_i$.
3. Let $\langle i, \mathsf{ID}_i, Q_i, b_i \rangle$ be the corresponding tuple. If the tuple is on the $H_1^{list1}$ and $i = j$, algorithm $B$ picks a random $c' \in \{0,1\}$ and outputs $c'$ as its guess for $c$. Algorithm $B$ then halts.
4. We know $Q_i = b_i P$ for some $b_i \in \mathbb{Z}_q^*$. Algorithm $B$ sets $d_i = b_i\, P_{pub}$, and gives $d_i$ to $A$.

**Phase 1: Decryption queries.** Let $\langle \mathsf{ID}_i, C_i \rangle$ be a decryption query issued by algorithm $A$. Let $C_i = \langle U_i, V_i, W_i \rangle$. Algorithm $B$ responds to this query as follows:

1. If there is a tuple $\langle \lambda, \mathsf{ID}_\lambda, Q_\lambda, b_\lambda \rangle$ on the $H_1^{list1}$ or $H_1^{list2}$ such that $\mathsf{ID}_i = \mathsf{ID}_\lambda$ then algorithm $B$ obtains a $Q_\lambda \in \mathbb{G}_1^*$ such that $H_1(\mathsf{ID}_i) = Q_\lambda$.
2. Otherwise:
   (a) if $L_1 \neq j - 1$, algorithm $B$ picks a random $b_i \in \mathbb{Z}_q^*$, sets $Q_i = b_i P$, and adds $\langle L_1 + 1, \mathsf{ID}_i, Q_i, b_i \rangle$ to the $H_1^{list1}$. Algorithm $B$ then obtains a $Q_i \in \mathbb{G}_1^*$ such that $H_1(\mathsf{ID}_i) = Q_i$.
   (b) if $L_1 = j-1$, algorithm $B$ sets $Q = Q_{\mathsf{ID}}$, and adds $\langle j, \mathsf{ID}_i, Q, * \rangle$ to the $H_1^{list1}$, where $*$ denotes a special symbol. Algorithm $B$ then obtains a $Q \in \mathbb{G}_1^*$ such that $H_1(\mathsf{ID}_i) = Q$.
3. Let $\langle i, \mathsf{ID}_i, Q_i, b_i \rangle$ be the corresponding tuple. If the tuple is on the $H_1^{list1}$ and $i = j$, algorithm $B$ relays the decryption query $C_i$ to the challenger and relays the challenger's response back to $A$.

4. Otherwise, algorithm $B$ runs the algorithm for responding to private key queries to obtain the private key for the public key $\mathsf{ID}_i$. Then use the private key to respond to the decryption query.

**Challenge:** Once algorithm $A$ decides that Phase 1 is over, it outputs a public key $\mathsf{ID}_{ch}$ and two messages $M_0, M_1$ on which it wishes to be challenged. Algorithm $B$ works as follows:

1. Algorithm $B$ gives the challenger $M_0$ and $M_1$ as the messages that it wishes to be challenged on. The challenger responds with a $\mathsf{BasicPub}^{hy}$ ciphertext $C = \langle U, V, W \rangle$ such that $C$ is the encryption of $M_c$ for a random $c \in \{0, 1\}$.
2. Suppose that there is a tuple $\langle i, \mathsf{ID}_i, Q_i, b_i \rangle$ on the $H_1^{list1}$ such that $\mathsf{ID}_{ch} = \mathsf{ID}_i$. (Note that the $\mathsf{ID}_{ch}$ never appears on the $H_1^{list2}$.)
   (a) If $i = j$ then algorithm $B$ obtains a $Q \in \mathbb{G}_1^*$ such that $H_1(\mathsf{ID}_{ch}) = Q$. Algorithm $B$ responds to $A$ with the challenge $C$.
   (b) Otherwise, algorithm $B$ picks a random $c' \in \{0, 1\}$ and outputs $c'$ as its guess for $c$. Algorithm $B$ then halts.
3. Suppose that there is no tuple $\langle i, \mathsf{ID}_i, Q_i, b_i \rangle$ on the $H_1^{list1}$ such that $\mathsf{ID}_{ch} = \mathsf{ID}_i$.
   (a) if $L_1 \neq j - 1$, algorithm $B$ picks a random $c' \in \{0, 1\}$ and outputs $c'$ as its guess for $c$. Algorithm $B$ then halts.
   (b) if $L_1 = j - 1$, algorithm $B$ sets $Q = Q_{\mathsf{ID}}$, and adds $\langle j, \mathsf{ID}_{ch}, Q, * \rangle$ to the $H_1^{list1}$, where $*$ denotes a special symbol. Algorithm $B$ then obtains a $Q \in \mathbb{G}_1^*$ such that $H_1(\mathsf{ID}_{ch}) = Q$. Algorithm $B$ responds to $A$ with the challenge $C$.

**Phase 2: Private key queries.** Algorithm $B$ works as in Phase 1, except for the extraction query for $\mathsf{ID}_{ch}$, which is rejected.

**Phase 2: Decryption queries.** Algorithm $B$ works as in Phase 1, except for the decryption query for $\langle \mathsf{ID}_{ch}, C \rangle$, which is rejected.

**Guess:** Eventually algorithm $A$ produces a guess $c'$ for $c$. Algorithm $B$ outputs $c'$ as its guess for $c$.

*Remark 1.* In the above, we note that $L_1$ implies the number of elements in $H_1^{list1}$ at the time that the underlying query is made or the challenge identity is given. Similarly, $L_2$ implies the number of elements in $H_1^{list2}$ at the time that the underlying query is made. It is easily known that $H_1^{list1}$ is the list that contains the identities that are possible candidates to become the challenge identity, and $H_1^{list2}$ is the list that contains the identities that never become the challenge identity.

In the original proof of Claim 5, algorithm $A$ is required to makes a query to the random oracle $H_1$ at the point $\mathsf{ID}_{ch}$ in order for algorithm $B'$ (IND-CCA adversary) to win the IND-CCA attack game. (This is induced from the fact that algorithm $B'$ selects a random $j$ from the set $\{1, 2, \ldots, q_{H_1}\}$ in the $H_1$-queries stage. See [13] for details.) In general, algorithm $A$, however, does not necessarily issue an $H_1$-query at the point $\mathsf{ID}_{ch}$. For example, we can construct an algorithm $C$ that never makes an $H_1$-query at $\mathsf{ID}_{ch}$ and that can break BasicIdent (see [6] for details of the algorithm) in the sense of IND-ID-CCA. In the challenge stage, algorithm $C$ chooses $\mathsf{ID}_{ch} \in \{0,1\}^*$ and $M_0, M_1 \in \mathcal{M}'$ at random, and it outputs $\langle\, \mathsf{ID}_{ch}, M_0, M_1 \,\rangle$ to its challenger, where $\mathcal{M}'$ denotes the message space of BasicIdent. For the given challenge ciphertext $\langle\, U, V \,\rangle = \langle\, rP, M_b \oplus H_2(\hat{e}(Q_{\mathsf{ID}_{ch}}, P_{pub})^r) \,\rangle$, algorithm $C$ picks a random $\alpha \in \{0,1\}^n$ and makes a decryption query $\langle\, U, \alpha \oplus V \,\rangle$ to obtain $\alpha \oplus M_b$. Thus, algorithm $C$ can always guess $b \in \{0,1\}$ without making an $H_1$-query at $\mathsf{ID}_{ch}$. On the other hand, in our proof, algorithm $B$ selects a random $j$ from the set $\{1, 2, \ldots, 1+q_{H_1}+q_D\}$. When the challenge identity $\mathsf{ID}_{ch}$ is output by $A$, algorithm $B$ runs the algorithm for responding to $H_1$-queries. And $\mathsf{ID}_{ch}$ does not appear in any private key query (i.e., $\mathsf{ID}_{ch}$ never appears on the $H_1^{list2}$.). Hence, the tuple $\langle\, \mathsf{ID}_{ch}, Q, * \,\rangle$ can be always found in the list $H_1^{list1}$ at the end of the guess stage. We note that $L_1 \le 1 + q_{H_1} + q_D$ and $L_2 \le q_E$. Thus, the flaw in the original proof is revised in our proof to obtain the precise security reduction of the BF-IBE scheme. Some other small flaws in the original proof are also revised. (The details of those flaws are omitted in this paper. However, they can be easily known by comparing our proof of Claim 7 and the original proof of Claim 5 in [13].)

It is easy to verify that the amount of running time of $B$ is as claimed: For example, from the specification of $B$, we know that the running time for checking two lists $H_1^{list1}$ and $H_1^{list2}$ is at most $\sum_{i=1}^{1+q_{H_1}+q_E+q_D}(l_1 i + l_2) = l_1(1+q_{H_1}+q_E+q_D)(2+q_{H_1}+q_E+q_D)/2 + l_2(1+q_{H_1}+q_E+q_D)$, where $l_1 = |\mathsf{ID}_i|$ and $l_2 = |i| + |Q_i| + |b_i|$ for the tuples $\langle\, i, \mathsf{ID}_i, Q_i, b_i \,\rangle$ on the lists.

To study $B$'s advantage, we define the following events:

- $\mathsf{E}_1$ is the event that algorithm $A$ issues a private key query $\mathsf{ID}_j$ that corresponds to the tuple $\langle\, j, \mathsf{ID}_j, Q, * \,\rangle$ on the $H_1^{list}$ during Phase 1 or 2.
- $\mathsf{E}_2$ is the event that algorithm $A$ sets the challenge identity $\mathsf{ID}_{ch}$ that does not correspond to the tuple $\langle\, j, \mathsf{ID}_j, Q, * \,\rangle$ on the $H_1^{list}$.
- $\mathsf{G} = \neg\mathsf{E}_1 \wedge \neg\mathsf{E}_2$.

If the event $\mathsf{G}$ is true then algorithm $A$'s view is identical to its view in the real attack. Therefore,

$$\left| \Pr[c = c' \mid \mathsf{G}] - \frac{1}{2} \right| \ge \epsilon(k). \tag{1}$$

Since $\neg\mathsf{E}_2$ implies $\neg\mathsf{E}_1$ (if $\mathsf{ID}_j$ is the challenge identity then $A$ can not ask at any point for its private key), we have

$$\Pr[\mathsf{G}] = \Pr[\neg\mathsf{E}_1 \wedge \neg\mathsf{E}_2] = \Pr[\neg\mathsf{E}_2] \ge \frac{1}{1+q_{H_1}+q_D}. \tag{2}$$

On the other hand, we have

$$\Pr[c = c' \mid \neg\mathsf{G}] = \frac{1}{2} \tag{3}$$

from the specification of $B$. From (1), (2), and (3), we finally have

$$\begin{aligned}\left|\Pr[c=c'] - \frac{1}{2}\right| &= \left|\Pr[c=c' \mid \mathsf{G}]\Pr[\mathsf{G}] + \Pr[c=c' \mid \neg\mathsf{G}]\Pr[\neg\mathsf{G}] - \frac{1}{2}\right| \\ &= \left|\Pr[c=c' \mid \mathsf{G}]\Pr[\mathsf{G}] + \frac{1}{2}(1-\Pr[\mathsf{G}]) - \frac{1}{2}\right| \\ &= \left|\Pr[c=c' \mid \mathsf{G}] - \frac{1}{2}\right|\Pr[\mathsf{G}] \geq \frac{\epsilon(k)}{1+q_{H_1}+q_D}.\end{aligned}$$

This shows that $B$'s advantage is at least $\frac{\epsilon(k)}{1+q_{H_1}+q_D}$ as claimed.

Claims 1, 6 and 7 induce that Full-Ident is IND-ID-CCA secure under the BDH assumption. Namely, Theorem 1 is our result for the security of the BF-IBE scheme.

**Theorem 1.** *Let the hash functions $H_1, H_2, H_3, H_4$ be random oracles. Then* Full-Ident *is IND-ID-CCA secure assuming BDH problem is hard in groups generated by $\mathcal{G}$. Concretely, suppose there is an IND-ID-CCA adversary $A$ that has advantage $\epsilon_A(k)$ against* Full-Ident *and $A$ runs in time at most $t_A(k)$. Suppose $A$ makes at most $q_E$ extraction queries, at most $q_D$ decryption queries, and at most $q_{H_i}$ queries to $H_i$ ($1 \leq i \leq 4$) respectively. Then there is an algorithm $B^*$ that solves BDH problem in groups generated by $\mathcal{G}$ with running time $t_{B^*}(k)$ where:*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{bdh}}_{\mathcal{G},B^*}(k) &\geq 2FO^{\text{ind-cpa}}_{adv}\left(\frac{\epsilon_A(k)}{1+q_{H_1}+q_D}, q_{H_3}, q_{H_4}, q_D\right){q_{H_2}}^{-1}, \\ t_{B^*}(k) &\leq FO^{\text{ind-cpa}}_{time}(t_A(k) + (q_E+q_D)t_{\mathbb{G}_1^*} + q_D t_{dec} \\ &\quad + O((1+(q_{H_1}+q_E+q_D)^2)l), q_{H_3}, q_{H_4}, q_D).\end{aligned}$$

*Here, $l$ denotes the length of the input to algorithm $A$, $t_{\mathbb{G}_1^*}$ denotes the time required for computing a scalar multiplication on $\mathbb{G}_1^*$, and $t_{dec}$ denotes the time required for decryption in* Full-Ident*. The functions $FO^{\text{ind-cpa}}_{adv}$ and $FO^{\text{ind-cpa}}_{time}$ are defined in Claim 6.*

Boneh and Franklin's result and Galindo's result (cf. Section 3.1) respectively induce that

$$\mathsf{Adv}^{\mathrm{bdh}}_{\mathcal{G},B}(k) \geq \frac{\epsilon_A(k)}{e\, q_{H_2}(q_{H_3}+q_{H_4})(1+q_E+q_D)} \tag{4}$$

and

$$\mathsf{Adv}^{\mathrm{bdh}}_{\mathcal{G},B}(k) \geq \frac{\epsilon_A(k)}{q_{H_1}q_{H_2}(q_{H_3}+q_{H_4})}\left(1-\frac{q_E}{q_{H_1}}\right) \tag{5}$$

for sufficiently large $k$ (although they are unreliable as described in Section 3.1 and this section). Theorem 1 induces that

$$\mathsf{Adv}^{\mathrm{bdh}}_{\mathcal{G},B^*}(k) \geq \frac{2\epsilon_A(k)}{q_{H_2}(1+q_{H_1}+q_D)} \tag{6}$$

for sufficiently large $k$. In general, it is difficult to determine which of the given security reductions is the most efficient, because the number of queries to the oracles depends on the (unknown) adversary. Galindo supposes that $q_{H_1} \gg q_E, q_D$ in his revised proof of the security of the BF-IBE scheme (although any explicit reason for this supposition is not given in [13]). In fact, the answer from the private key extraction oracle or the decryption oracle for the query including $\mathsf{ID}_i$ will be useful for the adversary only when he/she knows the value of $Q_{\mathsf{ID}_i}$. From the similar reason, we can suppose that $q_{H_3}, q_{H_4} \gg q_D$. Furthermore, while the denominators of the probabilities in (4) and (5) are represented in terms of a cubic expression of $q_{H_1}$, $q_{H_2}$, $q_{H_3}$, $q_{H_4}$, $q_E$, and $q_D$, the denominator of the probability in (6) is represented in terms of a quadratic expression of $q_{H_1}$, $q_{H_2}$, and $q_D$. From these reasons, we can consider the efficiency of the security reduction to be improved.

## 4 Conclusion

In this work, we have provided a new and correct security reduction for the Boneh and Franklin IBE scheme, showing that it is tigher than had been previously believed. Our corrections and improvements we have presented in this paper can be applied to many other cryptographic schemes [1, 7, 10, 14, 15, 16, 17] that are based on the BF-IBE scheme. However, we cannot say that our new security reduction is efficient enough because it is very worse than the security reductions for many practical public-key encryption schemes. Hence, it is a significant open problem to design a practical IND-ID-CCA secure IBE scheme with a tigher security reduction under a reasonable assumption.

## Acknowledgements

We would like to thank the anonymous reviewers of Indocrypt 2005 for giving helpful comments and suggestions.

## References

1. S. AlRiyami and K.G. Paterson. Certificateless public key cryptography. In *Advances in Cryptology – Asiacrypt 2003*, LNCS 2894, pages 452–473. Springer-Verlag, 2003.
2. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology – Crypto '98*, LNCS 1462, pages 26–45. Springer-Verlag, 1998.
3. M. Bellare and P. Rogaway. Random oracles are practical – a paradigm for designing efficient protocol. In *First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
4. D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In *Advances in Cryptology – Eurocrypt 2004*, LNCS 3027, pages 223–238. Springer-Verlag, 2004.
5. D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology – Crypto 2004*, LNCS 3152, pages 443–459. Springer-Verlag, 2004.

6. D. Boneh and M. Franklin. Identity-based encryption from the Weil paring. In *Advances in Cryptology – Crypto 2001*, LNCS 2139, pages 213–229. Springer-Verlag, 2001.
7. X. Boyen. Multipurpose identity-based signcryption: a swiss army knife for identity-based cryptography. In *Advances in Cryptology – Crypto 2003*, LNCS 2729, pages 383–399. Springer-Verlag, 2003.
8. R. Canetti and O. Goldreich and S. Halevi. The random oracle methodology, revisited. In *Proceedings of the 30th ACM STOC '98*, pages 209–218, 1998.
9. R. Canetti and S. Halevi and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology – Eurocrypt 2004*, LNCS 3027, pages 207–222. Springer-Verlag, 2004.
10. Z. Cheng and R. Comley. Efficient certificateless public key encryption. In *IACR Cryptology ePrint Archive*, 2005/012. Available from `http://eprint.iacr.org/2005/012`.
11. C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 26–28, 2001.
12. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology – Crypto '99*, LNCS 1666, pages 537–554. Springer-Verlag, 1999.
13. D. Galindo. Boneh-Franklin identity based encryption revisited. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, ICALP 2005. Also available on IACR Cryptology ePrint Archive, Report 2005/117.
14. C. Gentry. Certificate-based encryption and the certificate revocation problem. In *Advances in Cryptology – Eurocrypt 2003*, LNCS 2656, pages 272–293. Springer-Verlag, 2003.
15. C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Advances in Cryptology – Asiacrypt 2002*, LNCS 2501, pages 548–566. Springer-Verlag, 2002.
16. J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology – Eurocrypt 2002*, LNCS 2332, pages 466–481. Springer-Verlag, 2002.
17. B. Lynn. Authenticated identity-based encryption. In *IACR Cryptology ePrint Archive*, 2002/072. Available from `http://eprint.iacr.org/2002/072`.
18. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – Crypto '84*, LNCS 196, pages 47–53. Springer-Verlag, 1984.
19. V. Shoup. OAEP reconsidered. In *Advances in Cryptology – Crypto 2001*, LNCS 2139, pages 239–259. Springer-Verlag, 2001.
20. B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology – Eurorypt 2005*, LNCS 3494, pages 114–127. Springer-Verlag, 2005.

# Short Undeniable Signatures Without Random Oracles: The Missing Link

Fabien Laguillaumie[1] and Damien Vergnaud[2]

[1] Projet TANC - INRIA Futurs, Laboratoire d'informatique (LIX)
École polytechnique, 91128 Palaiseau cedex, France
laguillaumie@lix.polytechnique.fr
[2] Laboratoire de Mathématiques Nicolas Oresme,
Université de Caen, Campus II,
B.P. 5186, 14032 Caen Cedex, France
vergnaud@math.unicaen.fr

**Abstract.** We introduce a new undeniable signature scheme which is existentially unforgeable and anonymous under chosen message attacks in the standard model. The scheme is an embedding of Boneh and Boyen's recent short signature scheme in a group where the *decisional Diffie-Hellman problem* is assumed to be difficult. The anonymity of our scheme relies on a decisional variant of the *strong Diffie-Hellman assumption*, while its unforgeability relies on the strong Diffie-Hellman assumption.

## 1 Introduction

We design new undeniable signatures. Our approach is both practical and theoretical: we build a very efficient protocol with short signatures and analyze its security in the complexity theory setting (*i.e* with reductionist proofs). The security (in the sense of unforgeability and anonymity) relies on *strong Diffie-Hellman assumptions* in the standard model. It is worth noting that the new mechanism is the first efficient scheme proven secure without any random oracles.

*Related work.* The *self-authenticating* property of digital signatures can be suitable for many applications such as dissemination of public-key certificates or official announcements, but seems undesirable in personally or commercially sensitive applications. Therefore it may be preferable to put some restrictions on this property to prevent misuse of signatures. Undeniable signatures were introduced in 1989 by Chaum and van Antwerpen [6] to limit the self-authenticating property of digital signatures. In this setting, one has to interact with the signer in order to be convinced of the validity of a given signature. The security of the seminal protocol relies on the discrete logarithm problem, but suffers from the fact that the interactive protocols were not zero-knowledge. In 1990, Chaum [5] improved the initial proposal by providing a zero-knowledge version. The security of Chaum and van Antwerpen's undeniable signatures was eventually proven by Okamoto and Pointcheval in 2001 [20], using the so-called *gap-problems*. In

© Springer-Verlag Berlin Heidelberg 2005

[21], Ogata, Kurosawa, and Heng showed that the security can in fact be proven under a classical computational assumption. This concept has been investigated for years, and many proposals appear in the literature. In 1991, the concept has been refined by giving the possibility to transform an undeniable signature into a *self-authenticating* signature. These *convertible undeniable signatures*, proposed in [3] by Boyar, Chaum, Damgård and Pedersen, provide individual and universal conversions of the signatures. They were broken and repaired in 1996 by Michels, Petersen and Horster [17]. Several schemes were subsequently proposed, based on well-known signatures [10, 9, 8]. Recently, an identity-based undeniable signature scheme built on bilinear maps was proposed by Libert and Quisquater [16]. Monnerat and Vaudenay [18, 19] proposed short undeniable signatures based on characters (without the conversion property). Finally, we extended in [15] the concept of convertible undeniable signatures by giving the signer the ability to convert signatures pertaining to a specific time period.

*Our contributions.* In groups where there exists an oracle for the decisional Diffie-Hellman problem, Chaum and van Antwerpen's undeniable signatures become self-authenticating. They were revisited by Boneh, Lynn and Shacham (BLS) in 2001 [2] and considered on groups where there exists an admissible bilinear map. An elegant variant of these signatures, still pairing-based, was introduced in 2004 by Boneh and Boyen [1] (and also by Zhang, Safavi-Naini and Susilo [23]). Its unforgeability was proven in the standard model. Contrary to the latter approach, in this article, we remove the bilinear map from Boneh-Boyen signatures to obtain the first efficient undeniable signature scheme without random oracle.

In [12], Goldwasser and Waisbard proposed *designated confirmer signatures* without random oracles. Their techniques could be extended to construct secure undeniable signatures. Indeed, this transformation is straightforward, since their general construction remains secure if the designated confirmer is the signer himself. Goldwasser and Waisbard do not give efficient disavowal protocols for their instanciations. They argued for designated confirmer signatures there is no need for such a protocol. However, to get an efficient complete undeniable signature protocol, there are a number of non-trivial details that would need to be worked out. The resulting schemes will be far less efficient than our proposal.

Our undeniable signatures are to Chaum and van Antwerpen undeniable signatures what Boneh-Boyen's signatures are to BLS. In the dual way, the new scheme is to Boneh-Boyen's scheme what Chaum and van Antwerpen's construction is to BLS.

The security of previous proposals of undeniable signatures is carried in the random oracle model and therefore is only heuristic. Like Boneh-Boyen's scheme, the security of our protocol does not rely on any idealized primitive but needs stronger computational assumptions. As pointed out in [14], the confirming and denying protocols are important elements in the security analysis of an undeniable signature scheme. The main difficulties to study our scheme in the standard security model, arise in the simulation of the *interactive* confirmation and denying protocols in the reductionist proof. The present paper provides the first security analysis for undeniable signatures in this interactive setting.

## 2 Preliminaries

### 2.1 Undeniable Signatures

**Definition 1 (Undeniable Signatures).** *Given an integer $k$, an* undeniable signature scheme *US with security parameter $k$ is defined by the following:*

- *a* **common parameter generation algorithm** *US.Setup: it is a probabilistic algorithm which takes as input $k$. The outputs are* the public parameters*;*
- *a* **key generation algorithm for the signers** *US.SKeyGen: it is a probabilistic algorithm which takes as input the public parameters and outputs a pair of keys $(pk, sk)$;*
- *a* **key generation algorithm for a verifiers** *US.VKeyGen: it is a probabilistic algorithm which takes as input the public parameters and outputs a pair of keys $(pk, sk)$;*
- *a* **key registration protocol** *US.Register is a protocol between a "key registration authority" (KRA) and a verifier with common input the public parameters. At the end, the KRA outputs a pair $(pk, \mathsf{notif})$ where $pk$ is the verifier's public key and $\mathsf{notif} \in \{0,1\}^*$ is a key registration authorization decision.*
- *a* **signing algorithm** *US.Sign: it is a probabilistic algorithm which takes as input a message $m$, a secret key $sk$ and the public parameters. The output $\sigma$ is* an undeniable signature on *$m$;*
- **confirming/denying protocols** *US.{Confirm.Deny}: they are protocols which take as input a message $m$, a putative undeniable signature $\sigma$ on $m$, a pair of keys $(pk, sk)$ and the public parameters. The output is a (possibly non-interactive) non-transferable proof that $\sigma$ is actually a valid/invalid undeniable signature on $m$, with respect to the key $pk$;*

*and must satisfy the following properties (formally defined below):*

1. *correctness and soundness: the confirming and denying protocols and the verifying algorithms are complete and sound, where completeness means that valid (invalid) signatures can always be proved valid (invalid), and soundness means that no valid (invalid) signature can be proved invalid (valid);*
2. *unforgeability: given a public key, it is computationally infeasible, without the knowledge of the corresponding secret key to produce an undeniable signature that is accepted by the verification algorithm or by the confirming protocol;*
3. *anonymity: given a message $m$ and an undeniable signature $\sigma$ on $m$, it is computationally infeasible to find which secret key was used to generate $\sigma$;*
4. *non-transferability: a verifier participating in an execution of the confirming/denying protocols does not obtain information that could be used to convince a third party about the validity/invalidty of a signature.*

*Remark 1.* The aim of the protocol Register is to force the verifiers to "know" the secret key corresponding to their public key, in order to enforce the non-transferability property. We assume for simplicity that the verifier just reveals his key pair $(pk, sk)$ and the key registration authority authorizes it if and only if $(pk, sk) \in \mathsf{US.VKeyGen}(\mathtt{params})$.[1]

[1] This can always be done, since we can assume that the secret key contains the randomness input used to generate it.

## 2.2 Security Model

**Anonymity.** The notion of *anonymity* under a chosen message attack (Ano-CMA) was precisely defined by Galbraith and Mao in [8]. An Ano-CMA-adversary $\mathcal{A}$ runs in two stages. In the `find` stage, it takes as input two public keys $pk_0$ and $pk_1$ and outputs a message $m^\star$ (and some state information $s$). In the `guess` stage it gets a challenge undeniable signature $\sigma^\star$ formed by signing the message $m^\star$ at random under one of the two keys, and must say which key was chosen. In both stages, the adversary has access to the signing oracles $\Sigma_0$, $\Sigma_1$, to the confirming oracles $\Upsilon_{C,0}$ and $\Upsilon_{C,1}$ (with registered verifying keys) and to the denying oracles $\Upsilon_{D,0}$ and $\Upsilon_{D,1}$ (with registered verifying keys). The only restriction is that he cannot query the couple $(m^\star, \sigma^\star)$ on the confirming/denying oracles.

**Definition 2 (Anonymity).** *Let $\mathsf{US}$ be an undeniable signature scheme and let $\mathcal{A}$ be an Ano-CMA-adversary against $\mathsf{US}$. We consider the following two random experiments, for $r \in \{0,1\}$, where $k$ is a security parameter:*

```
$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{ano\text{-}cma}-r}_{\mathsf{US},\mathcal{A}}(k)}$
params $\xleftarrow{R} \mathsf{US.Setup}(k)$
$(pk_0, sk_0) \xleftarrow{R} \mathsf{US.KeyGen}(\text{params})$
$(pk_1, sk_1) \xleftarrow{R} \mathsf{US.KeyGen}(\text{params})$
$(m^\star, s) \leftarrow \mathcal{A}^{\Sigma_0,\Sigma_1,\Upsilon_{C,0},\Upsilon_{C,1},\Upsilon_{D,0},\Upsilon_{D,1}}(\texttt{find}, \text{params}, pk_0, pk_1)$
$\sigma^\star \leftarrow \mathsf{US.Sign}(\text{params}, m, sk_r)$
$d \leftarrow \mathcal{A}^{\Sigma_0,\Sigma_1,\Upsilon_{C,0},\Upsilon_{C,1},\Upsilon_{D,0},\Upsilon_{D,1}}(\texttt{guess}, \text{params}, pk_0, pk_1, m^\star, \sigma^\star)$
Return $d$
```

*We define the* advantage *of $\mathcal{A}$* via*:*

$$\mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{US},\mathcal{A}}(k) = \left|\Pr\left[\mathbf{Exp}^{\mathsf{ano-cma}-1}_{\mathsf{US},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{ano-cma}-0}_{\mathsf{US},\mathcal{A}}(k) = 1\right]\right|.$$

*Given $(k,\tau) \in \mathbb{N}^2$ and $\varepsilon \in [0,1]$, the scheme $\mathsf{US}$ is said to be $(k,\tau,\varepsilon)$-Ano-CMA secure, if no Ano-CMA-adversary $\mathcal{A}$ running in time $\tau$ has an advantage $\mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{US},\mathcal{A}}(k) \geq \varepsilon$.*

**Security against existential forgery under chosen message attack.** Security for digital signatures was defined by Goldwasser, Micali and Rivest [11] as *existential forgery against adaptive chosen message attacks* (EF-CMA). For undeniable signatures, unforgeability security is defined along the same lines, with the notable difference that, while mounting a chosen-message attack, we suppose that the adversary is allowed to query a confirming (*resp.* a denying) oracle $\Upsilon_C$ (*resp.* $\Upsilon_D$) on any couple message/signature of his choice, in addition to the classical access to the signing oracle $\Sigma$. As usual, in the adversary answer, there is the natural restriction that in the returned couple message/signature $(m^\star, \sigma^\star)$, the message $m^\star$ has not been queried to the oracle $\Sigma$.

**Definition 3 (Unforgeability).** *Let $\mathsf{US}$ be an undeniable signature scheme and let $\mathcal{A}$ be an EF-CMA-adversary against $\mathsf{US}$. We consider the following random experiment, where $k$ is a security parameter:*

$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{ef-cma}}_{\mathit{US},\mathcal{A}}(k)}$

params $\xleftarrow{R}$ $\mathit{US.Setup}(k)$
$(pk, sk) \xleftarrow{R} \mathit{US.KeyGen}(\text{params})$
$(m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\Sigma,\Upsilon_C,\Upsilon_D}(\text{params}, pk)$
*Return* 1 *if* $\sigma^\star$ *is a valid signature on* $m^\star$
0 *otherwise*

*We define the* success *of* $\mathcal{A}$ via $\mathbf{Succ}^{\textit{ef-cma}}_{\mathit{US},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\textit{ef-cma}}_{\mathit{US},\mathcal{A}}(k) = \mathtt{valid}\right]$.

*Given* $(k, \tau) \in \mathbb{N}^2$ *and* $\varepsilon \in [0, 1]$*, the scheme* $\mathit{US}$ *is said to be* $(k, \tau, \varepsilon)$*-EF-CMA secure, if no EF-CMA-adversary* $\mathcal{A}$ *running in time* $\tau$ *has a success* $\mathbf{Succ}^{\textit{ef-cma}}_{\mathit{US},\mathcal{A}}(k) \geq \varepsilon$*.*

### 2.3 Proof of Knowledge

We cannot replace the zero-knowledge interactive proofs by non-interactive non-transferable proofs, to obtain the security results in the standard model. As far as we know, all these non-interactive proofs are either highly inefficient or obtained by applying the Fiat-Shamir heuristic to interactive designated-verifier proofs, and therefore their security relies on the random oracle paradigm.

Let $\mathbb{G}$ be a group. To confirm or deny that a bit string is a signature in the new undeniable signature scheme, it is necessary to prove that a given quadruple $(L, M, N, O) \in \mathbb{G}^4$ is a Diffie-Hellman quadruple (or not). To face *blackmailing* or *mafia* attacks against our undeniable signatures, we use interactive designated verifier proofs, as introduced in [13] by Jakobsson, Sako, and Impagliazzo, in Chaum's proofs of equality (*cf.* Fig. 1) and inequality (*cf.* Fig. 2) of discrete logarithm of [5]. The idea is to replace the generic commitment scheme by a trapdoor commitment, as defined in [4], and using classical techniques, the proofs are readily seen to be complete, sound, and above all non-transferable. The

| Prover | | Verifier |
|---|---|---|
| | | $(u, v) \in_R [\![1, q-1]\!]^2$ |
| | $\xleftarrow{\quad C \quad}$ | $C = uL + vN$ |
| $r \in_R [\![1, q-1]\!]$ | | |
| $R_1 = C + rP_V$ | | |
| $R_2 = xR_1$ | $\xrightarrow{\quad R_1, R_2 \quad}$ | |
| | $\xleftarrow{\quad u, v \quad}$ | |
| $C \overset{?}{=} uL + vN$ | $\xrightarrow{\quad r \quad}$ | |
| | | $R_1 \overset{?}{=} C + rP_V$ |
| | | $R_2 \overset{?}{=} (u + yr)M + vO$ |

**Fig. 1.** ZKIP protocol to prove that $x = \log_L(M) = \log_N(O)$

$$\begin{array}{rcl}
\text{Prover} & & \text{Verifier} \\ \hline
& & s \in_R [\![0,\lambda]\!] \text{ and } u \in_R [\![1,q-1]\!] \\
& \xleftarrow{\qquad C, C' \qquad} & C = uL + sN \text{ and } C' = uM + sO \\
\text{find } s' \text{ such that} & & \\
(xC - C') = s'(xN - O) & & \\
r \in_R [\![1,q-1]\!] & & \\
C'' = s'L + rP_{\mathcal{V}} & \xrightarrow{\qquad C'' \qquad} & \\
& \xleftarrow{\qquad u \qquad} & \\
C \overset{?}{=} uL + s'N & \xrightarrow{\qquad (s', r) \qquad} & C'' \overset{?}{=} s'L + rP_{\mathcal{V}} \text{ and } s \overset{?}{=} s' \\ \hline
\end{array}$$

**Fig. 2.** ZKIP protocol to prove that $x = \log_L(M) \neq \log_N(O)$

**Table 1.** Some values of $\lambda$ and computational workload in the proof of inequality

| security | | $\lambda$ | $\lambda = 31$ | $\lambda = 1023$ |
|---|---|---|---|---|
| $2^{30}$ | #iterations | $30/\log_2(\lambda+1)$ | 6 | 3 |
| | #exp. | $\lambda/2 \times$ #iterations | 93 | 1534.5 |
| $2^{80}$ | #iterations | $80/\log_2(\lambda+1)$ | 16 | 8 |
| | #exp. | $\lambda/2 \times$ #iterations | 248 | 4092 |

protocols, involve a point $P_{\mathcal{V}} = yL$ where $y$ is the secret key of the verifier, and the prover must be convinced that $P_{\mathcal{V}}$ is well-formed (in the undeniable signature scheme, the registration protocol is used to force the users to know the secret-key corresponding to their public key).

In both protocols, the prover is given $(L, M, N, O)$, and he knows $x = \log_L(M)$. As argued in [5], in the proof of inequality, the prover can cheat with probability $(\lambda + 1)^{-1}$. This leads to the table 1 with examples for suitable $\lambda$ together with the round and computational complexities.

## 2.4 Underlying Problems

The security of asymmetric cryptographic tools relies on assumptions about the hardness of certain algorithmic problems. Throughout the paper $\mathbb{G}$ denotes an additive group of prime order $q$ (*e.g.* the group of points of an elliptic curve over a finite field, a subgroup of the multiplicative group of a finite field). Our scheme relies on the difficulty of the algorithmic problems described below in $\mathbb{G}$ but on no other special property. Therefore, we choose not to pin down a specific group and to describe the protocol in a generic way:

**Definition 4.** *A* prime-order-group-generator *(POGG) is a probabilistic algorithm that takes a security parameter $k$ as input and outputs a pair $(q, \mathbb{G})$ where $q$ is a prime with $2^k < q < 2^{k+1}$, $\mathbb{G}$ is a group of order $q$.*

Let $P$ be a generator of $\mathbb{G}$. In [1], Boneh and Boyen introduced a new computational problem in a bilinear context. For our purpose, we consider this problem in the classical discrete log setting, *i.e* without bilinear map.

**$\ell$-Computational Strong Diffie-Hellman ($\ell$-CSDH):** let $x$ be in $[\![1, q-1]\!]$. Given an integer $\ell \in \mathbb{N}$ and $(P, xP, x^2P, \ldots, x^\ell P) \in \mathbb{G}^{\ell+1}$, compute a pair $\big((x+h)^{-1}P, h\big)$ in $\mathbb{G} \times [\![1, q-1]\!]$ for some $h \in [\![1, q-1]\!]$.

The invisibility of our protocol relies on the decisional variant of this problem:

**$\ell$-Decisional Strong Diffie-Hellman ($\ell$-DSDH):** let $x$ be in $[\![1, q-1]\!]$. Given an integer $\ell \in \mathbb{N}$ and $(P, xP, x^2P, \ldots, x^\ell P) \in \mathbb{G}^{\ell+1}$, and a pair $(Q, h)$ in $\mathbb{G} \times [\![1, q-1]\!]$ for some $h \in [\![1, q-1]\!]$, decide whether $Q = (x+h)^{-1}P$.

In [20], Okamoto and Pointcheval proved the security of the FDH variant of Chaum and van Antwerpen's undeniable signatures by introducing a new class of computational problems, called *gap problems.* In [21], Ogata, Kurosawa and Heng proved that the unforgeability of the protocol is equivalent to the classical CDH problem. In the context of undeniable signatures, the confirming and denying protocols can be executed on any message/putative signature chosen by the adversary. To take into account this kind of oracle access, we have to introduced a *gap-variant* of the Strong Diffie-Hellman problem if we do not want to lose a large factor in the unforgeability security reduction to CSDH.

**$\ell$-Gap Strong Diffie-Hellman ($\ell$-GSDH):** let $x$ be in $[\![1, q-1]\!]$. Given an integer $\ell \in \mathbb{N}$ and $(P, xP, x^2P, \ldots, x^\ell P) \in \mathbb{G}^{\ell+1}$, compute a pair $\big((x+h)^{-1}P, h\big)$ in $\mathbb{G} \times [\![1, q-1]\!]$ for some $h \in [\![1, q-1]\!]$, with the help of a $\ell$-DSDH oracle.

## 3 Short Undeniable Signatures Without Random Oracles

### 3.1 The New Scheme

In this section, we describe our new undeniable signature scheme, parameterized by a prime-order-group-generator $\mathsf{Gen}$. Note, that as mentioned above, for this basic version of the scheme, we use the direct key registration.

Common parameter generation algorithm $\mathsf{US.Setup}$: on input a security parameter $k$, the algorithm $\mathsf{Gen}(k)$ is run to produce a pair $(q, \mathbb{G})$. An element $P$ is picked at random in $\mathbb{G} \setminus \{0_{\mathbb{G}}\}$ and the public parameters are $(q, \mathbb{G}, P)$.

Key generation algorithm for the signers $\mathsf{US.SKeyGen}$: Alice picks at random her secret key $(a_1, a_2) \in [\![1, q-1]\!]^2$ and sets $(P_1, P_2)$ as her public key, with $P_1 = a_1 P$ and $P_2 = a_2 P$.

Key generation algorithm for the verifiers $\mathsf{US.VKeyGen}$: Bob picks at random his secret key $b \in [\![1, q-1]\!]$ and sets $P_B = bP$ as his public key.

Signing algorithm $\mathsf{US.Sign}$: to sign a message $m \in [\![1, q-1]\!]$, Alice picks at random $r \in [\![1, q-1]\!]$ and sets $S = (a_1 + m + r a_2)^{-1} P$. The signature is $\sigma = (S, r)$.

Confirming/Denying protocol $\mathsf{US.\{Confirm, Deny\}}$: given a message $m$ and a putative signature $\sigma = (S, r)$ on $m$, Alice proves to Bob (who has published a registered valid public key $P_B$) that $\log_S(P - mS) = \log_P(P_1 + rP_2)$ or not, using the protocols described in section 2.3. For the sake of simplicity, we suppose that the signer do not interleave several instances asynchronously nor concurrently.

*Remark 2.* The notion of on-line/off-line signatures was introduced by Even, Goldreich and Micali [7]. The idea is to generate signatures in two phases. The first one is performed off-line (*i.e.* before the message to be signed is given) and the second phase is performed on-line (once the message to be signed is known). On-line/Off-line signatures are useful since in many applications the signer has a very limited response time once the message is presented but he can carry out costly computations between consecutive signing requests.

Using the *sign and switch* paradigm, we can convert our undeniable signature scheme into a highly efficient on-line/off-line scheme. The signer computes off-line $S = (a_1 + t)^{-1}P$ for a random $t \in [\![1, q-1]\!]$. Once he is given the message $m$, the signature is completed with $r = a_2^{-1}(t - m)$ where $a_2^{-1} \mod q$ can also be precomputed. The on-line signature completion procedure then amounts to computing a hash value, a substraction and a multiplication modulo $q$.

*Remark 3.* One may require, of course, unforgeability and anonymity of the undeniable signatures, even against the key registration authority. To achieve this, one can replace the direct key registration protocol by a zero-knowledge proof of knowledge of the verifier's secret key (using for instance the Schnorr proof of knowledge of discrete logarithms [22]). The unforgeability and the anonymity of the scheme, can still be proved in the standard security model (by using rewinding techniques). Details on the security arguments will be given in the full version of the paper.

## 3.2 Security Results

*Anonymity.* For any $\mathsf{Ano\text{-}CMA}$ adversary $\mathcal{A}$, we denote by $\mathsf{Bad}_{\mathcal{A}}$ the event that $\mathcal{A}$ queries a valid signature to the confirming/denying oracle, which has not been obtained from the signing oracle.

**Proposition 1.** *Let $\mathsf{Gen}$ be a $\mathsf{POGG}$ and let $\mathsf{US}$ be the associated undeniable signature scheme. For any $\mathsf{Ano\text{-}CMA}$ adversary $\mathcal{A}$ against $\mathsf{US}$, with security parameter $k$, which has advantage $\varepsilon = \mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{US},\mathcal{A}}(k)$, running time $\tau$, making $q_\Sigma$ queries to the signing oracle, $q_C$ to the confirming oracle, $q_D$ to the denying oracle and registers up to $q_{\mathcal{R}}$ keys, there exists an adversary $\mathcal{B}$ against $(q_\Sigma + 1)$-DSDH of advantage $\varepsilon' = \mathbf{Adv}^{(\mathsf{q}_\Sigma+1)-\mathsf{dsdh}}_{\mathsf{Gen},\mathcal{B}}(k)$ and running time $\tau'$, such that $\varepsilon' \geq \varepsilon/2 - (q_\Sigma + 2)2^{-k} - \Pr[\mathsf{Bad}]$ and $\tau' \leq q_C\tau + q_\Sigma(q_\Sigma T_{\mathbb{G}} + O(1))$, where $T_{\mathbb{G}}$ denotes the time complexity to perform a scalar multiplication in $\mathbb{G}$.*

*Proof.* Let $k$ be a security parameter, $(q, \mathbb{G})$ be a couple generated by $\mathsf{Gen}$. We consider a random instance of $\ell$-DSDH denoted by $(P, xP, x^2P, \ldots, x^\ell P, Q, h)$ and we may assume that $q_\Sigma = \ell + 1$. We denote by $A_i$ the point $x^iP$, for all $i \in [\![0, q_\Sigma - 1]\!]$. We construct a simulation which solves this instance.

$\mathsf{Game}_0$ We consider an $\mathsf{Ano\text{-}CMA}$-adversary $\mathcal{A}$ with advantage $\mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{US},\mathcal{A}}(k)$, within time $\tau$. The key generation algorithm is run twice to produces two pairs of keys $(pk_0, sk_0)$ and $(pk_1, sk_1)$. In his first stage, the adversary $\mathcal{A}$ is fed with $pk_0$ and $pk_1$, and, querying the signing oracles $\Sigma_0$ and $\Sigma_1$, and the confirming/denying oracles $\Upsilon_{C,0}$, $\Upsilon_{C,1}$, $\Upsilon_{D,0}$ and $\Upsilon_{D,1}$, outputs a message $m^\star$. A challenger picks $b^\star \in \{0,1\}$ at random and queries the signing oracle $\Sigma_{b^\star}$ on $m^\star$ and sets the answer as $\sigma^\star$. In its second stage the attacker is given $\sigma^\star$ and has once more a permanent access to the oracles, with the natural restriction not to query the challenge signature on the confirming/denying oracles. We denote by $q_\Sigma$ the number of queries to the signing oracle and $q_C$ to the confirming oracle and $q_D$ to the denying oracle.

In any game $\mathsf{Game}_i$, we denote by $\mathsf{Guess_i}$ the event $b^\star = b$. By definition, $|2\Pr[\mathsf{Guess}_0] - 1| = \mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{USBM},\mathcal{A}}(k)$.

$\mathsf{Game}_1$ First, $\mathcal{B}$, picks at random $(\alpha, a_0, a_1) \in [\![1, q-1]\!]^3$, initializes a counter $c$ to the value 1 and two lists $\Sigma\text{-}\mathsf{List} = \{\}$ and $\overline{\Sigma}\text{-}\mathsf{List} = \{\}$. $\mathcal{B}$ prepares $q_\Sigma$ random elements $h_i \in [\![1, q-1]\!]$, for $i \in [\![1, q_\Sigma]\!]$. If $h \in \{h_1, \ldots, h_{q_\Sigma}\}$ or $(\alpha h \mod q) \in \{h_1, \ldots, h_{q_\Sigma}\}$, $\mathcal{B}$ aborts: this happens with probability at most $q_\Sigma 2^{1-k}$. $\mathcal{B}$ computes the following polynomial:

$$f(y) = \prod_{i=1}^{q_\Sigma}(y + h_i) = \sum_{i=0}^{q_\Sigma} \alpha_i y^i \in \mathbb{F}_q[y], \text{and the points } P = \sum_{i=0}^{q_\Sigma} \alpha_i A_i = f(x)P,$$

$$P_0 = \sum_{i=1}^{q_\Sigma+1} \alpha_{i-1} A_i = xf(x)P = xP' \text{ and } P_1 = \alpha P_0 = \alpha x P'.$$ Finally $\mathcal{B}$ sets $\mathtt{params} = (q, \mathbb{G}, P')$, $pk_0 = (P_0, a_0 P')$, $pk_1 = (P_1, a_1 P')$ and feeds $\mathcal{A}$ with $pk_0$ and $pk_1$. The distribution of $(pk_0, pk_1)$ is unchanged since $(A_0, \ldots, A_\ell, Q, h)$ is a random instance of the $\ell$-DSDH problem and $(\alpha, a_0, a_1)$ is picked at random. Therefore we have $\Pr[\mathsf{Guess}_1] = \Pr[\mathsf{Guess}_0]$.

$\mathsf{Game}_2$ From this game, $\mathcal{B}$ performs a specific stage that allows it to retrieve each verifier secret key $y$. If the direct key registration is used, then this is straightforward, otherwise, $\mathcal{B}$ might have to replay the simulation once with the same random tape such that monitoring the re-registration of $P_\mathcal{A}$ by $\mathcal{A}$ allows to extract $y$. In our case, this simulation is obviously perfect, therefore we obtain $\Pr[\mathsf{Guess}_2] = \Pr[\mathsf{Guess}_1]$.

$\mathsf{Game}_3$ Now $\mathcal{B}$ simulates the signing oracles. It initializes a counter to $c = 1$, and for each new request $m \in \{0,1\}^*$, $\mathcal{B}$ constructs this polynomial of $\mathbb{F}_q[y]$: $f_c(y) = \dfrac{f(y)}{y + h_c} = \prod_{\substack{j=1\\ j\neq c}}^{q_\Sigma}(y + h_j) = \sum_{j=0}^{q_\Sigma - 1} \beta_j^{(c)} y^j$ and then computes

$$S_c = \frac{1}{\alpha^b} \sum_{j=0}^{q_\Sigma - 1} \beta_j^{(c)} A_j = \frac{1}{\alpha^b (x + h_c)} P'$$ where $\Sigma_b$, with $b \in \{0,1\}$ is the oracle queried. Then $\mathcal{B}$ sets $r_c = (h_c \alpha^b - m_c) a_b^{-1}$. If $r_c = 0$, then $\mathcal{B}$ aborts, else it outputs $(S_c, r_c)$ as a valid signature on $m_c$ for the public key $pk_b$ and adds $(m_c, (S_c, r_c), b)$ in the $\Sigma\text{-}\mathsf{List}$. $\mathcal{B}$ increments the counter. During its whole execution, $\mathcal{B}$ reports failure in the signing simulation with prob-

ability at most $q_\Sigma 2^{-k}$. This game perfectly simulates the signing oracle if it does not abort. Therefore $|\Pr[\mathsf{Guess}_3] - \Pr[\mathsf{Guess}_2]| \leq q_\Sigma 2^{-k}$.

$\mathsf{Game}_4$ When the adversary queries the confirming oracle $\Upsilon_{C,b}$ with $b \in \{0,1\}$ on a putative signature $\sigma$ on $m$, $\mathcal{B}$ checks whether $(m,\sigma,b)$ appears in the $\Sigma$-List. If not, $\mathcal{B}$ adds this signature $\sigma$ in the $\overline{\Sigma}$-List and outputs `Invalid`. Otherwise $\mathcal{B}$ computes $R = P' - mS$ and $Q = P_b + ra_bP'$ where $\sigma = (S,r)$ and simulates the proof of equality $\log_S(R) = \log_{P'}(Q)$ as follows : $\mathcal{A}$ sends a commitment $C$ to $\mathcal{B}$, who picks $\gamma \in [\![1, q-1]\!]$ at random and computes $R_1 = \gamma P'$ and $R_2 = \gamma Q$ and sends $(R_1, R_2)$ to $\mathcal{A}$ who decommits $(u,v)$ and $\mathcal{B}$ verifies that $C = uP' + vR$. If it is the case, $\mathcal{B}$ rewinds $\mathcal{A}$, and resets the simulation with the same random tape. He replays the same simulation up to the moment where $\mathcal{A}$ sends $C$. $\mathcal{B}$, now that he knows $u$, $v$, and $y$, picks $r \in [\![1, q-1]\!]$ at random and computes $R_1 = C + rP_{\mathcal{A}}$ (where $P_{\mathcal{A}} = yP'$) and $R_2 = (u+yr)Q + vS$, and sends it to $\mathcal{A}$, which accepts the proof.

When the adversary queries the denying oracle $\Upsilon_{D,b}$ with $b \in \{0,1\}$ on a putative signature $\sigma = (S,r)$ on $m$, $\mathcal{B}$ verifies that $\sigma$ does not appear in the $\Sigma$-List (if it does it outputs `Valid`) and updates the $\overline{\Sigma}$-List with $\sigma$. Then $\mathcal{B}$ computes $R = P' - mS$ and $Q = P_b + ra_bP'$ and simulates the denying protocol as follows : when $\mathcal{A}$ sends $(C, C')$ to him, $\mathcal{B}$ picks randomly $\tilde{r} \in [\![0, \lambda]\!]$ and sends to $\mathcal{A}$ the point $C'' = \tilde{r}P'$. $\mathcal{A}$ sends him back $u$, and $\mathcal{B}$ looks for $s' \in [\![0, \lambda]\!]$ such that $C = uP' + s'R$. If he does not find such an $s'$, he aborts. Otherwise $\mathcal{B}$ computes $r = (\tilde{r} - s')y^{-1}$ so that $C'' = s'P' + rP_{\mathcal{A}}$ (where $P_{\mathcal{A}} = yP'$). Then $\mathcal{B}$ sends $(s', r)$.

This simulation is perfect, therefore we have $\Pr[\mathsf{Guess}_4] = \Pr[\mathsf{Guess}_3]$.

$\mathsf{Game}_5$ Now $\mathcal{B}$ simulates the challenge signature. The euclidean division of $f(y)$ by $(y+h)$ gives $\dfrac{f(y)}{y+h} = \dfrac{\gamma}{x+h} + \displaystyle\sum_{i=0}^{q_\Sigma - 2} \gamma_i y^i$. $\mathcal{B}$ picks at random $b^\star \in \{0,1\}$ and the challenge signature is $(S^\star, r^\star)$ where $S^\star = \dfrac{\gamma}{\alpha^{b^\star}} Q + \displaystyle\sum_{i=0}^{q_\Sigma - 2} \gamma_i \alpha^{b^\star (i-1)} A_i$ and $r^\star = (h\alpha^{b^\star} - m^\star) a_{b^\star}^{-1}$ which is likely to be zero with probability at most $2^{-k}$. If this happens $\mathcal{B}$ aborts the simulation, otherwise he feds $\mathcal{A}$ with $(S^\star, r^\star)$. This game perfectly simulates the signing oracle unless it aborts.

This completes the description of $\mathcal{B}$. If $Q = Q_{\mathsf{real}} = (x+h)^{-1}P$, this game perfectly simulates the challenge generation if the event $\mathsf{Bad}$ does not occur and $\mathcal{B}$ does not abort (which happens with probability at most $2^{-k}$). Therefore $|\Pr[\mathsf{Guess}_5 | Q = Q_{\mathsf{real}}] - \Pr[\mathsf{Guess}_4]| \leq 2^{-k} + \Pr[\mathsf{Bad}]$.

If $Q = Q_{\mathsf{random}}$ is a random element from $\mathbb{G}$, the adversary gains no information on $b$, in an information theoretic sense, therefore:
$\Pr[\mathsf{Guess}_5 | Q = Q_{\mathsf{random}}] \leq 1/2 + 2^{-k} + 2^{-k}$.

The last term accounts for the probability that $Q_{\mathsf{random}} = Q_{\mathsf{real}}$. By definition, the advantage in the $\mathsf{Game}_5$ simulation in solving the $(q_\Sigma + 1)$-DSDH problem is: $\mathbf{Adv}^{(\mathsf{q}_\Sigma+1)-\mathsf{dsdh}}_{\mathsf{Gen},\mathsf{Game}_5}(k) = |\Pr[\mathsf{Guess}_5 | Q = Q_{\mathsf{real}}] - \Pr[\mathsf{Guess}_5 | Q = Q_{\mathsf{random}}]|$.
A simple computation gives the claimed bounds for $\varepsilon'$ and $\tau'$. □

**Proposition 2.** *Let Gen be a POGG and let US be the associated undeniable signature scheme. For any Ano-CMA adversary $\mathcal{A}$ against US, with security parameter $k$, which has advantage $\varepsilon = \mathbf{Adv}^{\mathsf{ano-cma}}_{US,\mathcal{A}}(k)$, running time $\tau$, making $q_\Sigma$ queries to the signing oracle, $q_C$ to the confirming oracle, $q_D$ to the denying oracle and registers up to $q_\mathcal{R}$ keys, there exists an EF-CMA-adversary $\mathcal{C}$ with success $\varepsilon'' = \mathbf{Succ}^{\mathsf{ef-cma}}_{US,\mathcal{A}}(k)$, running time $\tau''$, making $q_\Sigma$ queries to the signing oracle, $q_C$ to the confirming oracle, $q_D$ to the denying oracle and registers up to $q_\mathcal{R}$ keys such that $\varepsilon'' \geq \Pr[Bad_\mathcal{A}]$ and $\tau'' \leq \tau + O(1)$.*

*Proof.* Let $k$ be a security parameter, $(q, \mathbb{G})$ be a couple generated by Gen. We consider random public key $pk$ and we construct a simulation which produces an existential forgery associated to $pk$.

$\mathsf{Game}_0$ Exactly the same game as in the previous proof. By definition, we still have $|2\Pr[\mathsf{Guess}_0] - 1| = \mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{US},\mathcal{A}}(k)$.

$\mathsf{Game}_1$ In this game, the algorithm $\mathcal{C}$ simulates $\mathcal{A}$'s access to the oracles this way. It forwards $pk_0 = pk$ to $\mathcal{A}$ with a new public key $pk_1$ randomly reducible to $pk$ (as in $\mathsf{Game}_?$? of the previous proof). $\mathcal{C}$ simulates $\mathcal{A}$'s signing and confirming/denying protocols by using its own signing and confirming/denying oracles for each of $\mathcal{A}$'s query. During the simulation, $\mathcal{C}$ stored in a $\overline{\Sigma}$-List any pair message/signature accepted by the confirming oracle, not obtained from his signing oracle.
At the end of its Find stage, $\mathcal{A}$ produces a message $m^\star$ and sends it to its challenger. $\mathcal{C}$ simulates this challenger by picking at random a bit $b^\star$ and produces either a real signature of $m^\star$ thanks to its signing oracle. Let $\sigma^\star$ be this signature. $\mathcal{C}$ sends $\sigma^\star$ to $\mathcal{A}$, who begins its Guess stage. The simulation of all oracles is the same as in the Find stage. Finally $\mathcal{A}$ produces a bit $b^\star$.
This game is clearly identical to the previous one. Hence, we obtain $\Pr[\mathsf{Guess}_1] = \Pr[\mathsf{Guess}_0]$.

Finally, $\mathcal{A}$'s output bit is discarded by $\mathcal{C}$, which outputs an element of the $\overline{\Sigma}$-List if it is not empty, and a random element of $\{0,1\}^* \times \mathbb{G} \times [\![1, q-1]\!]$ otherwise. Its running time is the same as $\mathcal{A}$'s, and its success it at least $\Pr[\mathsf{Bad}]$. □

**Theorem 1 (Anonymity of US).** *Let Gen be a POGG and let US be the associated undeniable signature scheme. For any Ano-CMA adversary $\mathcal{A}$ against US, with security parameter $k$, which has advantage $\varepsilon = \mathbf{Adv}^{\mathsf{ano-cma}}_{US,\mathcal{A}}(k)$, running time $\tau$, making $q_\Sigma$ queries to the signing oracle, $q_C$ to the confirming oracle, $q_D$ to the denying oracle and registers up to $q_\mathcal{R}$ keys, there exist*

– *an adversary $\mathcal{B}$ against $(q_\Sigma+1)$-DSDH of advantage $\varepsilon' = \mathbf{Adv}^{(\mathsf{q}_\Sigma+1)-\mathsf{dsdh}}_{Gen,\mathcal{B}}(k)$ and running time $\tau'$;*
– *an EF-CMA-adversary $\mathcal{C}$ with success $\varepsilon'' = \mathbf{Succ}^{\mathsf{ef-cma}}_{US,\mathcal{A}}(k)$, running time $\tau''$, making $q_\Sigma$ queries to the signing oracle, $q_C$ to the confirming oracle, $q_D$ to the denying oracle and registers up to $q_\mathcal{R}$ keys*

*such that $\varepsilon' + \varepsilon'' \geq \varepsilon/2 - (q_\Sigma + 2)2^{-k}$, $\tau' \leq q_C\tau + q_\Sigma(q_\Sigma T_\mathbb{G} + O(1))$ and $\tau'' \leq \tau + O(1)$ where $T_\mathbb{G}$ denotes the time complexity to perform a scalar multiplication in $\mathbb{G}$.*

*Proof.* The result is an obvious consequence of the two previous propositions. □

*Unforgeability.*

**Theorem 2 (Unforgeability of US).** *Let Gen be a POGG and let US be the associated undeniable signature scheme. Let $\mathcal{A}$ be an EF-CMA-adversary against US with success $\varepsilon = \mathbf{Succ}^{\mathsf{ef-cma}}_{US,\mathcal{A}}(k)$ within time $\tau$ making $q_\Sigma$ queries to the signing oracle, $q_C$ to the confirming oracles and $q_D$ to the denying oracle.*

1. *There exists an adversary $\mathcal{B}$ against $(q_\Sigma + 1)$-GSDH of advantage $\varepsilon' = \mathbf{Adv}^{(\mathsf{q}_\Sigma+1)-\mathsf{gsdh}}_{Gen,\mathcal{B}}(k)$ with running time $\tau'$ such that:*
   $\varepsilon' \le \varepsilon/2 - q_\Sigma 2^{-k}$ *and* $\tau' \le \tau + (q_C + q_D)T_{\mathcal{DSDH}} + O(q_\Sigma)$
   *where $T_{\mathcal{DSDH}}$ denotes the time complexity of the oracle $\mathcal{DSDH}$.*
2. *There exists an adversary $\mathcal{C}$ against $(q_\Sigma + 1)$-CSDH of advantage $\varepsilon'' = \mathbf{Adv}^{(\mathsf{q}_\Sigma+1)-\mathsf{csdh}}_{Gen,\mathcal{C}}(k)$ with running time $\tau''$ such that:*
   $\varepsilon'' \le \varepsilon(2(q_C + q_D + 1))^{-1} - q_\Sigma 2^{-k}$ *and* $\tau'' \le \tau + O(q_\Sigma)$.

Proof.(Sketch) We consider an EF-CMA-adversary $\mathcal{A}$ with success $\mathbf{Succ}^{\mathsf{ef-cma}}_{\mathsf{US},\mathcal{A}}(k)$ within time $\tau$. The key generation algorithm is run to produces a pair of keys $(pk, sk)$. The adversary $\mathcal{A}$ is fed with $pk$, and, querying the signing oracle $\Sigma$, and the confirming and denying oracles $\Upsilon_C$ and $\Upsilon_D$, outputs a couple message/signature $(m^\star, \sigma^\star)$, where $\sigma^\star$ was not obtained from the signing oracle. We denote by $q_\Sigma$ the number of queries to the signing oracle and $q_C$ to the confirming oracle and $q_D$ to the denying oracle.

As in Boneh and Boyen proof of security, we will construct an algorithm $\mathcal{B}$ (*resp.* $\mathcal{B}$) which is likely to break the random instance of $(q_\Sigma + 1)$-GSDH (*resp.* the $(q_\Sigma + 1)$-CSDH): $(P, xP, x^2P, \ldots, x^{q_\Sigma+1}P)$.

We distinguish two type of forgers. The simulation of any interaction with the adversary is indistinguishable from the real attack. The only difference comes from the possibility offered to the adversary to query a confirming/denying oracle on any couple message/signature of his choice.

1. Thanks to the decisional oracle associated to $(q_\Sigma + 1)$-SDH and the points $P, xP, x^2P, \ldots, x^{q_\Sigma+1}P$, $\mathcal{B}$ can construct a static oracle $\mathcal{O}_x$, which, given $Q$ and $R$ as inputs, answers whether $R = xQ$. Therefore, $\mathcal{B}$ an perfectly simulate an appropriate proof as in the proof of the theorem 1. The rest of the simulation follows *mutatis mutandis* the one of Boneh and Boyen [1] from which we obtain the claimed bound on $\varepsilon'$ and $\tau'$ once taken into account the computational cost of the simulation of the confirming and denying oracles.
2. We can suppose without loss of generality that the potential forgery output by $\mathcal{A}$ is queried to the confirming oracle at the end of $\mathcal{C}$'s execution. We say that a couple message/signature is *special* if it is a valid message/signature pair queried by $\mathcal{A}$ to the confirming oracle or the denying oracle such that the signature has not been obtained from the signing oracle (in particular $(m^\star, \sigma^\star)$ is special if $\mathcal{A}$ succeeds). $\mathcal{C}$ picks at random an index

$\ell \in [\![1, q_C + q_D + 1]\!]$ as its guess of first query of a special message/signature couple. In $\mathcal{A}$'s execution, we denote by $s$ the actual index of this first query (and $s = \infty$ if $\mathcal{A}$ does not make such a request). For the $i$-th query with $i < \ell$, $\mathcal{C}$ chooses to confirm the signature if it has been made by the signing oracle and to deny it otherwise. This simulation is done as in the previous proof. If the $\ell$-th query $(m_\ell, \sigma_\ell)$, has been obtained from the signing oracle, then $\mathcal{C}$ aborts. Otherwise following *mutatis mutandis* Boneh and Boyen's simulation $\mathcal{C}$ tries to solve the $(q_\Sigma + 1) - CSDH$ problem using the value $\sigma_\ell$. $\mathcal{C}$ does not abort with probability $1/(q_C + q_D + 1)$ and we get the bounds on $\varepsilon''$ and $\tau''$ once taken into account the computational cost of this simulation. □

**Corollary 1 (Security of US).** *Let Gen be a POGG and US be the associated undeniable signature scheme. Under the DSDH assumption in Gen, US is EF-CMA and Ano-CMA secure against polynomial-time adversaries.*

## 4 Conclusion

We designed the first efficient construction for undeniable signatures. It is a variant of Boneh-Boyen's signature scheme in a situation where the DDH problem is supposed to be difficult. The unforgeability and the anonymity are related to variants of the strong Diffie-Hellman assumption. The new scheme offers the advantage of issuing short signatures. Moreover, the computational costs for the signer in the signature generation, the confirmation/denial protocols and the receipt generation algorithms are the lowest of all known schemes.

Zhang and Chen [24] proposed very recently a new digital signature scheme in a bilinear setting whose resistance to forgery is reduced, in the standard security model, to a new algorithmic problem called the *$k$-square roots problem.* This protocol is very close to Boneh and Boyen's scheme, the underlying non-linear operation in $[\![1, q-1]\!]$ being the square root extraction, instead of the inversion. The computational costs of generation and verification and the size of these signatures are identical to those of Boneh-Boyen's signatures. By embedding this scheme in a classical cryptographic setting we can construct, with the same technique, a new efficient undeniable signature scheme which can be proved unforgeable in the standard security model.

## References

1. D. Boneh, X. Boyen: Short Signatures Without Random Oracles. Proc. of Eurocrypt'04, Springer LNCS Vol. 3027, 56–73 (2004)
2. D. Boneh, B. Lynn, H. Shacham: Short Signatures from the Weil Pairing. J. Cryptology 17 (4), 297–319 (2004)
3. J. Boyar, D. Chaum, I. B. Damgård, T.P. Pedersen: Convertible Undeniable Signatures. Proc. of Crypto'90, Springer Vol. LNCS 537, 189–205 (1991)
4. G. Brassard, D. Chaum, C. Crpeau: Minimum Disclosure Proofs of Knowledge. J. Comput. Syst. Sci., 37 (2) 156–189 (1988)

5. D. Chaum: Zero-Knowledge undeniable signatures. Proc. of Eurocrypt'90, Springer LNCS Vol. 473, 458–464 (1991)
6. D. Chaum, H. van Antwerpen: Undeniable Signatures. Proc. of Crypto'89, Springer LNCS Vol. 435, 212–216 (1990)
7. S. Even, O. Goldreich, S. Micali: On-Line/Off-Line Digital Signatures. J. Cryptology, 9 (1), 35–67 (1996)
8. S. Galbraith, W. Mao: Invisibility and anonymity of undeniable and confirmer signatures. Proc. of CT-RSA 2003, Springer LNCS Vol. 2612 80–97 (2003)
9. S. Galbraith, W. Mao, K.G. Paterson: RSA-based undeniable signatures for general moduli. Proc. of CT-RSA 2002, Springer LNCS Vol. 2271, 200–217 (2002)
10. R. Gennaro, T. Rabin, H. Krawczyk: RSA-based undeniable signatures. J. Cryptology 13 (4), 397–416 (2000)
11. S. Goldwasser, S. Micali, R. Rivest: A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. SIAM J. Computing, 17 (2), 281–308 (1988)
12. S. Goldwasser, E. Waisbard: Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes. Proc. of TCC'04, Springer LNCS Vol. 2951, 77–100 (2004)
13. M. Jakobsson, K. Sako, R. Impagliazzo: Designated Verifier Proofs and their Applications. Proc.of Eurocrypt'96, Springer LNCS Vol. 1070, 142–154 (1996)
14. K. Kurosawa, S.-H. Heng: 3-Move Undeniable Signature Scheme. Proc of Eurocrypt'05, Springer LNCS Vol. 3494, 181–197 (2005)
15. F. Laguillaumie, D. Vergnaud: Time-Selective Convertible Undeniable Signatures. Proc. of CT-RSA'05, Springer LNCS Vol. 3376, 154-171 (2005)
16. B. Libert, J.-J. Quisquater: Identity Based Undeniable Signatures. Proc. of CT-RSA 2004, Springer LNCS Vol. 2964, 112–125 (2004)
17. M. Michels, H. Petersen, P. Horster: Breaking and repairing a convertible undeniable signature scheme. Proc. of ACM Conference on Computer and Communications Security 1996, 148–152, ACM Press (1996)
18. J. Monnerat, S. Vaudenay: Generic Homomorphic Undeniable Signatures. Proc. of Asiacrypt'04, Springer LNCS Vol. 3329, 354–371 (2004)
19. J. Monnerat, S. Vaudenay: Undeniable Signatures Based on Characters: How to Sign with One Bit. Proc. of PKC 2004, Springer LNCS Vol. 2947, 69–85 (2004)
20. T. Okamoto, D. Pointcheval: The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. Proc. of PKC 2001, Springer LNCS Vol. 1992, 104–118 (2001)
21. W. Ogata, K. Kurosawa, S.-H. Heng: The Security of the FDH Variant of Chaum's Undeniable Signature Scheme. Proc of PKC 2005, Springer LNCS Vol. 3386, 328–345 (2005)
22. C. P. Schnorr: Efficient Signature Generation by Smart Cards. J. Cryptology 4 (3), 161–174 (1991)
23. F. Zhang, R. Safavi-Naini, W. Susilo: An Efficient Signature Scheme from Bilinear Pairings and its Applications. Proc. of PKC 2004, Springer LNCS Vol. 2947, 277–290 (2004)
24. Z. Zhang, X. Chen: Yet Another Short Signatures Without Random Oracles from Bilinear Pairings. Cryptology ePrint Archive, Report 2005/203 (2005)

# Short Threshold Signature Schemes Without Random Oracles[⋆]

Hong Wang, Yuqing Zhang, and Dengguo Feng

State Key Laboratory of Information Security,
Graduate School of the Chinese Academy of Sciences, Beijing, 100049, PRC
wanghong@is.ac.cn

**Abstract.** Recently, Boneh and Boyen proposed a short digital signature scheme where signatures are as short as DSA signatures, but are provably secure in the absence of random oracles. We propose threshold signature schemes based on their short signature scheme. Signatures of our schemes are the same as the underlying short signature scheme. We also prove security of our schemes under q-SDH assumption without using random oracles. To the best of our knowledge, this is the first threshold construction for the short signature scheme without random oracles.

## 1 Introduction

Threshold cryptography and secret sharing have been given considerable attention. The first threshold secret sharing schemes, based on the Lagrange interpolating polynomial and linear project geometry, were proposed by Shamir [20] and Blakley [5], respectively.

Many efficient digital signature and threshold signature schemes are proved secure in the random oracle model. However, most prominently is the result from [9] that there exists an encryption scheme which is secure in the random-oracle model (RO model), but is not secure in the complexity-theoretic model (CT model), no matter the instantiation of the RO. This leads to focus on constructing secure cryptosystems proved without random oracles.

Most current practical signature schemes secure without random oracles are based on the Strong RSA assumption[11][12]. Recently, Boneh and Boyen [6] proposed a short digital signature scheme where signatures are as short as DSA signatures, but are provably secure in the absence of random oracles. Boneh-Boyen short signature scheme is proved to be existentially unforgeable under a chosen message attack without using random oracles. They prove security of their scheme using a complexity assumption called the Strong Diffie-Hellman assumption, or SDH for short. Roughly speaking, the $q$-SDH assumption in a group $G$ of prime order $p$ states that the following problem is intractable: given $g, g^x, g^{x^2}, \cdots, g^{x^q} \in G$ as inputs, output a pair $(c, g^{1/(x+c)})$ where $c \in Z_p^*$. $q$-SDH may be viewed as a discrete logarithm analogue of the Strong RSA assumption.

⋆ This work was supported by the National Natural Science Foundation of China (60403005), and partially (60373040).

© Springer-Verlag Berlin Heidelberg 2005

The properties of $q$-SDH make it a useful tool for constructing cryptographic systems. Several signature schemes based on $q$-SDH assumption are presented [6][7][10].

In this paper, we present threshold signature schemes based on the short signature scheme without random oracles. To our best knowledge, short threshold signature schemes without random oracles have not been given. Our schemes give several distributed constructions for the short signature scheme without random oracles based on known techniques, such as Pedersen VSS [18], and distributed multiplication [1][2], etc. Though it significantly simplifies signature verification in the underlying short signature scheme, the bilinear map is not easily applied to the share validation in our threshold constructions. The main obstacle is the distributed inversion computation in the exponent. To the date, the best methods to distributedly compute the inverse of a shared secret is taken from [4]. For the distributed inversion in the exponent, the inverse of some random variable must be constructed publicly. Intuitively, the bilinear map is not adapted for simplifying this situation.

The remainder of this paper is organized as follows. In section 2, we give a brief review of Boneh et al.'s scheme. In section 3, we present some build blocks will be used in our proposed schemes. In section 4, we propose a threshold signature scheme based on Boneh-Boyen scheme using error correct technique. In section 5, we propose an optimal resilient short threshold signature scheme not using error correct technique. In section 6, we shall present security proof for our new schemes. Finally, section 7 is our conclusion.

## 2 Brief Review of Boneh-Boyen Short Signature Scheme

We give a brief review of Boneh-Boyen short signature scheme in the standard model using the $q$-SDH assumption.

Let $(G_1, G_2)$ be bilinear groups where $|G_1| = |G_2| = p$ for some prime $p$. For the moment we assume that the messages $m$ to be signed are elements in $Z_p^*$, but it is pointed out that, the domain can be extended to all of $\{0,1\}^*$ using a collision resistant hash function $H : \{0,1\}^* \to Z_p^*$.

**Key generation:** Pick a random generator $g_2 \in G_2$ and set $g_1 = \psi(g_2)$. Pick random $x, y \xleftarrow{R} Z_p^*$, and compute $u \leftarrow g_2^x \in G_2$ and $v \leftarrow g_2^y \in G_2$. Also compute $z \leftarrow e(g_1, g_2) \in G_T$. The public key is $(g_1, g_2, u, v, z)$. The secret key is $(x, y)$.

**Signing:** Given a secret key $x, y \in Z_p^*$ and a message $m \in Z_p^*$, pick a random $r \in Z_p^*$ and compute $\sigma \leftarrow g_1^{1/(x+m+yr)} \in G_1$. Here $1/(x+m+yr)$ is computed modulo $p$. In the unlikely event that $x+m+yr = 0$ we try again with a different random $r$. The signature is $(\sigma, r)$.

**Verification:** Given a public key $(g_1, g_2, u, v, z)$, a message $m \in Z_p^*$, and a signature $(\sigma, r)$, verify that

$$e(\sigma, u \cdot g_2^m \cdot v^r) = z$$

If the equality holds the result is valid; otherwise the result is invalid.

## 3 Building Blocks

Here we briefly recall and present a few techniques that we use in our solutions.

### 3.1 Pedersen's Verifiable Secret Sharing

Our scheme uses an information theoretically secure verifiable secret sharing scheme given by Pedersen [18]. Let $g$ and $h$ be elements of a group $G$ with the order $p$ where $\log_g h$ is unknown. To share a secret $S$ in $Z_p$, the dealer first chooses two $t$-degree random polynomials $f(x)$ and $d(x)$ from $Z_p[x]$ such that $f(0) = S$. Let $R$ denote a random companion secret of $S$, the free term of $d(x)$.

(1) The dealer chooses two random polynomials $f(x)$ and $d(x)$ over $Z_p$, where $f(0) = S$. Then the dealer secretly transmits to each party $P_i$ his shares $S_i = f(i) \bmod p$ and $R_i = d(i) \bmod p$ respectively. The dealer also publishes $E_k = g^{f_k} h^{g_k}$ where $f_k$ and $g_k$, $k = 0, \cdots, t$, are the k-th coefficients of $f(x)$ and $d(x)$ respectively.

(2) Each player $P_i$ verifies his shares as: $g^{S_i} h^{R_i} = \prod_{k=0}^{t} E_k^{i^k}$. If the verification is not passed, then $P_i$ broadcasts a complaint to the dealer. When the complaints are more than $t$, then dealer is disqualified. Otherwise, dealer must reveal the correct shares satisfied with the verification equation. If it is not satisfied, the dealer is disqualified.

Let $\Lambda_G$ be the set of players that are not disqualified. Apparently, the secret $S$ can be reconstructed: $S = \sum_{i \in \Lambda_G} \lambda_{i,\Lambda_G} S_i \bmod p$, where $\lambda_{i,\Lambda_G}$ is the Lagrange interpolation coefficient.

We use the notations referred to [1][2]. The protocol described above could be denoted as $PedVSS(S, R)[g, h] \xrightarrow{f,d} (S_i, R_i)(E_0, \cdots, E_t)$. For simplicity, if no misunderstanding is expected, polynomials put on an arrow may be omitted. The Pedersen VSS protocol is denoted as $PedVSS(S, R)[g, h] \rightarrow (S_i, R_i)(E_0, \cdots, E_t)$.

### 3.2 Distributed Random Secret Sharing

As it is well-known, the players can share a random value unknown to any set of players less than $t + 1$ by using Pedersen's VSS. We refer to this shared random value generation protocol as $RndVSS([S], [R])[g, h] \rightarrow (S_i, R_i)(E_0, \cdots, E_t)$. Brackets $[\cdot]$ imply that the values are unknown to any set of players less than the threshold.

(1) Each player $P_i$ runs $PedVSS(S_0^{(i)}, R_0^{(i)})[g, h] \rightarrow (S_j^{(i)}, R_j^{(i)})(E_0^{(i)}, \cdots, E_t^{(i)})$ as a dealer. Note that $(i)$ is a superscript, not an exponent.

(2) Let $Q_G$ be the set of players that are not disqualified in the above step. Each player $P_i$ computes locally its shares: $S_i = \sum_{j \in Q_G} S_i^{(j)} \bmod p$, $R_i = \sum_{j \in Q_G} R_i^{(j)} \bmod p$ and all players compute the verification information: $E_k = \prod_{j \in Q_G} E_k^{(j)}$, $k = 0, \cdots, t$.

Notice that the shared secret $S$ is defined as $S = \sum_{i \in Q_G} S_0^{(i)} \bmod p$.

Using $RndVSS([S], [R])[g, h] \rightarrow (S_i, R_i)(E_0, \cdots, E_t)$ as subroutines, a joint Feldman-like random secret sharing protocol can be given, with the value $g^S$ is public where $S$ is the random secret in $Z_p$. This protocol is called DKG in [13][14] to highlight its role as distributed key generation in threshold DSS schemes. The protocol achieves an unconditionally secure joint random secret sharing where the secret $S$ is uniformly distributed but also the public value $g^S$. We denote it by $ExpRSS([S],[R])[g,h] \to (S_i,R_i)(E_0,\cdots,E_t;F_0,\cdots,F_t)$ as follows.

(1) The players run $RndVSS([S],[R])[g,h] \to (S_i,R_i)(E_0,\cdots,E_t)$.

(2) Let $Q_G$ be the set of players that are not disqualified in the above step. If $P_i \in Q_G$, $P_i$ broadcasts values $F_k^{(i)} = g^{f_k^{(i)}}$, where $f_k^{(i)}$ is the k-th coefficients of the polynomial for sharing $S_0^{(i)}$, $k = 0,\cdots,t$.

(3) Each player $P_j$ verified the values broadcasted by $P_i \in Q_G$: $g^{S_j^{(i)}} = \prod_{k=0}^{t}(F_k^{(i)})^{j^k}$. If the check fails for player $P_i$, $P_j$ complains against $P_i$ by broadcasting the values $S_j^{(i)}$ and $R_j^{(i)}$, which pass the verification in step (1) but fail in this step.

(4) For player $P_i$ who receive at least one valid complaint, the other players reconstruct in the clear the polynomial for sharing $S_0^{(i)}$ and values $F_k^{(i)} = g^{f_k^{(i)}}$, $k = 0,\cdots,t$.

(5) All players compute the verification information: $F_k = \prod_{j\in Q_G} F_k^{(j)}$, $k = 0,\cdots,t$, and the public value $g^S = F_0$.

### 3.3 Fault Tolerant Interpolation

Interpolation techniques have proved useful tools for various system-theoretic problems. In Shamir's secret sharing scheme, the secret is shared by a random polynomial. The shared secret and polynomial can be reconstructed from $t+1$ correct shares via Lagrange interpolation equation. If the shares gathered by the combiner are not totally correct, we call this type of polynomial reconstruction problem as fault tolerant interpolation.

If $c_1,\cdots,c_n$, $n \geq 3t+1$, are the shares after running a Shamir secret sharing scheme, and we assume that at most $t$ values are modified by malicious adversary, then the shared secret can be recovered by using the Welch-Berlekamp decoding algorithm [21]. We denote this procedure as $c = WBInterpolation(c_1,\cdots,c_n)$.

Let $G$ be a cyclic group of order $p$, and $g$ be its generator. Set $w_i = g^{c_i}$, $i = 1,\cdots,n$, $n \geq 3t+1$, and $W = \{w_1,\cdots,w_n\}$. It is assumed that at most $t$ values are modified by malicious adversary. From $t+1$ values, interpolation "in the exponent" means that we can calculate $\beta = \prod_{j=1}^{t+1} w_{i_j}^{\lambda_{i_j,\Lambda}}$, where $\Lambda = \{i_j, j = 1,\cdots,t+1\}$, a $(t+1)$-subset of the correct $w_i$'s and $\lambda_{i,\Lambda}$'s are the corresponding Lagrange interpolation coefficients. C. Peikert in [19] claims that unique decoding in the exponent, when the number of errors $e < d/2$, is no harder than the CDH problem in the same group. Thus, we can not achieve in general an efficient algorithm of fault tolerant interpolation in the exponent. The most "efficient" procedure of calculating $g^c$ "in the exponent" we could achieve is presented as follows, only adapted to certain cases, such as when $t$ and $n$ are small, or $n$ is much larger than $t$, etc. We denote this procedure as $g^c = ExpInterpolation(g^{c_1},\cdots,g^{c_n})$.

(1) Select $O_1$ be a set of $t+1$ values from $W$. Interpolating over exponent [14] from these $t+1$ sample values, there exists a polynomial $c(x)$ of degree $t$ over exponent such that $g^{c(i)} = w_i$, where $w_i \in O_1$. Construct a set $S_1$ of all the values lie in that polynomial. Clearly, $O_1 \subseteq S_1$, $|S_1| \geq t+1$. If $|S_1| \geq 2t+1$, then the algorithm stops, $g^{c_0}$ can be interpolated over exponent from $O_1$ and outputted. Set $k = 1$.

(2) Set $k = k+1$. If there exists a set $O_k$ of $t+1$ values from $W$ with $O_k \not\subset S_i$, $i = 1, \cdots, k-1$, then select the set. Otherwise, the algorithm stops and outputs "not found".

(3) Construct a set $S_k$ for the values lie in the $t$-degree interpolation polynomial over exponent from the $t+1$ sample values in $O_k$.

(4) If $|S_k| \geq 2t+1$, then the algorithm stops, $g^{c_0}$ can be interpolated over exponent from $O_k$ and outputted. Otherwise, go to step (2).

## 4 Proposed Short Threshold Signature Scheme, $n \geq 4t+1$

In this section, we shall propose the short threshold signature scheme. Our scheme is comprised of three phases: (1) distributed key generation stage, (2) threshold signature generation stage, (3) signature verification stage. The signature verification stage is just as in the original short signature scheme.

Before we present the details of our scheme, we shall first outline the communication and adversary models.

*Communication model.* We assume that the involved $n$ participants are connected by a broadcast channel. Furthermore, any one pair of the participants is connected by a private channel. We also assume that there is a universal clock such that each participant knows the absolute time, and the communication channel is (partially) synchronous by rounds.

*Adversary model.* We consider a static adversary who chooses corrupted participants at the beginning of each time period. For the robustness, it means that the scheme can be successfully finished even if the adversary corrupts $t$ participants at most.

### 4.1 Distributed Key Generation Stage

We are interested in joint verifiable threshold key generation algorithms producing Shamir's secret sharing of a secret without a trusted dealer. DKG protocol of [13][14] is based on the ideas similar to the protocol of Pedersen's [17], has comparable complexity, but provably fixes the weakness of the latter.

Of course, we can use the New-DKG protocols in [13][14] to distributedly generate the shared secret keys and output public keys. However, when $n \geq 4t+1$, we can use error correct technique to simplify the distributed key generation.

(k.1) The players generate secret values $x$,$y$, uniformly distributed in $Z_p$, by running two instances of $RndVSS(\cdot)$ with polynomials of degree $t$:

$$RndVSS([x],[R^{(x)}])[g_2,h_2] \rightarrow (x_i, R_i^{(x)})(E_0^{(x)}, \cdots, E_t^{(x)})$$

$$RndVSS([y],[R^{(y)}])[g_2,h_2] \rightarrow (y_i, R_i^{(y)})(E_0^{(y)},\cdots,E_t^{(y)})$$

(k.2) Each player $P_i$ broadcasts $u_i = g_2^{x_i}$, $v_i = g_2^{y_i}$.

(k.3) Every player $P_i$ computes locally

$$u = g_2^x = ExpInterpolation(u_1,\cdots,u_n)$$

$$v = g_2^y = ExpInterpolation(v_1,\cdots,v_n)$$

The public key is $(g_1, g_2, h_2, u, v, z)$. The secret key is $(x, y)$.

### 4.2 Threshold Signature Generation Stage

Assume a message $m$, let a group of $n$ participants performs the following steps to generate the signature:

(s.1) Generate $r$

(a) The players generate a secret value $r$, uniformly distributed in $Z_p^*$, by running

$$RndVSS([r],[R^{(r)}])[g_2,h_2] \rightarrow (r_i, R_i^{(r)})(E_0^{(r)},\cdots,E_t^{(r)})$$

with a polynomial of degree $t$.

(b) Player $P_i$ broadcasts $r_i$. If $P_i$ doesn't broadcast a value, set $r_i$ to *null*.

(c) Player $P_i$ computes locally

$$r = WBInterpolation(r_1,\cdots,r_n)$$

(s.2) Share a random polynomial with constant term 0

Execute $RndVSS(0,0])[g_2,h_2] \rightarrow (b_i, R_i^{(b)})(E_0^{(b)},\cdots,E_{2t}^{(b)})$ with a polynomial of degree $2t$.

(s.3) Share $(x+m+ry)^{-1} \bmod p$

(a) Generate a random value $a$, uniformly distributed in $Z_p^*$, with a polynomial of degree $t$, using $RndVSS([a],[R^{(a)}])[g_2,h_2] \rightarrow (a_i, R_i^{(a)})(E_0^{(a)},\cdots,E_t^{(a)})$.

(b) Player $P_i$ broadcasts $c_i = a_i(x_i + m + ry_i) + b_i \bmod p$.

(c) Player $P_i$ computes locally

$$c = WBInterpolation(c_1,\cdots,c_n) \bmod p$$

$$\bar{c} = c^{-1} \bmod p, \bar{a}_i = \bar{c}\cdot a_i \bmod p$$

Notice that the reciprocal $\bar{c} = a^{-1}\cdot(x+m+ry)^{-1} \bmod p$. As a result of these rounds, each player $P_i$ obtains his secret information: a share $\bar{a}_i$ of $(x+m+ry)^{-1} \bmod p$.

(s.4) Generate signature $\sigma$

(a) Player $P_i$ broadcasts $\sigma_i = g_1^{\bar{a}_i}$.

(b) Player $P_i$ computes locally

$$\sigma = ExpInterpolation(\sigma_1,\cdots,\sigma_n)$$

Notice the result of these rounds is $\sigma = g_1^{1/(x+m+ry)}$.

The final signature is $(r, \sigma)$.

*Notation.* Because we use the fault tolerant algorithm in the exponent, the scheme is adapt to $t$ and $n$ are relatively small. But this scheme can achieve adaptive security with the additional erasure assumption [8].

However, if $G_1 = G_2$ or $g_1 = g_2$, we can construct a threshold scheme avoiding fault tolerant interpolation in the exponent. The modified version is given below:

(1) For the distributed key generation stage, just run the New-DKG protocols in [13] to generate the secret and public keys.

(2) Modify the step (s.3)(a) in the threshold signature generation stage as follows:

(s.3)(a') Jointly Run $ExpVSS([a], [R^{(a)}])[g_2, h_2] \rightarrow (a_i, R_i^{(a)})(E_0^{(a)}, \cdots, E_t^{(a)}; F_0^{(a)}, \cdots, F_t^{(a)})$ to randomly generate a secret $a$ and a public value $F_0^{(a)} = g_2^a$.

(3) After step (s.3)(b), the signature $\sigma = g_1^{1/(x+m+ry)}$ can be computed locally as

$$\sigma = (g_2^a)^{c^{-1}} = (F_0^{(a)})^{c^{-1} \bmod p}$$

## 5 Optimally Resilient Short Threshold Signature Scheme, $n \geq 2t + 1$

In the above scheme, because the error correct techniques are used, it is not optimally resilient, i.e. the number of corrupted parties is tolerant to be less than $n/4$. In this section, we shall propose an optimally resilient short threshold signature scheme. This scheme is comprised of four phases: (1) distributed commitment-key generation stage, (2) distributed key generation stage, (3) threshold signature generation stage, (4) signature verification stage. The signature verification stage is just as in the underlying short signature scheme.

We set $G_1 = G_2$, $g_1 = g_2$.

### 5.1 Distributed Commitment-Key Generation Stage

To achieve the optimal resilience, when the distributed key generation stage is executed on an input $g_2$ only, a (trapdoor) commitment-key $h_2 \in G_2$ must first be generated for Pedersen's VSS protocols. There are several so-called $h$-generation protocols existed [13][17][3][8]. The commitment-key generation protocols used in [3][8] are applied to adaptive security, and could also be adapted to our models. Note that the commitment-key generation protocol is needed to be performed only once for threshold signature application.

We use the Ped-DKG protocol presented in [17] to generate the first trapdoor commitment-key $h_2 \in G_2$. Although Gennaro et al. claim in [13] that Ped-DKG protocol cannot output a random commitment-key with uniform distribution, they also prove that it is infeasible to compute the discrete log of the output of Ped-DKG. This captures the property which is sufficient for our purpose. We denote the Ped-DKG protocol presented in [17] (a little modification to fit our models) as $HGen(\cdot)$ here.

(h.1) The players perform commitment-key generation protocol $HGen(\cdot)$ to output a (trapdoor) commitment-key $h_2 \in G_2$ for Pedersen's commitment. Note that the corresponding trapdoor $\log_{g_2} h_2$ is unknown to any set of $t$ players.

## 5.2 Distributed Key Generation Stage

It is exactly the distributed key generation protocol DKG for discrete-log based systems of Gennaro et al. [13][14].

(k.1) The players generate secret values $x$,$y$, uniformly distributed in $Z_p$, by running two instances of $ExpVSS(\cdot)$ with polynomials of degree $t$:

$$ExpVSS([x],[R^{(x)}])[g_2,h_2] \rightarrow (x_i, R_i^{(x)})(E_0^{(x)},\cdots,E_t^{(x)};F_0^{(x)},\cdots,F_t^{(x)})$$

$$ExpVSS([y],[R^{(y)}])[g_2,h_2] \rightarrow (y_i, R_i^{(y)})(E_0^{(y)},\cdots,E_t^{(y)};F_0^{(y)},\cdots,F_t^{(y)})$$

Notice $u = F_0^{(x)} = g_2^x$, $v = F_0^{(y)} = g_2^y$ is public.

The public key is $(g_1, g_2, h_2, u, v, z)$. The secret key is $(x, y)$.

## 5.3 Threshold Signature Generation Stage

Let a group of $n$ participants performs the following signature generation steps:

(s.1) Generate $r$

(a) The players generate a secret value $r$, uniformly distributed in $Z_p^*$, by running

$$RndVSS([r],[R^{(r)}])[g_2,h_2] \rightarrow (r_i, R_i^{(r)})(E_0^{(r)},\cdots,E_t^{(r)})$$

with polynomials of degree $t$.

(b) Player $P_i$ broadcasts $r_i$ and its companion $R_i^{(r)}$. Player $P_j$ verifies

$$g_2^{r_i} h_2^{R_i^{(r)}} = \prod_{k=0}^{t} (E_k^{(r)})^{i^k}$$

(c) Player $P_i$ selects any subset $\Lambda^{(r)}$ of $t+1$ players who pass the verification in the above round, and computes locally: $r = \sum_{i\in\Lambda^{(r)}} \lambda_{i,\Lambda^{(r)}} r_i \bmod p$, where $\lambda_{i,\Lambda^{(r)}}$ is the appropriate Lagrange coefficient for the set $\Lambda^{(r)}$.

(s.2) Share $(x+m+ry)^{-1} \bmod p$ and generate signature

(a) Generate a random value $a$, uniformly distributed in $Z_p^*$, with a polynomial of degree $t$, using $ExpVSS([a],[R^{(a)}])[g_2,h_2] \rightarrow (a_i, R_i^{(a)})(E_0^{(a)},\cdots,E_t^{(a)}; F_0^{(a)},\cdots,F_t^{(a)})$.

(b) Each player $P_i$ calculates $b_i = x_i + m + ry_i \bmod p$. Then $P_i$ backs-up $a_i$ and $b_i$ using

$$PedVSS(<a_i>,<R_i^{(a)}>)[g_2,h_2] \rightarrow (a_{i,j}, R_{i,j}^{(a)})(<E_0^{(a_i)}>,E_1^{(a_i)},\cdots,E_t^{(a_i)})$$

$$PedVSS(<b_i>,<R_i^{(b)}>)[g_2,h_2] \rightarrow (b_{i,j}, R_{i,j}^{(b)})(<E_0^{(b_i)}>,E_1^{(b_i)},\cdots,E_t^{(b_i)})$$

$$PedVSS(<a_i>,R_i^{(a_i)})[<E_0^{(b_i)}>,h_2] \rightarrow (<a_{i,j}>,R_{i,j}^{(a_i)})(E_0^{(va_i)},\cdots,E_t^{(va_i)})$$

where $E_0^{(a_i)} = \prod_{k=0}^{t} (E_k^{(a)})^{i^k}$, $R_i^{(b)} = R_i^{(x)} + m + rR_i^{(y)}$, and $E_0^{(b_i)} = g_2^{b_i} h_2^{R_i^{(b)}} = g_2^m h_2^m \cdot \prod_{k=0}^{t} (E_k^{(x)} \cdot (E_k^{(y)})^r)^{i^k}$. Brackets $\langle \cdot \rangle$ mean that the variable can be locally computed by the receivers or senders, or it has been sent before. At least $t+1$ players succeed in back-up sharing. Set $\Lambda^{(ab)}$ of all players who pass the verification in this round.

(c) Each player $P_i$, $i = 1, \cdots, 2t+1$, shares $c_i = a_i b_i \bmod p$ using $(t,n)$ Pedersen's VSS as follows:

$$PedVSS(<c_i>, <R_i^{(c)}>)[g_2, h_2] \rightarrow (c_{i,j}, R_{i,j}^{(c)})(<E_0^{(va_i)}>, E_1^{(c_i)}, \cdots, E_t^{(c_i)})$$

where $R_i^{(c)} = R_i^{(b)} \cdot a_i + R_i^{(a_i)} \bmod p$, and $E_0^{(va_i)} = (E_0^{(b_i)})^{a_i} h_2^{R_i^{(a_i)}} = g_2^{c_i} h_2^{R_i^{(b)} \cdot a_i + R_i^{(a_i)}}$.

If player $P_l$ fails in the verification of these Pedersen's VSS, because his shares $a_l$ and $b_l$ are linear combinations of $\{a_i, b_i, i \in \Lambda^{(ab)}\}$, set $a_l = \sum_{i \in \Lambda^{(ab)}} \delta_{i,\Lambda^{(ab)}} a_i$ and $b_l = \sum_{i \in \Lambda^{(ab)}} \delta_{i,\Lambda^{(ab)}} b_i \bmod p$. Then, $P_j$ broadcasts $a'_j = \sum_{i \in \Lambda^{(ab)}} \delta_{i,\Lambda^{(ab)}} a_{i,j}$, $b'_j = \sum_{i \in \Lambda^{(ab)}} \delta_{i,\Lambda^{(ab)}} b_{i,j} \bmod p$, together with their associated randomness (computed as the same linear combination of the associated randomness generated in the above steps). Bad values are sieved out using the public commitments. Let $\Lambda_1$ be any $t+1$ players passed all former verification. Then, it is easily computed that $a_l = \sum_{j \in \Lambda_1} \lambda_{j,\Lambda_1} a'_j \bmod p$, and $b_l = \sum_{j \in \Lambda_1} \lambda_{j,\Lambda_1} b'_j \bmod p$. Thus, $c_l = a_l b_l \bmod p$ is publicly recovered and can be shared using Pedersen's VSS with constant polynomials.

(d) The current secret value $c = ab \bmod p$ is a linear combination of the values $c_1, \cdots, c_{2t+1}$. Let $\Lambda^{(c)} = \{1, \cdots, 2t+1\}$ and $\lambda_{i,\Lambda^{(c)}}$, $i = 1, \cdots, 2t+1$, be the corresponding Lagrange coefficients. Each player $P_j$ broadcasts the appropriate linear combination $c'_j = \sum_{i \in \Lambda^{(c)}} \lambda_{i,\Lambda^{(c)}} c_{i,j} \bmod p$, together with their associated randomness. Bad values are sieved out using the public commitments. Let $\Lambda$ be any $t+1$ players passed all former verification. Then, it is easily computed that $c = \sum_{j \in \Lambda} \lambda_{j,\Lambda} c'_j \bmod p$.

(e) Each player can computes locally

$$\sigma = (g_2^a)^{c^{-1}} = (F_0^{(a)})^{c^{-1}}$$

Notice the result of these rounds is actually $\sigma = g_1^{1/(x+m+ry)}$. The final signature is $(r, \sigma)$.

## 6 Security Proof

Let $G_1$, $G_2$ be cyclic group of prime order $p$, where possibly $G_1 = G_2$. Let $g_1$ be a generator of $G_1$ and $g_2$ a generator of $G_2$. We formally defined the computation assumptions of Boneh-Boyen short signature scheme and our threshold schemes as follows.

**Definition 1.** ($q$-*SDH Assumption* [6][7]) *For every PPT algorithm $\mathcal{A}$, the following function $Adv_{\mathcal{A}}^{q-SDH}(l)$ is negligible.*

$$Adv_{\mathcal{A}}^{q-SDH}(l) = Pr[(\mathcal{A}(g_1, g_2, g_2^x, \cdots, g_2^{x^q}) = (c, g_1^{1/(x+c)})) \wedge (c \in Z_p)]$$

*where the probability is over the random choice of a generator $g_2$ in $G_2$ (with $g_1 \leftarrow \psi(g_2)$), of $x$ in $Z_p^*$, and of the random bits of $\mathcal{A}$.*

It is easy to see that if the $q$-SDH assumption holds, then the DL assumption holds.

The standard notion of security for a signature scheme is called existential unforgeability under a chosen message attack, was formally defined in [16]. In Boneh-Boyen short signature scheme, a slightly stronger notion of security, called strong existential unforgeability, is considered. This stronger security notion is also required for threshold DSS signature schemes in [14]. From this security notion, Gennaro et al. [14] presented the definitions of unforgeability and robustness for threshold signature schemes.

**Definition 2.** *We say that a $(t, n)$-threshold signature scheme $\mathcal{TS} = ($Dist-Key-Gen, Thresh-Sig$)$ is unforgeable, if no malicious adversary who corrupts at most $t$ players can produce, with nonnegligible probability, the signature on any new (i.e., previously unsigned) message $m$, given the view of the protocol Dist-Key-Gen and of the protocol Thresh-Sig on input messages $m_1, \cdots, m_k$ which the adversary adaptively chose.*

**Definition 3.** *A threshold signature scheme $\mathcal{TS} = ($Dist-Key-Gen, Thresh-Sig$)$ is $(t, n)$-robust if both Dist-Key-Gen and Thresh-Sig can complete successfully even in the presence of an adversary who corrupts maliciously $t$ players.*

In order to prove unforgeability, Gennaro et al. use the concept of simulatable adversary view [15]. The following definition [14] is actually a stronger property than Definition 2.

**Definition 4.** *A threshold signature scheme $\mathcal{TS} = ($Dist-Key-Gen, Thresh-Sig$)$ is simulatable if the following properties hold:*

(1) *The protocol Dist-Key-Gen is simulatable. That is, there exists a simulator SIM1 that, on input the public key $PK$ and the public output generated by an execution of Dist-Key-Gen, can simulate the view of the adversary on that execution.*

(2) *The protocol Thresh-Sig is simulatable. That is, there exists a simulator SIM2 that, on input the public input of Thresh-Sig (in particular the public key $PK$ and the message $m$), $t$ shares $x_{i_1}, \cdots, x_{i_t}$, and the signature $\sigma$ of $m$, can simulate the view of the adversary on an execution of Thresh-Sig that generates $\sigma$ as an output.*

Indeed, one can prove that if the underlying signature scheme $\mathcal{S}$ is unforgeable and $\mathcal{TS}$ is simulatable (satisfies Definition 4) then $\mathcal{TS}$ is unforgeable (satisfies Definition 2).

Now, we give the security analysis of our schemes following the above definitions and the models defined in section 4.

**Theorem 1.** *Under the q-SDH assumption, short threshold signature scheme presented in section 4 is a secure (unforgeable and robust) threshold signature scheme for short signature scheme resistant to $t$ faults against a static malicious adversary, when the number of player is $n \geq 4t + 1$.*

*Proof.* (sketch) The robustness is evident. To prove unforgeable security by reduction to underlying short signature scheme, it is needed to construct a simulator protocol by standard techniques. We assume w.l.o.g. that the adversary corrupted the first $t$ players $P_1, \cdots, P_t$. The simulator protocol is shown below.

SIM1-Short-DKG:

Input: public key $u, v \in G_2$.

(SIM1-k.1) Simulator runs $RndVSS(\cdot)$ protocols on behalf of the honest players. Notice that $(x_1, \cdots, x_t)$ and $(y_1, \cdots, y_t)$ of the corrupted players are known to the simulator.

(SIM1-k.2.0) Simulator computes $u_i = g_2^{x_i}$, $v_i = g_2^{y_i}$ for $i = 1, \cdots, t$. Then, from $u_0 = u, u_1, \cdots, u_t$, and $v_0 = v, v_1, \cdots, v_t$, compute $u_{t+1}, \cdots, u_n$ and $v_{t+1}, \cdots, v_n$ by interpolation "in the exponent".

(SIM1-k.2) Simulator broadcasts $u_{t+1}, \cdots, u_n$ and $v_{t+1}, \cdots, v_n$ for honest players.

(SIM1-k.3) Follow the instructions of the protocol for the honest players.

SIM1-Short-TSG:

Input: public key $u, v \in G_2$, message $m$, signature $(r, \sigma)$, shares $(x_1, \cdots, x_t)$ and $(y_1, \cdots, y_t)$ of the corrupted players.

(SIM1-s.1) (a) Simulator runs $RndVSS([r^*], [R^{(r^*)}])[g_2, h_2] \rightarrow (r_i^*, R_i^{(r^*)})$ $(E_0^{(r^*)}, \cdots, E_t^{(r^*)})$ on behalf of the honest players. Notice that all the values are known to the simulator.

(b.0) For $f^*(0) = r, f^*(1) = r_1^*, \cdots, f^*(t) = r_t^*$, compute the polynomial $f^*(x)$. Set $r_{t+1}^* = f^*(t+1)$, $\cdots$, $r_n^* = f^*(n)$.

(b) Simulator controls the honest player $P_i$ broadcasts $r_i^*$, $i = t+1, \cdots, n$.

(c) Follow the instructions of the protocol for the honest players. Notice that all players can compute locally $r = WBInterpolation(\cdots, r_{t+1}^*, \cdots, r_n^*)$.

(SIM1-s.2) Follow the instructions of the protocol for the honest players.

(SIM1-s.3) (a) Follow the instructions of the protocol for the honest players.

(b.0) Choose a random value $\hat{c}$ uniformly distributed in $[0, p-1]$. For $g^*(0) = \hat{c}$, $g^*(1) = c_1 = a_1(x_1 + m + ry_1) + b_1$, $\cdots$, $g^*(t) = c_t = a_t(x_t + m + ry_t) + b_t$, compute the polynomial $g^*(x)$. Set $c_{t+1}^* = g^*(t+1)$, $\cdots$, $c_n^* = g^*(n)$.

(b) Simulator controls the honest player $P_i$ broadcasts $c_i^*$, $i = t+1, \cdots, n$.

(c) Follow the instructions of the protocol for the honest players. Notice that all players can compute locally $\hat{c} = WBInterpolation(\cdots, c_{t+1}^*, \cdots, c_n^*)$, $\bar{c} = \hat{c}^{-1} \bmod p$.

(SIM1-s.4) (a.0) Compute $\sigma_i = g_1^{\bar{c} \cdot a_i}$, $i = 1, \cdots, t$. From the values $\sigma_0 = \sigma$, and $\sigma_i$, $i = 1, \cdots, t$, simulator generates $\sigma_j^* = \sigma^{\lambda_{j,0}} \cdot \prod_{i=1}^{t} \sigma_i^{\lambda_{j,i}}$ for $j = t+1, \cdots, n$ with known Lagrange interpolation efficients $\lambda_{j,i}$.

(a) Simulator controls the honest player $P_i$ broadcasts $\sigma_i^*$, $i = t+1, \cdots, n$.

(b) Follow the instructions of the protocol for the honest players. Notice that all players can compute locally $\sigma = ExpInterpolation(\cdots, \sigma_{t+1}^*, \cdots, \sigma_n^*)$.

Since the simulator steps do not involve the secret key (that the simulator does not know), it is clear that the view of the adversary in these steps is identically distributed between the real and the simulated execution.

**Theorem 2.** *Under the q-SDH assumption, short threshold signature scheme presented in section 5 is a secure (unforgeable and robust) threshold signature scheme for short signature scheme resistant to t faults against a static malicious adversary, when the number of player is $n \geq 2t + 1$.*

*Proof.* (sketch) Since $\log_{g_2} h_2$ is not known to any player, player $P_i$ could compute $\log_{h_2} E_0^{(b_i)}$ only if $b_i = 0$. However, as $b = x + m + ry$ is assumed to be honestly shared, $b_i = 0$ happens with probability $1/p$. Thus, the chance that player $P_i$ distributes inconsistent shares remain negligible. Then, the robustness can be evidently proved.

For the unforgeability, as Theorem 1, we also assume w.l.o.g. that the adversary corrupted the first $t$ players $P_1, \cdots, P_t$. The simulator protocol is shown below.

SIM2-Short-CKG:

(SIM2-h.1) Simulator runs the simulator protocol for $HGen(\cdot)$ as lemma 2 in [13]. Notice that the trapdoor $\log_{g_2} h_2$ is known to the simulator.

SIM2-Short-DKG:

Input: public key $u, v \in G_2$, $h_2 \in G_2$.

(SIM2-k.1) Simulator runs simulator protocols for

$$ExpVSS([x], [R^{(x)}])[g_2, h_2] \rightarrow (x_i, R_i^{(x)})(E_0^{(x)}, \cdots, E_t^{(x)}; F_0^{(x)}, \cdots, F_t^{(x)})$$

$$ExpVSS([y], [R^{(y)}])[g_2, h_2] \rightarrow (y_i, R_i^{(y)})(E_0^{(y)}, \cdots, E_t^{(y)}; F_0^{(y)}, \cdots, F_t^{(y)})$$

with inputs $F_0^{(x)} = u$ and $F_0^{(y)} = v$. Notice that $(x_1, \cdots, x_t)$ and $(y_1, \cdots, y_t)$ of the corrupted players are known to the simulator.

SIM2-Short-TSG:

Input: public key $u, v \in G_2$, $h_2 \in G_2$, trapdoor $\lambda = \log_{g_2} h_2$, message $m$, signature $(r, \sigma)$, shares $(x_1, \cdots, x_t)$ and $(y_1, \cdots, y_t)$ of the corrupted players.

(SIM2-s.1) (a) Simulator runs $RndVSS([r^*], [R^{(r^*)}])[g_2, h_2] \rightarrow (r_i^*, R_i^{(r^*)})$ $(E_0^{(r^*)}, \cdots, E_t^{(r^*)})$ on behalf of the honest players. Notice that all the values are known to the simulator.

(b.0) For $f^*(0) = r, f^*(1) = r_1^*, \cdots, f^*(t) = r_t^*$, compute the polynomial $f^*(x)$. Set $\hat{r}_{t+1}^* = f^*(t+1)$, $\cdots$, $\hat{r}_n^* = f^*(n)$. Using the trapdoor, compute the $\hat{R}_j^{(r^*)}$ such that $\hat{r}_j^* + \lambda \cdot \hat{R}_j^{(r^*)} \equiv r_j^* + \lambda \cdot R_j^{(r^*)} \bmod p$, $j = t+1, \cdots, n$.

(b) Simulator controls the honest player $P_i$ broadcasts $\hat{r}_i^*$ and $\hat{R}_i^{(r^*)}$, $i = t+1, \cdots, n$.

(c) Follow the instructions of the protocol for the honest players.

(SIM2-s.2) (a.0) Choose a random value $\hat{c}$ uniformly distributed in $[0, p-1]$. Compute $A = \sigma^{\hat{c}}$.

(a) Simulator runs the simulator protocol for

$$ExpVSS([a], [R^{(a)}])[g_2, h_2] \rightarrow (a_i, R_i^{(a)})(E_0^{(a)}, \cdots, E_t^{(a)}; F_0^{(a)}, \cdots, F_t^{(a)})$$

with the input $F_0^{(a)} = A$. Notice that $(a_1, \cdots, a_t)$ of the corrupted players are known to the simulator.

(b) Follow the instructions of the protocol for the honest players. Note that $a_i, R_i^{(a)}$ are generated in (SIM2-s.2)(a), $x_i, R_i^{(x)}; y_i, R_i^{(y)}$ are generated in (SIM2-k.1).

(c) Follow the instructions of the protocol for the honest players.

(d) For $g^*(0) = \hat{c}$, $g^*(1) = c'_1$, $\cdots$, $g^*(t) = c'_t$, compute the polynomial $g^*(x)$. Set $\hat{c'}_{t+1} = g^*(t+1)$, $\cdots$, $\hat{c'}_n = g^*(n)$. Simulator controls the honest player $P_i$ broadcasts $\hat{c'}_i, \hat{R'}_i$, $i = t+1, \cdots, n$, where $\hat{c'}_i + \lambda \cdot \hat{R'}_i \equiv c'_i + \lambda \cdot R'_i \bmod p$.

(e) Follow the instructions of the protocol for the honest players. Notice that

$$(g_2^a)^{\hat{c}^{-1}} = (F_0^{(a)})^{\hat{c}^{-1}} = (\sigma^{\hat{c}})^{\hat{c}^{-1}} = \sigma$$

It is clear that the view of the adversary in these steps is identically distributed between the real and the simulated execution, except (SIM2-h.1). In SIM2-h.1, we cannot generate $h_2$ to be random because a rushing party can affect its distribution. However, the step captures the property which is sufficient for our purpose: a simulator can compute the DL of $h_2$, but the adversary is not able to compute the trapdoor $\log_{g_2} h_2$.

## 7 Conclusion

In this paper, we have proposed threshold signature schemes based on Boneh et al's short signature scheme. Under the $q$-SDH assumption, our schemes is robust and fully secure without using random oracles. We proved security of our schemes in the standard model.

## References

1. M. Abe. Robust distributed multiplication without interaction. In *Advances in Cryptology – CRYPTO'99*, LNCS 1666, Springer-Verlag, pp. 130–147 (1999).
2. M. Abe, R. Cramer and S. Fehr. Non-interactive distributed-verifier proofs and proving relations among commitments. In *Advances in Cryptology – ASIACRYPT 2002*, LNCS 2501, Springer-Verlag, pp. 206–223 (2002).
3. M. Abe and S. Fehr. Adaptively secure Feldman VSS and applications to universally-composable threshold cryptography. In *Advances in Cryptology – CRYPTO 2004*, LNCS 3152, Springer-Verlag, pp. 317–334 (2004).
4. J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction. In *Proc. of the 8th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 201–209 (1989).
5. G.R. Blakley. Safeguarding cryptographic keys. In *Proc. of AFIPS'79*, pp. 313–317 (1979).
6. D. Boneh and X. Boyen. Short signatures without random oracles. In *Advances in Cryptology – EUROCRYPT 2004*, LNCS 3027, Springer-Verlag, pp. 56–73 (2004).
7. D. Boneh, X. Boyen and H. Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO 2004*, LNCS 3152, Springer-Verlag, pp. 41–55 (2004).
8. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Adaptive security for threshold cryptosystems. In *Advances in Cryptology – CRYPTO'99*, LNCS 1666, Springer-Verlag, pp. 98–115 (1999).

9. R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, revisited. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 209–218 (1998).
10. S.G. Choi. Traceable signatures based on bilinear pairings. Available at *http://theory.snu.ac.kr/~ sgchoi/sts.pdf* (2004).
11. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM TISSEC*, Vol. 3, No. 3, pp. 161–185 (2000).
12. R. Gennaro, S. Halevi and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology – EUROCRYPT'99*, LNCS 1592, Springer-Verlag, pp. 123–139 (1999).
13. R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology – EUROCRYPT'99*, LNCS 1592, Springer-Verlag, pp. 295–310 (1999).
14. R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Robust threshold DSS signatures. *Information and Computation*, Vol. 164, No. 1, pp. 54–84 (2001).
15. S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM J. Computing*, Vol. 18, No. 1, pp. 186–208 (1989).
16. S. Goldwasser, S. Micali and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, Vol. 17, No. 2, pp. 281–308 (1988).
17. T. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology – EUROCRYPT'91*, LNCS 547, Springer-Verlag, pp. 522–526 (1991).
18. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO'91*, LNCS 576, Springer-Verlag, pp. 129–140 (1991).
19. C. Peikert. On error correction in the exponent. Available at *http://eprint.iacr.org/2005/105/* (2005).
20. A. Shamir. How to share a secret. *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613 (1979).
21. L.R. Welch and E.R. Berlekamp. Error correction of algebraic block codes. *U. S. Patent 4 633 470* (1983).

# Attacking an Asynchronous Multi-party Contract Signing Protocol

Jose A. Onieva[1], Jianying Zhou[2], and Javier Lopez[1]

[1] Computer Science Department,
University of Malaga, 29071 - Malaga, Spain
{onieva, jlm}@lcc.uma.es
[2] Institute for Infocomm Research,
21 Heng Mui Keng Terrace, Singapore 119613
jyzhou@i2r.a-star.edu.sg

**Abstract.** Contract signing is a fundamental service in doing business. The Internet has facilitated the electronic commerce, and it is necessary to find appropriate mechanisms for contract signing in the digital world. From a designing point of view, digital contract signing is a particular form of electronic fair exchange. Protocols for generic exchange of digital signatures exist. There are also specific protocols for two-party contract signing. Nevertheless, in some applications, a contract may need to be signed by multiple parties. Less research has been done on multi-party contract signing. In this paper, we analyze an optimistic $N$-party contract signing protocol, and point out its security problem, thus demonstrating further work needs to be done on the design and analysis of secure and optimistic multi-party contract signing protocols.

**Keywords:** Secure electronic commerce, multi-party contract signing, security protocol analysis.

## 1 Introduction

The Internet has facilitated the electronic commerce. Many business transactions have been shifted to the Internet. The motivation for such a trend is the efficiency and cost-saving. However, as new risks may arise in the digital world, sufficient security measures should be taken. This will help users to establish the confidence for doing business on the Internet.

*Contract signing* is a fundamental service for business transactions, and has been well practiced in the traditional paper-based business model. Now, it is necessary to find appropriate mechanisms for contract signing in the digital world. Consider several parties on a computer network who wish to exchange some digital items but do not trust each other to behave honestly. *Fair exchange* is a problem of exchanging data in a way that guarantees either all participants obtain what they want, or none does. From a designing point of view, contract

© Springer-Verlag Berlin Heidelberg 2005

signing is a particular form of fair exchange, in which the parties exchange commitments to a contract (typically, a text string spelling out the terms of a deal). That is, a contract is a non-repudiable agreement on a given text such that after a contract signing protocol instance, either *each* signer can prove the agreement to any verifier *or none* of them can. If several signers are involved, then it is a *multi-party contract signing* (MPCS) protocol.

There are some two-party contract signing protocols in the literature. Nevertheless, less research has been done on multi-party contract signing. In this paper, we analyze an optimistic multi-party contract signing protocol, and point out its security problem, thus demonstrating further work needs to be done on the design and analysis of secure and optimistic multi-party contract signing protocols.

The rest of this paper is organized as follows. In Section 2, we review the previous work related to contract signing, outline the properties to be satisfied when designing an optimistic contract signing protocol and give explicit definitions for some terms used along the descriptions of these protocols. In Section 3, we analyze an optimistic $N$-party contract signing protocol presented in [11] and demonstrate that the protocol cannot achieve fairness. We conclude the paper in Section 4.

## 2 Related Work

As contract signing is a particular case of fair exchange, any fair exchange protocol found in the literature in which digital signatures are exchanged can be considered as the related work. In all practical schemes, contract signing involves an additional player, called *Trusted Third Party* (TTP). This party is (at least to some extent) trusted to behave correctly, thus playing the role of a notary in paper-based contract signing and somehow sharing the legal duties the former ones have. In fact, designing and implementing a contract signing protocol using an on-line TTP should not be a complicated task. In this case, if Alice and Bob wish to enter into a contract, they each sign a copy of the contract and send it to the TTP through a secure channel. The TTP will forward the signed contracts only when it has received valid signatures from both Alice and Bob.

Nevertheless, in our continuous search for speeding up our daily life activities, it is desirable not using a TTP in a contract signing protocol. Additionally, if the TTP is not involved, the notary fee could be avoided. Some protocols appear in the literature trying to eliminate the TTP's involvement using *gradual exchange* of signatures [9, 10]. But these solutions are not deterministic, thus may not be accepted by signatories. Our objective is to focus on contract signing protocols that necessarily use a TTP only in those cases in which an exception occurs (i.e., a network communication failure or a dishonest party's misbehavior). Otherwise (all-honest-case), the TTP will not be contacted, and parties will bring the protocol to its end by themselves. In the literature, these protocols are called *optimistic contract signing* protocols [2, 3, 4, 14, 15, 16, 17].

Some properties extracted from the different previous work on optimistic contract signing are summarized as follows.

– *Effectiveness* - if each party behaves correctly, the TTP will not be involved in the protocol.
– *Fairness* - no party will be in an advantageous situation at the end of the protocol.
– *Timeliness* - any party can decide when to finish a protocol run without loosing fairness.
– *Non-repudiation* - no party can deny its action.
– *Verifiability of TTP* - if the TTP misbehaves, all harmed parties will be able to prove it.
– *Transparency of TTP* - if the TTP is contacted to resolve the protocol, the resulting contract will be similar to the one obtained in case the TTP is not involved.
– *Abuse-Freeness* - it is not possible for an attacker (either a legitimate participant or an outsider) to show a third party that the contract final state is under its control.

In [8], Ben-Or *et al.* presented an optimistic contract signing protocol based on a *probabilistic approach.* Such a contract signing protocol is said to be $(\nu, \epsilon)$-fair if for any contract $C$, when signer $A$ follows the protocol properly, if the probability that signer $B$ is privileged to validate the contract with the TTP's help is greater than $\nu$, the conditional probability that "$A$ is not privileged", given that "$B$ is privileged", is at most $\epsilon$.

Previous work in which several signatories are involved in a contract can be found in [1, 6, 11, 12]. Only Asokan *et al.* addressed the MPCS problem in synchronous networks [1]. As Asokan states, this solution clearly improves the efficiency of those asynchronous protocols previously presented with respect to the number of messages; $4(n-1)$ messages in the all-honest-case and $6n-4$ messages in the worst case. This is possible due to a better reliability of the underlaying network as we can see in Definition 1 below.

Some authors considered the abuse-freeness property in [13, 7]. Baum-Waidner proposed new protocols in [6] that improve the solutions presented for asynchronous networks in [7] such that the number of rounds is significantly reduced in the case that the number of dishonest participants $t$ is considerably less than the total number of participants $n$ - the smaller $t$ is, the better results the new protocols achieve.

**Definition 1.** *A "synchronous" contract signing protocol is used in synchronous networks in which there is a limited time for a message to reach its destination (otherwise it has been lost and the appropriate transport layer manages these events) even if an attack occurs. Thus a party can determine that a message has not been sent by other party if it did not arrive within the limited time. Users' clocks are assumed to be synchronized.*

**Definition 2.** *An "asynchronous" contract signing protocol is used in asynchronous networks in which there is no limited time for a message to reach* *its destination. Loss and unsorted arrival of messages are possible and have to be managed by the contract signing protocol itself. Clocks are not assumed to be synchronized among users.*

A number of protocols exist in the literature which use an asynchronous model of network (i.e., messages can be reordered and lost) with deadline parameters. But when a deadline is introduced, and thus, synchronized clocks among users are assumed (at least at the moment the deadline is approaching), these protocols are converted into synchronous protocols.

In the literature, MPCS protocols make use of either a ring or a matrix topology. Throughout these solutions, authors use the terms *round* and *step* without clearly defining them, which often brings on confusion with respect to the metric to be used for its efficiency evaluation. For this reason we explicitly define these terms as follows:

- *Round* is understood as the existing time slot in which messages are distributed in synchronous networks. In asynchronous networks, the entities need to wait a local time before going to the next round (in case the round is not completed).
- *Step* refers to the action of sending or receiving a message. It is the operation performed by a participating entity. Each round means one step (when all the messages from all entities are distributed or broadcasted in the same time slot, usually in matrix topologies) or several steps (when messages from the same round are distributed from one entity to another, usually in ring topologies).

It has to be noted that there is some confusion in the literature with respect to the term 'round'. Some authors explain that when the next message to be sent depends on the previous one, that is a different round. But we claim two different cases can be found (1) message to be sent depends on the previous one because the entity needs to compute/verify it before sending the next one or (2) message to be sent depends on the previous one because there is a distribution order to be respected (as in ring topologies). We consider a round occurs in Case (1).

All of previous solutions to the asynchronous multi-party contract signing problem reach the lower bound on the number of rounds described in Theorem 3 given in [13]:

> *Any complete and optimistic* asynchronous *contract-signing protocol with $n$ participants requires at least $n$ rounds in an optimistic run.*

Describing the theorem, Garay *et al.* stated that for each party $P_i$, when it sends a message that can be used (together with other information) by other entities to obtain a valid contract, as the protocol is fair, it must have received in a previous round, a message from the rest of participants in order to be able to get a valid contract too (probably with the TTP's help), no matter how others behave. By an inductive argument, they showed the number of rounds is at least $n$. This stands for $t = n - 1$ and the lower bound decreases with $t$ number of dishonest parties.

Ferrer's asynchronous protocol presented in [11] with only three rounds is an exception. It claimed some years ago to use a number of rounds independent from the number of participants. As this is supposed a huge improvement with respect to efficiency, we analyze the protocol and demonstrate in the next section, it is flawed.

## 3 Analysis of a MPCS Protocol

To the best of our knowledge, only the asynchronous protocol in [11] can finish multi-party contract signing in 3 rounds. Here we demonstrate that this protocol does not fulfill the property of fairness.

It is not clear whether the informal argument given in the theorem above apply to this protocol, since at the end of the first round (that is, after the first n+1 steps needed for all entities to distribute all messages in a ring topology), every party has received at least the initial commitment from every other party [1]. So we give another argument for the theorem above to make clearer that this protocol is not compliant with the theorem, and demonstrate with an attack, the validity of our argument.

We base our argument on the number of rounds a TTP needs to determine whether a party is misbehaving when requesting resolution (i.e., it requests the TTP to cancel but continues the main protocol): The TTP cannot determine whether a party is misbehaving until $round = round_{current} + 1$, since the TTP needs to wait until the next round to see whether this entity cheated and continued the protocol. That means if $n-1$ dishonest parties exist in the worst case, and each of them requests the *cancel* sub-protocol in a different round, $n$ rounds are the minimum required to satisfy fairness in an asynchronous optimistic contract signing protocol.

### 3.1 Protocol Description

The original notation used in the protocol description is as follows:

- $M$ : message containing the contract to be signed. The contract specifies the order of players in a ring for exchanging signature of principal $i$ on the contract $M$.
- $h_i = S_i(H(M))$ : signature of principal $i$ on contract $M$, where $H(.)$ is a collusion-resistant one-way hash function.
- $ACK_i = S_i(H(h\text{-}ACK))$ : signature of principal $i$ on $h\text{-}ACK$, all the signatures and acknowledgments given by other parties in the ring.
- $ACK2_B = S_B(H(ACK_C))$ : last acknowledgment sent to the last player in the protocol.
- $cancel, cancel_A, cancel_C, finish$: variables used by the TTP to maintain the state of a protocol run.

[1] Note that only one step would have been needed if a matrix topology is used and all distribute their commitments at the same time.

Suppose $A$, $B$, and $C$ are 3 parties going to sign a contract. $A$ is the first principal in the ring architecture, $C$ is the last one, and all the rest ($P_2 \cdots P_{n-1}$ for $n$ parties) behave as $B$. This has been previously agreed upon a setup phase by all entities. The *exchange* sub-protocol is as follows:

1. $A \rightarrow B : M, h_A$
2. $B \rightarrow C : M, h_A, h_B$
3. $C \rightarrow A : h_B, h_C$
4. $A \rightarrow B : h_C, ACK_A$
5. $B \rightarrow C : ACK_A, ACK_B$
6. $C \rightarrow A : ACK_B, ACK_C$
7. $A \rightarrow B : ACK_C$
8. $B \rightarrow C : ACK2_B$

If the protocol run is completed, everybody will hold non-repudiation (NR) evidence. In order to demonstrate to an external party the existence of a contract signed by all parties, following NR evidence has to be provided:

- A holds $h_B, h_C, ACK_B, ACK_C$
- B holds $h_A, h_C, ACK_A, ACK_C$
- C holds $h_A, h_B, ACK_A, ACK_B, ACK2_B$

If there is an exception in the main exchange protocol, then the parties get involved in either *cancel* or *finish* sub-protocols with the TTP. First of all, the TTP's intervention is to verify the correctness of the information given by parties. If this information is incorrect, the TTP will send an error message to that party. Some state variables (*cancel*, $finish$, $cancel_A$ and $cancel_C$) are used, all of which with a value false at the beginning for a particular exchange. $ACK_{TTP}$ is the TTP's signature on $H(M)$; this is equivalent to an acknowledgement from a party that should have sent.

If $A$ says that she has not received the first message sent by $C$, $A$ may initiate the following *cancel* sub-protocol:

1. $A \rightarrow TTP : h(M), h_A$
2. $TTP \rightarrow A :$ IF $finish = true$ THEN $ACK_{TTP}$
   ELSE $S_{TTP}(H(cancelled, h_A))$;
   TTP stores $cancel = true$

If the variable $finish$ is true, it means $B$ or/and $C$ had previously finished the protocol with the TTP (see paragraphs below). The TTP had given the NR token $ACK_{TTP}$ to $B$ or/and $C$, and now it has to give the same NR token to $A$. If $B$ and $C$ had not contacted the TTP previously, the TTP will send a message to $A$ to cancel the transaction, and it will store this information ($cancel$ = true) in order to satisfy future requests from $B$ or $C$.

If $A$ says that she has not received the last message from $C$, $A$ may initiate the following *finish* sub-protocol:

```
1. $A \rightarrow TTP : h(M), h_A, h_B, h_C$
2. $TTP \rightarrow A :$ IF $cancel = true$ THEN $S_{TTP}(H(cancelled, h_A))$;
                   TTP stores $cancel_A = true$
       ELSE $ACK_{TTP}$;
                   TTP stores $finish = true$
```

If the variable *cancel* is true, it means $B$ had previously contacted the TTP (see paragraphs below). The TTP had given a message to $B$ to cancel the transaction, and now it has to send a similar message to $A$. Additionally, the TTP will store the variable $cancel_A$ with value true to satisfy potential petitions from $C$. If $B$ had not contacted the TTP previously, the TTP will send the NR token $ACK_{TTP}$ to $A$. In this case the TTP will assign the value true to the variable $finish$, in order to satisfy future petitions from $B$ or/and $C$.

If $B$ says that he has not received the second message from $A$, $B$ may initiate the following *cancel* sub-protocol:

```
1. $B \rightarrow TTP : h(M), h_A, h_B$
2. $TTP \rightarrow B :$ IF $finish = true$ THEN $ACK_{TTP}$
       ELSE $S_{TTP}(H(cancelled, h_B))$;
                   TTP stores $cancel = true$
```

If the variable $finish$ is true, it means $A$ or/and $C$ had previously finished the protocol with the TTP. The TTP had given the NR token $ACK_{TTP}$ to $A$ or/and $C$, and now it has to give the same NR token to $B$. If $A$ and $C$ had not contacted the TTP previously, the TTP will send a message to $B$ to cancel the transaction, and it will store this information (*cancel* = true) in order to satisfy future petitions from $A$ or $C$.

If $B$ says that he has not received the last message from $A$, $B$ may initiate the following *finish* sub-protocol:

```
1. $B \rightarrow TTP : h(M), h_A, h_B, h_C, ACK_A$
2. $TTP \rightarrow B :$ IF $cancel_C = true$ THEN $S_{TTP}(H(cancelled, h_B))$
       ELSE $ACK_{TTP}$;
                   TTP stores $finish = true$
```

If the variable $cancel_C$ is true, it means $A$ and $C$ (in this order) had previously contacted the TTP (see paragraphs above). The TTP had given a message to $A$ and $C$ to cancel the transaction, and now it has to send a similar message to $B$. If $A$ had not cancelled the exchange, the TTP will send the NR token $ACK_{TTP}$ to $B$. In this case the TTP will assign the value true to the variable $finish$, in order to satisfy future petitions from $A$ or/and $C$.

If $C$ says that he has not received the second message from $B$, $C$ may initiate the following *finish* sub-protocol:

```
1. $C \rightarrow TTP : h(M), h_A, h_B, h_C$
2. $TTP \rightarrow C :$ IF $cancel = true$ THEN $S_{TTP}(H(cancelled, h_C))$;
                   TTP stores $cancel_C = true$
       ELSE $ACK_{TTP}$;
                   TTP stores $finish = true$
```

If the variable *cancel* is true, it means $A$ or/and $B$ had previously contacted the TTP (see paragraphs above). The TTP had given a message to $A$ or/and $B$ to cancel the transaction, and now it has to send a similar message to $C$. Additionally, the TTP will store the variable $cancel_C$ with value true to satisfy potential future petitions from $B$. If $A$ and $B$ had not cancelled the exchange with the TTP previously, the TTP will send the NR token $ACK_{TTP}$ to $C$. In this case the TTP will assign the value true to the variable $finish$, in order to satisfy future petitions from $A$ or/and $B$.

If $C$ has not received the last message from $B$, $C$ may initiate the following second *finish* sub-protocol:

```
1. $C \rightarrow TTP : h(M), h_A, h_B, h_C, ACK_A, ACK_B$
2. $TTP \rightarrow C :$ IF $cancel_A = true$ THEN $S_{TTP}(H(cancelled, h_C))$
                 ELSE $ACK_{TTP}$;
                      TTP stores $finish = true$
```

If the variable $cancel_A$ is true, it means $B$ and $A$ (in this order) had previously contacted the TTP. The TTP had given a message to $A$ and $B$ to cancel the transaction, and now it has to send a similar message to $C$. If $A$ and $B$ had not cancelled the exchange with the TTP previously, the TTP will send the NR token $ACK_{TTP}$ to $C$. In this case the TTP will assign the value true to the variable $finish$, in order to satisfy future petitions from $A$ or/ and $B$.

### 3.2 Security Analysis

Once we have described the protocol, the first step of our analysis is to check whether this protocol fulfills all the defined properties.

– It is an asynchronous protocol, i.e., messages can be lost or reach their destination in an unsorted way.
– It is effective, since no TTP participation is needed if all parties behave correctly.
– It is *not* fair, since some parties can be in an advantageous position at the end of the protocol. Furthermore, this flaw cancels the rest of properties. It does *not* fulfill timeliness, non-repudiation (parties could deny their actions), verifiability of TTP, transparency of TTP (the TTP signs affidavits), and abuse-freeness (it allows an entity to decide the final outcome of the protocol).

Using the TTP's resolution protocol just described, let us have a look at the following scenario. Suppose there are 4 participants $(A, B_1, B_2, C)$, where $B_1$ and $B_2$ behave as $B$ in the resolution protocol as they are the intermediate participants in the ring architecture.

| | |
|---|---|
| 1. $A \to B_1$ | : $M, h_A$ |
| 2. $B_1 \to B_2$ | : $M, h_A, h_{B_1}$ |
| 3. $B_2 \to C$ | : $M, h_A, h_{B_1}, h_{B_2}$ |
| 4. $C \to A$ | : $h_{B_1}, h_{B_2}, h_C$ |
| 5. $A \to B_1$ | : $h_{B_2}, h_C, ACK_A$ |
| | : $B_1$ resolves:: *cancel* $=$ *true*; and continues the protocol |
| 6. $B_1 \to B_2$ | : $h_C, ACK_A, ACK_{B_1}$ |
| 7. $B_2 \to C$ | : $ACK_A, ACK_{B_1}, ACK_{B_2}$ |
| 8. $C \to A$ | : $ACK_{B_1}, ACK_{B_2}, ACK_C$ |
| | : $A$ resolves:: $cancel_A = true$; and continues the protocol |
| 9. $A \to B_1$ | : $ACK_{B_2}, ACK_C$ |
| 10. $B_1 \to B_2$ | : $ACK_C$ |
| | : $B_2$ resolves:: $finish = true$; gets the contract |
| | : and continues the protocol |
| 11. $B_2 \to C$ | : $ACK2_{B_2}$ |
| | : $C$ resolves:: *cancelled* |

In this arbitrary protocol execution, all entities contact the TTP for resolving the protocol even though they continue its execution, getting different results. $B_1$ invokes its cancel sub-protocol in step 5, $A$ invokes its finish sub-protocol in step 8, $B_2$ invokes its finish sub-protocol in step 10, and after finishing the protocol $C$ invokes its second finish sub-protocol.

At the end of the protocol, and with the TTP's help, $B_2$ and $C$ obtain different results. This is due to lack of state variables to maintain the actual status of the protocol. $cancel_A$ and $cancel_C$ are not enough for controlling the status if $n-1$ $(n > 3)$ parties are dishonest when they cancel the protocol. In other words, there is no sub-protocol when $B_i$ says it has not received the last message from $B_j$ $(i > j)$ $\in [P_2 \cdots P_{n-1}]$ and the wrong sub-protocol is used.

Furthermore, an inconsistency can be found in its explanation for 3 parties. In the 3-party version, $B$ and $A$ cancel the protocol (in this order) but continue with its execution. If the protocol stops before the last step, $A$ and $B$ have the signed contract, but $C$ can obtain only a cancel token instead. If priority to the cancel token is assigned by the arbitrator, then the protocol is fair, since $C$ presents its cancel token to the arbitrator in case of dispute and the arbitrator settles that the contract has been cancelled.

But if the last step occurs, then now, all get the signed contract but $A$ and $B$ have cancel tokens. In this case priority to the finished state should be assigned, as the honest party acting properly ($C$) got the contract. Nevertheless, the arbitrator does not know who is the honest party, and cannot assign priority to the tokens. This is a serious contradiction. And this happens with any sequence (e.g., also if $A$ and $C$ cancel in this order while $B$ is honest). As shown above, parties cheat about their protocol state when contacting the TTP. Obviously, the TTP can detect it, but can do nothing to repair the situation.

Moreover, in the original work there is no dispute resolution process defined for the multi-party version, which makes it more difficult to explicitly resolve possible conflicts. We claim that in any design of a contract signing protocol, a well-defined dispute resolution process has to be provided.

## 4 Conclusions

Contract signing is a particular form of fair exchange, in which the parties exchange commitments to a contract. Previous work mainly focused on two-party contract signing. In some applications, however, a contract may need to be signed by multiple parties. In this paper, we analyzed an optimistic $N$-party contract signing protocol, and pointed out its security problem. This clearly demonstrates that if we want a bank deposit to be made with several beneficiaries, further research is needed on the multi-party fair exchange protocols and more concretely on multi-party contract-signing protocols. Part of this research contemplates formal verification and analysis of existing solutions.

This finding encourages us to study and improve existing multi-party contract signing solutions. Definitely further work is needed in synchronous networks, and an improvement of efficiency in asynchronous networks needs to be achieved to bring these applications into reality with the user's confidence.

## Acknowledgements

The work described here is partially funded by the FP6-2002-IST-1 project UBISEC, contract number 506926 and the first author has been funded by the Consejeria de Innovacion, Ciencia y Empresa (Junta de Andalucia) under the III Andalusian Research Plan.

## References

1. N. Asokan, Birgit Baum-Waidner, Matthias Schunter, and Michael Waidner. Optimistic synchronous multi-party contract signing. Technical Report RZ 3089, IBM Zurich Research Lab, 1998.
2. N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for multi-party fair exchange. Technical Report RZ 2892 (# 90840), IBM, Zurich Research Laboratory, 1996.
3. N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *Proceedings of 4th ACM conference on Computer and communications security*, pages 7–17. ACM Press, 1997.
4. N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.
5. Feng Bao, Robert Deng, and Wenbo Mao. Efficient and practical fair exchange protocols with off-line ttp. In *Proceedings of 1998 IEEE Symposium on Security and Privacy*, pages 77–85. IEEE, May 1998.

6. Birgit Baum-Waidner. Optimistic asynchronous multi-party contract signing with reduced number of rounds. In *Proceedings of 28th International Colloquium on Automata, Languages and Programming*, pages 898–911. Springer, 2001.
7. Birgit Baum-Waidner and Michael Waidner. Round-optimal and abuse-free multi-party contract signing. In *Proceedings of 27th International Colloquium on Automata, Languages and Programming*, LNCS 1853, pages 524–535. Springer, 2000.
8. M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, volume 36, pages 40–46, 1990.
9. M. Blum. Three applications of the oblivious transfer, 1981.
10. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, volume 28, pages 637–647, 1985.
11. Josep Lluís Ferrer-Gomila, Magdalena Payeras-Capellà, and Llorenç Huguet-Rotger. Efficient optimistic n-party contract signing protocol. In *Proceedings of 4th International Conference on Information Security*, pages 394–407. Springer, 2001.
12. Josep Lluís Ferrer-Gomila, Magdalena Payeras-Capellà, and Llorenç Huguet-Rotger. Optimality in asynchronous contract signing protocols. In *Proceedings of 1st International Conference on Trust and Privacy in Digital Business*, LNCS 3184. Springer, August 2004.
13. Juan A. Garay and Philip D. MacKenzie. Abuse-free multi-party contract signing. In *Proceedings of 13th International Symposium on Distributed Computing*, pages 151–165. Springer, 1999.
14. N. González-Deleito and O. Markowitch. An optimistic multi-party fair exchange protocol with reduced trust requirements. In *Proceedings of 4th International Conference on Information Security and Cryptology*, LNCS 2288, pages 258–267. Springer, December 2001.
15. O. Markowitch and S. Saeednia. Optimistic fair-exchange with transparent signature recovery. In *Proceedings of Financial Cryptography 2001*. Springer, February 2001.
16. Silvio Micali. Simple and fast optimistic protocols for fair electronic exchange. In *Proceedings of 22nd Annual Symposium on Principles of Distributed Computing*, pages 12–19. ACM Press, 2003.
17. Birgit Pfitzmann, Matthias Schunter, and Michael Waidner. Optimal efficiency of optimistic contract signing. In *Proceedings of 17th Annual ACM Symposium on Principles of Distributed Computing*, pages 113–122. ACM Press, 1998.
18. ITU-T. Information technology - Open Systems Interconnection - Security frameworks for open systems: Non-repudiation framework. October 1996.

# Fairness and Correctness in Case of a Premature Abort

Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich

Institut für Algorithmen und Kognitive Systeme / E.I.S.S.,
Universität Karlsruhe (TH), Am Fasanengarten 5, 76128 Karlsruhe, Germany
{bohli, muellerq, sr}@ira.uka.de

**Abstract.** When using cryptographic protocols for security critical applications premature abort is a serious threat. We define two important properties called quit fairness and quit correctness for protocols to resist attacks by premature abort. The main result of the paper is that quit fairness and quit correctness can be achieved for two-party secure function evaluation whereas for multi-party protocols the two properties of quit fairness and quit correctness are mutually exclusive.

This negative result implies that countermeasures to premature abort, e.g. optimistic protocols, are vital for secure electronic applications.

**Keywords:** Fairness, Cryptography, Security requirements.

## 1 Introduction

When using security protocols for electronic applications premature abort is a serious threat. Imagine a secure multi-party computation for which knowing the result is advantageous at the stock market, e.g. several companies securely compute joint statistics on commercial data. In such a situation corrupted participants can gain an unfair advantage if they can abort the computation while having substantially more knowledge about the result than the honest participants.

In this work we consider two security aspects of a premature abort:

– *Quit fairness*: The corrupted parties should not obtain a substantial amount of information about the result of the computation without having the uncorrupted participants learning an almost equal amount of information about the result. More formally: When neglecting prior knowledge then the probability for the corrupted parties of guessing the correct result is always close to the probability with which each honest party can guess the correct result. This security property will be called *quit fairness*.
– *Quit correctness*: The adversary should not be able to "bias" the partial information to lead an uncorrupted party to wrong conclusions and subsequently to wrong decisions. The corrupted parties could do this by aborting the protocol in a moment when the partial information of the honest participants hint at a specific result. The corresponding security property is called

© Springer-Verlag Berlin Heidelberg 2005

*quit correctness* and ensures that the corrupted parties cannot condition an abort on the partial information known to the honest parties.

We use the term quit fairness from [1] to distinguish this kind of fairness from other notions of fairness, e.g. fair scheduling or fair in the sense of unbiased coin tosses. The main result of the paper is that quit fairness and quit correctness can be achieved for two-party secure function evaluation whereas for multi-party protocols the two properties quit fairness and quit correctness are mutually exclusive. This negative result implies that countermeasures to premature abort, e.g. optimistic protocols, are vital for secure electronic business.

*Related Work.* Quit fairness has mainly been studied in protocols for fair exchange, where two parties try to exchange information, such that neither party gets the other party's information without giving her information to the other party. A survey article about fair exchange can be found in [2]. Luby, Micali and Rackoff [3] describe a protocol for fair exchange of a secret bit by flipping a symmetrically biased coin. The idea of the coin tosses is that from one coin flip the parties receive only a small piece of information about the other party's secret and the symmetry guarantees that both participants obtain the same amount of information. However, it allows a participant to recognize once he is ahead of the expected knowledge and thus cannot achieve quit fairness in our strong sense. Other methods for distributing a bit are discussed in [4]. Cleve [5] introduces an improved version of the protocol of [3] which avoids the mentioned attack and takes quit correctness into account. Other occurrences of (quit) fair two-party protocols are fair non-repudiation (for a survey see e.g. [6]) or fair contract signing [7].

The problem of general fair multi-party computation is considered by Goldwasser and Levin in [1] where they propose protocols for secure function evaluation based on the oblivious transfer primitive, although, they do not consider the attack mentioned above. Garay, MacKenzie and Yung [8] describe a way for quit fair computations in a variant of the model of universal composability [9]. However, they rely on computational assumptions plus the additional assumption that the computational power of the adversary is approximately known. We will consider quit fairness without exact bounds for the computational power of the adversary. Our results remain valid for polynomially bounded adversaries, but for simplicity we consider the adversary to be unbounded. Another approach with timing assumptions, namely "timed commitments", is presented in [10].

In this work, we show that a quit fair and quit correct secure function evaluation withstanding all known attacks is impossible for more than two parties. As in [1] we study secure function evaluation in a multi-party setting, i.e., $n$ parties $P_i, \ldots, P_n$ aim at computing the result $y$ of a boolean function $f(x_1, \ldots, x_n)$ where $x_i$ is the input of party $P_i$. We consider just deterministic computations, but the results are also valid for randomized computations by giving an additional input $\alpha$ corresponding to the random choices of the ideal functionality and evaluation of the function $f(x_1, \ldots, x_n, \alpha)$.

*Outline.* First, in Sect. 2, we summarize the security model that is the basis for the following section. In Sect. 3 we define and analyze quit fairness and quit correctness and prove that quit fairness and quit correctness are mutually exclusive in the multi-party case. Finally, in Sect. 4, we outline implications of our result and conclude.

## 2 Security Model

We formulate the security notions of protocols in a simulation based security model like the universal composability model [9] or the model of [11, 12]. In the simulation based models a protocol is given in a real model and an ideal functionality specifying the protocol task is given in an ideal model. The protocol in the real model is called a secure realisation of an ideal functionality if the two models are indistinguishable for a special machine, called the environment $\mathcal{Z}$ in [9] (respectively honest protocol user $\mathsf{H}$ in [11, 12]).

In the real model a set of connected machines runs the protocol obtaining the protocol inputs from $\mathcal{Z}$ and giving the protocol output to $\mathcal{Z}$. Another special machine $\mathcal{A}$ is given to model the adversary that has the network under her control. In the ideal model an ideal functionality $\mathcal{F}$ as a specification for the function evaluation is doing the computation. $\mathcal{F}$ gets the protocol input $x_i$ for each uncorrupted party from $\mathcal{Z}$ and delivers the output to $\mathcal{Z}$, after it has received the inputs from every party. In this model only a limited adversary, called simulator, is given. The simulator has no access to messages sent between $\mathcal{F}$ and uncorrupted participants; the simulater controls only inputs and outputs of corrupted participants. A protocol is regarded secure, if for any adversary in the real model, there exists an simulator in the ideal model, such that the environment cannot distinguish real and ideal model.

Our impossibility results will be derived in the ideal model and hence apply to all real protocols which implement a real functionality. To derive the impossibility results we rely only on two properties of the security models:

- There is at most one machine active at any time.
- The adversary can always abort the protocol when she is active.

The first property is achieved in both models by a message driven scheduling, i.e. a machine becomes only active when it receives a message. The message scheduling is generally controlled by the adversary, though we assume an exception: Output channels of the ideal functionality $\mathcal{F}$ will be scheduled by the functionality $\mathcal{F}$ itself.

We assume the adversary can abort the protocol at any time during her turn. This assumption is justified not only because of the adversarial scheduling, but also because the adversary is allowed to corrupt a *majority* of participants and instruct them not to follow the protocol any further, while on the other hand a minority cannot be able to complete the protocol. The result of an abort is that no honest party will give any further output. Because we look at fairness in such abort situations, settings like optimistic fair exchange (a trusted third party is available which is used to recover the result if something goes wrong) are not considered.

We denote a single participant by $\mathsf{M}$, $\mathcal{M}$ denotes the set of all protocol participants. Further let $\mathcal{H} \subseteq \mathcal{M}$ be the subset of honest and $\mathcal{C} := \mathcal{M} \setminus \mathcal{H}$ the set of corrupted participants.

## 3 Quit Fairness and Quit Correctness

### 3.1 Quit Fairness

Informally, by quit fairness we demand that an adversary cannot abort the protocol having received more knowledge than the honest parties. Quit fairness is not implied in the ideal model. The standard formulation of a functionality $\mathcal{F}$ in the sense of [9] for secure evaluation of a function $f$ does not guarantee quit fairness: The functionality waits for inputs $x_1, \ldots, x_n$, computes $y := f(x_1, \ldots, x_n)$ and outputs the result $y$ to all participants. According to the properties of the security model the messages are delivered one by one. Thus the adversary can corrupt a party $\mathsf{M}$ and deliver the message $y$ at first from $\mathcal{F}$ to $\mathsf{M}$ and then abort the protocol. In this case only the adversary learns $y$ while the honest parties will not know more about $y$ than they could already deduce from their inputs.

Obviously, even in the ideal model, $\mathcal{F}$ should not output the result "at once", but generate an *output sequence* similar to the protocols for fair exchange [3, 5]. In order to bound the additional knowledge a single corrupted party can have to 'being maximally one packet ahead' it needs to be ensured that no one will get the $(n+1)$-th packet of the output sequence before every party knows the $n$-th packet. This cannot be guaranteed if the functionality $\mathcal{F}$'s output channels are scheduled by the adversary since the adversary could schedule the channels to corrupted parties more frequently than the channels to honest participants. Therefore we assume the ideal functionality $\mathcal{F}$ assures that no messages will be delivered too early. This is modelled by "localized" output channels, i.e., the delivery of messages on these channels is self-scheduled by $\mathcal{F}$ itself ad seriatim.[1]

First we fix some notation. We denote the output sequence sent from $\mathcal{F}$ to $\mathsf{M}$ by $out_{\mathsf{M}}$. For $r \in \mathbb{N}$ the elements of the output sequence that were scheduled up to the $r$-th activation of the adversary will be denoted with $out^r_{\mathsf{M}}$. Note that the output sequence is finite and in absence of an abort all packets will be delivered, i.e., there exists a $R \in \mathbb{N}$, such that for $r \geq R$ $out^r_{\mathsf{M}} = out_{\mathsf{M}}$. For a set $\mathcal{N}$ of participants we also write $out_{\mathcal{N}}$, $out^r_{\mathcal{N}}$ for the sets of according output sequences. Finally, to measure the information in an output sequence, we introduce an estimation function $\delta$. Let $\delta(out^r_{\mathsf{M}})$ be the estimation of the result based only on the ideal functionality's output sequence $out^r_{\mathsf{M}}$:

$$\delta(out^r_{\mathsf{M}}) := \operatorname{argmax}_{\hat{y}} P(\hat{y} \mid out^r_{\mathsf{M}}) \ .$$

In Sect. 3.3 we will use the abbreviation $\delta_i := \delta(out^r_{\mathsf{M}_i})$.

Now we can define the property *quit fairness.* With a quit fair protocol it is impossible for the adversary to abort the protocol such that she learns significantly more about the output than honest parties, i.e., her probability to predict the result

[1] As an alternative solution one could imagine that $\mathcal{F}$ sends the outputs to the different parties one by one and waits for an acknowledgment for a message by the receiving party before sending further messages to any other party. Since this confirmations would be additional protocol inputs, it would be a legal usage for the environment $\mathcal{Z}$ to omit the messages and provoke an abort.

based only on the output is no more than $\frac{1}{k}$ greater than the probability of honest parties for security parameter $k$ and some constant $c$. This leads to the following formal definition:

Let $\mathcal{F}$ be an ideal functionality for secure evaluation of a function $y = f(x_1, \ldots, x_n)$ for inputs $(x_1, \ldots, x_n)$. Let $\mathcal{M}$ be the set of participating parties, $\mathcal{H} \subseteq \mathcal{M}$ be the subset of honest participants and $\mathcal{C} := \mathcal{M} \setminus \mathcal{H}$ the subset of corrupted participants. Let $out_{\mathsf{M}}$, with $P(\delta(out_{\mathsf{M}}) = y) = 1$, denote the output delivered to $\mathsf{M} \in \mathcal{M}$ and $out_{\mathcal{C}}$ the adversary's output.

**Definition 1 (Quit fairness).** *A functionality $\mathcal{F}$ is called* quit fair *if the adversary can not abort in a moment when she has an advantage greater than $\frac{1}{k}$ for the security parameter $k$:*

$$\forall r \in \mathbb{N}\ \forall \mathcal{A}\ \forall \mathsf{M} \in \mathcal{H} :$$

$$P(\delta(out^r_{\mathsf{M}}) = y \mid \mathcal{A} \textit{ aborts in } r) \geq P(\delta(out^r_{\mathcal{C}}) = y) \mid \mathcal{A} \textit{ aborts in } r) - \frac{1}{k},$$

*where the probability is taken over the distribution of all outputs $out_{\mathsf{M}}$ of $\mathcal{F}$.*

*Remark 1.* Note that in [1] the definition of quit fairness is different. The knowledge of a participant is measured in the *increase* of the probability to guess the correct output. The definition of Goldwasser and Levin demands that per round every party gets an information less than $\frac{1}{k}$ for a security parameter $k$ with additional conditions on the mean value and the standard deviation of the probability ratio of wrong and correct guesses for a player. They prove a protocol to be quit fair according to their definition. This protocol delivers one bit of information by sending bits which are results from coin tosses which have a bias towards the correct result.

In this work we do not adopt the definition of [1], but introduce a conceptually simpler definition of quit fairness. However, it can easily be seen that the impossibility results can be carried over to the definition of [1].

## 3.2 Output Distribution

We study now the distribution of the output sequences $out_M$ of a quit fair functionality. Let us assume as a first attempt a two party case where both parties get independent coin tosses biased to the result. As shown in [3], by theory of random walks, the adversary can get an advantage in some runs of such sequences of biased coin tosses. The reason is that the adversary can recognize when it is very likely that she has an advantage. Namely, when she gets a higher ratio of equal bits than expected from the bias, the adversary can abort the protocol in this moment and has with high probability more information than average. Obviously, if every party gets the same bits, the quit fairness property is fulfilled. However, as shown in Sect. 3.3, quit correctness is missing.

A method for the output generation that achieves quit fairness is proposed in [5]. Cleve introduces a protocol for fair exchange of secrets. The protocol uses a gradual disclosure scheme with $T$ rounds. The bit to be learnt is denoted by $y$.

Before the protocol starts $T-1$ values $p_i$ are fixed such that $\frac{1}{2} < p_1 < \cdots < p_T = 1$ and the protocol computes in round $i$ a bit $z_i$ so that $P(\delta(z_1,\ldots,z_i) = y) = P(z_i = y) = p_i$. The values $p_i$ are the steps in which the probability advances that a uncorrupted participant learns the result.

This approach yields a method for the ideal functionality $\mathcal{F}$ to compute for every party $\mathsf{M} \in \mathcal{M}$ output sequences $out_{\mathsf{M}} := \{z_1^{\mathsf{M}},\ldots,z_T^{\mathsf{M}}\}$. We assume that $\mathcal{F}$ computes an independent sequence for each participant $\mathsf{M}$. At first $k$ values $p_i = \frac{i/k+1}{2}$ are fixed. The ideal functionality $\mathcal{F}$ has to assure that no party receives $z_{i+1}$ before all parties have received $z_i$. Then the protocol guarantees that after output round $t$ the confidence level for all $\mathsf{M}$ is $P(\delta(out_{\mathsf{M},t}) = y) = p_t$. An ideal functionality for two-party computation using Cleve's approach for computing the output sequences is indeed quit fair.

**Lemma 1.** *Let $\mathcal{F}$ be an ideal functionality for the computation of $y = f(x_1,x_2)$ with $n = 2$ participants that—step by step—outputs independent sequences for the two participants according to [5]. Then $\mathcal{F}$ is quit fair.*

*Proof.* If the adversary has corrupted one party, she can be only one output packet ahead. Therefore her certainty about the result can be $p_i$ contrary to $p_{i-1}$ for the honest party. But the difference $p_i - p_{i-1}$ goes to zero in the security parameter $k$ which determines the number of rounds. If the adversary has corrupted both parties the protocol is trivially fair, since there are no uncorrupted parties left. □

In spite of this lemma for the two-party case we have to be careful if more than two parties are involved. A quit fair functionality $\mathcal{F}$ that outputs sequences $out_{\mathcal{M}}$ independently generated for each $\mathsf{M}$ can not be quit fair for three and more parties. Obviously the adversary can corrupt two parties and will get two output sequences. Because the output sequences are (conditionally) independent, and hence different with high probability, it will provide additional information. We characterize the output distribution of quit fair functionalities formally in a lemma:

**Lemma 2.** *Let $\mathcal{F}$ be a quit fair ideal functionality for the computation of $y = f(x_1,\ldots,x_n)$ with $n \geq 3$ participants $\mathcal{M} = \{\mathsf{M}_1,\ldots,\mathsf{M}_n\}$. Let $out^r_{\mathsf{M}_i}$ be the output received by $\mathsf{M}_i \in \mathcal{M}$ before the $r$-th activation of the adversary and $p_r := P({\delta_i}^2 = y) \geq 0.5$ be the probability that participant $\mathsf{M}_i$'s output points to the correct result. Then the outputs $out_{\mathsf{M}_i}$ and $out_{\mathsf{M}_j}$ of two different participants $i, j$ are equal with a probability converging to $1$ for $k \to \infty$. More precisely: for all $r \in \mathbb{N}$, $i, j \in \{1,\ldots,n\}$:*

$$P(\delta_i = \delta_j) \geq \frac{p_r - \frac{1}{2} - \frac{1}{k}}{p_r - \frac{1}{2} + \frac{1}{k}} \quad .$$

*Proof.* Let $\mathcal{F}$ be a quit fair ideal functionality for a multi-party computation with parties $\mathcal{M} = \{\mathsf{M}_1,\mathsf{M}_2,\mathsf{M}_3,\ldots,\mathsf{M}_n\}$. We will concentrate on the parties $\mathsf{M}_1,\mathsf{M}_2,\mathsf{M}_3$ and assume without restriction that the adversary has corrupted two of them: $\mathcal{C} = \{\mathsf{M}_1,\mathsf{M}_2\}$. The output sequences accessible for the adversary are denoted by $out_{\mathcal{C}} =$

[2] $\delta_i$ is a short form for $\delta(out^r_{\mathsf{M}_i})$.

$\{out_{\mathsf{M}_1}, out_{\mathsf{M}_2}\}$; the uncorrupted party $\mathsf{M}_3$ gets output $out_{\mathsf{M}_3}$. Now consider the adversary aborts the protocol in activation $r$ if both of her outputs at this time yield the same result $\delta_1 = \delta_2$. Obviously, the adversary's guess for the result in this case will be $\delta(out^r_{\mathsf{M}_\mathcal{C}}) := \delta_1 = \delta_2$. By quit fairness we have for the probability that the adversary guesses the correct result:

$$P(\delta(out^r_\mathcal{C}) = y \mid \delta_1 = \delta_2) \leq P(\delta_3 = y) + \frac{1}{k} \ . \tag{1}$$

Note that:

$$P(\delta_1 = \delta_2) + P(\delta_1 = y) + P(\delta_2 = y) = 1 - 2 \cdot P(\delta_1 = \delta_2 = y) \ . \tag{2}$$

By applying Bayes' rule we get:

$$\begin{aligned} P(\delta(out^r_{\{\mathsf{M}_1,\mathsf{M}_2\}}) = y \mid \delta_1 = \delta_2) &= \frac{P(\delta_1 = \delta_2 = y)}{P(\delta_1 = \delta_2)} \\ &\overset{(2)}{=} \frac{P(\delta_1 = \delta_2) - 1 + P(\delta_1 = y) + P(\delta_2 = y)}{2P(\delta_1 = \delta_2)} \overset{(1)}{\leq} P(\delta_3 = y) + \frac{1}{k} \ , \end{aligned}$$

and by solving for $P(\delta_1 = \delta_2)$:

$$P(\delta_1 = \delta_2) \geq \frac{\frac{1}{2}(P(\delta_1 = y) + P(\delta_2 = y) - 1)}{P(\delta_3 = y) - \frac{1}{2} + \frac{1}{k}} \geq \frac{P(\delta_3 = y) - \frac{1}{2} - \frac{1}{k}}{P(\delta_3 = y) - \frac{1}{2} + \frac{1}{k}} \ ,$$

using $P(\delta_i = y) \geq P(\delta_3 = y) - \frac{1}{k}$ for $i \in \{1, 2\}$. □

This lemma shows that in a setting with more than two parties the outputs have to be almost identical for all parties. Unfortunately, the protocol of [5] turns out not to be quit fair, if the parties get identical outputs with a high probability. This can be easily understood in a two-party scenario. The adversaries strategy, having corrupted one participant, is to abort in some fixed round $r$ if she just has got a new output element that differs from the previous one and that is still unknown to the honest party. Let us assume that this element was the $t$-th element, so she can now correctly guess the output with a probability $p_t$, hence without considering this element she was wrong with a probability of $p_t$ and because this probably wrong output is still the current output of the honest party, this party will guess right only with a probability of $1 - p_t$, significantly less than the expected $p_{t-1}$.

## 3.3 Quit Correctness

So we reconsider the output generated by a random coin biased towards the result. When every party gets independent coins, the protocol cannot be quit fair if the adversary can corrupt more than one participant. But the protocol will be quit fair if all participants get the same bit every round. Although we have quit fairness for more than two parties now a new problem arises if the outputs of the adversary and an honest party $\mathsf{M}$ are identical. The adversary can abort the protocol execution in a moment the party's output $out_{\mathsf{M}}$ is to her favor. Even worse, if the adversary already knows the result $y$, e.g. by her inputs or by available external information, the abort can be in a moment the honest participants are misled. In this case party $\mathsf{M}$ will guess the result wrong. This is not violating the quit fairness property since the adversary's output does not contain more information about $y$[3], but the honest party $\mathsf{M}$ gets less information than expected. We call a protocol that is not susceptible to this attack quit correct.

For a quit correct protocol the average information in the output honest parties have got at an abort in round $r$ should be independent of the adversary's abort strategy, i.e. the adversary cannot condition her abort strategy on the information of the honest parties. We try to capture this in the following definition, using the notation from Definition 1: Let $\mathcal{F}$ be an ideal functionality for secure function evaluation of a function $y = f(x_1, \ldots, x_n)$ for inputs $(x_1, \ldots, x_n)$. Let $\mathcal{M}$ be the set of participants, $\mathcal{H} \subseteq \mathcal{M}$ be the subset of honest participants and $\mathcal{C} := \mathcal{M} \setminus \mathcal{H}$ the subset of corrupted participants. Let $out_{\mathsf{M}}$, with $P(\delta(out_{\mathsf{M}}) = y) = 1$, denote the output delivered to $\mathsf{M} \in \mathcal{M}$ and $out_{\mathcal{C}}$ the adversary's output. We compare the output distribution of honest participants in the case of a premature abort when interacting with different adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$.

**Definition 2 (Quit correctness).** *A functionality $\mathcal{F}$ is called* quit correct *if for all inputs $(x_1, \ldots, x_n)$ and all adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, which have a non-zero probability to abort in the same round (based on different strategies):*

$$\forall r\ \forall \mathsf{M} \in \mathcal{H} : \quad P(\delta(out^r_{\mathsf{M}}) = y \mid \mathcal{A}_1 \textit{ aborts in } r) = P(\delta(out^r_{\mathsf{M}}) = y \mid \mathcal{A}_2 \textit{ aborts in } r) \ .$$

We will now study how quit correctness affects the output distribution of $\mathcal{F}$ and compare it to the results obtained in Sect. 3.2.

**Lemma 3.** *Let $\mathcal{F}$ be a quit correct ideal functionality for secure function evaluation of a function $y = f(x_1, \ldots, x_n)$ for inputs $(x_1, \ldots, x_n)$. Then for every pair of parties $\mathsf{M}_1, \mathsf{M}_2 \in \mathcal{M}$ and for all $r \in \{1, \ldots, k\}$ the output sequences $out^r_{\mathsf{M}_1}$ and $out^r_{\mathsf{M}_2}$ are indistinguishable from conditionally independent given $y$:*

$$P(\delta_1 = y, \delta_2 = y) = P(\delta_1 = y) \cdot P(\delta_2 = y) \ ,$$

*with overwhelming probability.*

*Proof.* Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two adversaries that know the result of the computation in advance. Given $y$, let $out_{\mathsf{M}_1}$ and $out_{\mathsf{M}_2}$ be distinguishable from conditionally independent. Then there exists a $r$ where with a non-negligible probability:

$$P(\delta_2 = y \mid \delta_1 = y) \neq P(\delta_2 = y) \ ,$$

with probabilities taken over the outputs. Let $\mathcal{A}_1$ abort the protocol always in the $r$-th round and $\mathcal{A}_2$ abort in the $r$-th round only if $\delta_1 \neq y$. Then the protocol is not quit correct. □

[3] To carry out this attack one can think of the adversary knowing the result a priori or she has an interest in leading an honest party to a specific decision.

**Theorem 1.** *For an ideal functionality realizing a nontrivial secure function evaluation with $n > 2$ participants the properties of quit fairness and quit correctness are mutually exclusive.*

*Proof.* The result follows immediately by combining Lemma 2 and 3. The outputs have to be almost equal to reach quit fairness, while quit correctness requires a output distribution indistinguishable from conditionally independent. □

## 4 Conclusions

The desirable security properties of quit fairness and quit correctness are individually achievable, but mutually exclusive for protocols involving three or more parties. It is difficult to say which of the two properties is more important, because learning a partial result seems to be useless if it can (with a certain probability) be biased by the adversary.

This dilemma becomes apparent if one bases a decision at the stock market on the result of a secure computation. If the protocol is not quit fair it might give the adversary a competitive edge, whereas if it is not quit correct one cannot actually use the partial results. Applications of secure computations should ensure quit fairness and quit correctness. Hence, future research should continue to develop reasonable assumptions[4] to guarantee this.

## References

1. Goldwasser, S., Levin, L.A.: Fair computation of general functions in presence of immoral majority. In Menezes, A., Vanstone, S.A., eds.: Advances in Cryptology, Proceedings of CRYPTO '90. Volume 537 of Lecture Notes in Computer Science., Springer-Verlag (1991) 77–93
2. Pagnia, H., Vogt, H., Gärtner, F.C.: Fair exchange. The Computer Journal **46** (2003) 55–75
3. Luby, M., Micali, S., Rackoff, C.: How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin. In: 24th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 1983, IEEE Computer Society (1983) 11–21
4. Tedrick, T.: How to exchange half a bit. In Chaum, D., ed.: Advances in Cryptology: Proceedings of Crypto 83, Plenum Press, New York (1984) 147–151
5. Cleve, R.: Controlled gradual disclosure schemes for random bits and their applications. In Brassard, G., ed.: Advances in Cryptology, Proceedings of CRYPTO '89. Volume 435 of Lecture Notes in Computer Science., Springer-Verlag (1990) 573–588
6. Kremer, S., Markowitch, O., Zhou, J.: An intensive survey of fair non-repudiation protocols. Computer Communications **25** (2002) 1606–1621
7. Even, S.: A protocol for signing contracts. In Gersho, A., ed.: Advances in Cryptology: A Report on CRYPTO 81, U.C. Santa Barbara Dept. of Elec. and Computer Eng. (1981) 148–153

[4] E.g. as they are used in optimistic protocols.

8. Garay, J.A., MacKenzie, P., Yang, K.: Efficient and secure multi-party computation with faulty majority and complete fairness. IACR ePrint Archive (2004) Online available at `http://eprint.iacr.org/2004/009/`.
9. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42th Annual Symposium on Foundations of Computer Science, Proceedings of FOCS 2001, IEEE Computer Society (2001) 136–145 Full version online available at
`http://www.eccc.uni-trier.de/eccc-reports/2001/TR01-016/revisn01.ps`.
10. Pinkas, B.: Fair secure two-party computation (extended abstract). In Biham, E., ed.: Advances in Cryptology, Proceedings of EUROCRYPT 2003. Volume 2656 of Lecture Notes in Computer Science., Springer-Verlag (2003) 87–105
11. Pfitzmann, B., Waidner, M.: A model for asynchronous reactive systems and its application to secure message transmission. In: IEEE Symposium on Security and Privacy, Proceedings of SSP 2001, IEEE Computer Society (2001) 184–200 Full version online available at `http://eprint.iacr.org/2000/066.ps`.
12. Backes, M., Pfitzmann, B., Waidner, M.: Secure asynchronous reactive systems. IACR ePrint Archive (2004) Online available at
`http://eprint.iacr.org/2004/082.ps`.

# Short E-Cash

Man Ho Au[1], Sherman S.M. Chow[2,⋆], and Willy Susilo[3]

[1] Department of Computer Science,
The University of Hong Kong, Pokfulam, Hong Kong
mhau@cs.hku.hk

[2] Department of Computer Science,
Courant Institute of Mathematical Sciences,
New York University, NY 10012, USA
schow@cs.nyu.edu

[3] Center for Information Security Research,
School of Information Technology and Computer Science,
University of Wollongong, Wollongong 2522, Australia
wsusilo@uow.edu.au

**Abstract.** We present a bandwidth-efficient off-line anonymous e-cash scheme with traceable coins. Once a user double-spends, his identity can be revealed and all his coins in the system can be traced, *without* resorting to TTP. For a security level comparable with 1024-bit standard RSA signature, the payment transcript size is only 512 bytes. Security of the proposed scheme is proven under the $q$-strong Diffie-Hellman assumption and the decisional linear assumption, in the random oracle model. The transcript size of our scheme can be further reduced to 192 bytes if external Diffie-Hellman assumption is made. Finally, we propose a variant such that there exists a TTP with the power to revoke the identity of a payee and trace all coins from the same user, which may be desirable when a malicious user is identified by some non-cryptographic means.

**Keywords:** E-cash, Coin-traceability, Bilinear Pairing.

## 1 Introduction

To conduct business transaction over the Internet, one of the ways to make payment is to use e-cash. The simplest model of an e-cash scheme involves three types of parties: *banks* $B$, *shops* $S$, and *customers* $C$. An e-cash scheme is a set of protocols which includes *withdrawal* (by $C$ from $B$), *purchase* (by $C$ to $S$) and *deposit* (by $S$ to $B$). In the electronic world, all objects are represented by data; e-cash is by no means an exception. Special design can be incorporated in real cash to prevent counterfeiting, but it is easy to duplicate e-cash. Thus it is necessary to prevent a user from spending the same coin twice (*double-spending*).

Resembling real cash, it is desirable that the shop can accept a payment autonomously, without consult any other parties, possibly the bank. E-cash scheme

⋆ Corresponding author.

© Springer-Verlag Berlin Heidelberg 2005

satisfying this property is described as an *off-line* one. The coins are most probably spent in two different shops when they are double-spent. It is kind of impossible for the shops to check for double-spending on their own. Instead, the bank checks for double-spending when the shops deposit the coins. Either the shops will get the real payment, or the bank will identify the double-spender. On the other hand, honest spenders cannot be slandered to have double spent (*exculpability*), and when the shops deposit the money from the payee, the bank should not be able to trace who the actual spender is (*anonymity).*

Many e-cash systems allow the identification of double-spender have been proposed, but most of them rely on the existence of a trusted third party (TTP) to *revoke* the anonymity (so as to identify the double-spender) when double-spending occurs. The revocation is done probably with the help of a *database* maintained by the bank, where certain tracing information obtained during the withdrawal protocol are stored. This information is usually in an *encrypted* form that is believed to be decryptable by the TTP only.

Even though a secure e-cash system prevents the TTP from slandering an honest spender, the revocation feature gives the TTP an elusive power to revoke the anonymity of honest spender as well. To remove this high level of trust, an anonymous e-cash scheme should support owner-tracing without TTP. Identity of double spender should be revoked while the identity of honest user is always protected. To further punish the double spender, all coins spent (and possibly to be spent) by a cheating user can be linked while the withdrawals and payments of an honest user remains unlinkable. That is, certain information can be put in a blacklist so that the coin from the double-spenders can be recognized when it is spent. Moreover, such coin-tracing can only be (instead of trusted to be) performed after double-spending has occurred.

Recent proposal by Camenisch, Hohenberger and Lysyanskaya [8] supports traceability of owner and coin without a TTP. Moreover, their scheme (hereinafter referred as CHL scheme) has the distinctive feature that a user can withdraw more than one coin in a single withdrawal protocol, and these coins can be spent in an unlinkable manner. Put it in a more formal way, $2^\ell$ coins can be withdrawn with the cost of $O(\ell \cdot k)$ instead of $O(2^\ell \cdot k)$, where $k$ is a security parameter. As a result, a "compact electronic wallet" is made possible.

**Our Contributions.**

- We propose three short e-cash systems with different features:
  1. identification and coin-tracing of double-spender without TTP.
  2. even shorter payment transcript size.
  3. owner-tracing and coin-tracing of honest users with the help of a TTP.
- We reinvestigate the efficiency of the CHL scheme, which includes the bandwidth requirements in payment and deposit protocol, and also the bank's storage requirement. We compare it with our proposal for typical usage.

**Organization.** Next two sections discuss related works and technical preliminaries. We define our security model in Section 4. The constructions of the e-cash systems are presented in Section 5, accompanied by a comparison of our proposal with the CHL scheme. We conclude the paper in Section 6.

## 2 Related Work

To protect the benefit of the banks, e-cash should deter counterfeiting. A secure digital signature, being unforgeable, is a good candidate for implementing e-cash. The idea of blind signature was proposed in [11] to support untraceable payment system. The bank can sign on the information associated with the transaction in a blinded way without knowing the information about an individual's whereabouts and lifestyle. Beside, blind signature ensures *unlinkability*: even the bank is given the message/signature pair at later stage, it is impossible to recollect the corresponding invocation of signing protocol. However, the property that user can ask the bank to blindly sign any message is undesirable. Cut-and-choose methodology was applied in [12] such that the bank can ensure by statistical probability that the user has not presented a malformed message. But it is very inefficient by nature. Alternatively, later research work proposed using variations of blind signature scheme, such as restrictive blind signature [6] and partially blind signature [1], to prove a user has not breached security.

Group signatures, introduced by Chaum and Heyst [13], allow individual members to make signatures on behalf of the group. The identity of the actual signer is kept secret, but there is a TTP that can revoke this anonymity. Group signature also provides "another kind" of *unlinkability*, such that the signature produced by the same signer is unlinkable. These privacy-oriented properties (signer-anonymity and unlinkability) have been utilized in various e-cash proposals. The concept of "member" plays different roles in various e-cash proposal; for examples, the issuing banks [18], the payees who spend the coins [18, 19, 23, 25], and the coins themselves (referred as "group of coins" model) [10, 20].

The unlinkability of these signatures could be used maliciously, like money laundering and obtaining a ransom safely [27]. *Fair* e-cash system, suggested independently by [7] and [24], can detect the misuse by criminal when necessary. In fair blind signature [24] and group signature, a TTP can revoke the unlinkability and anonymity respectively. The existence of TTP is especially useful in designing fair e-cash systems. Examples include [25, 19, 23, 10].

For detection of double-spending, the idea of cut-and-choose can also help. However, many similar components are involved in the cash, which make the scheme inefficient. More efficient mechanism involves a single-term only, an example is the secret sharing line method in [14, 15]. The technique to realize this "single-term" property may vary in different schemes [6, 14, 15, 21].

In the "group of coins" model, double-spending detection mechanism can be achieved by compromising the unlinkability of signer-anonymous signatures. Some schemes exploited this idea implicitly. For example, the scheme in [10] incorporated a "linkability tag" to the underlying group signature scheme [2] to ensure the linkage of double-spent coins. As noted in [26], accusatory linking that outputs the identity of the double-spender is needed for offline e-cash system, or the cheater has already benefited by exchanging the double-spent coins with the goods or services before the coins are voided by the bank.

In addition to double-spending detection, it is beneficial to have the *coin-traceability*, such that all the coins withdrawn by a particular payee can be traced. Early fair e-cash systems either do not support coin tracing (e.g. [19] and [25]), rely on the online participation of a TTP (e.g. [7]), or rely on the offline presence of a TTP (e.g. [10] and [24]). Usually the TTP is overpowered. For examples, the TTP in [17] can trace the coins spent by any honest user, and the TTP in the linkable group signature extension of [26] can reveal the identity of any honest user. A new idea of coin-tracing is to do , the coin-tracing without a TTP: any party can trace the coins of the same payee once this payee double-spent [8]. The mechanism in [8] is efficient in the sense that one-by-one checking on spent coins is not necessary, in contract with the traceable signatures in [17].

Note that coin-traceability is different from double-spent coins detection. The later only applies on the coins spent by a double-spender, but the former notion has said nothing about it. For examples, the scheme in [22] and the e-cash system from the transaction escrow scheme in [16] support coin-tracing of *any* user.

## 3 Preliminaries

We review concepts related to bilinear pairings $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

- $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic multiplicative groups of prime order $p$.
- $g_1$, $g_2$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.
- $\psi$ is a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ and $\psi(g_2) = g_1$.
- $\forall x \in \mathbb{G}_1$, $y \in \mathbb{G}_2$ and $a$, $b \in \mathbb{Z}_p$, $\hat{e}(x^a, y^b) = \hat{e}(x, y)^{ab}$.
- $\hat{e}(g_1, g_2) \neq 1$.

$\mathbb{G}_1$ and $\mathbb{G}_2$ can be the same or different groups. We say that two groups ($\mathbb{G}_1$, $\mathbb{G}_2$) are a bilinear group pair if the group action in $\mathbb{G}_1$, $\mathbb{G}_2$, the isomorphism $\psi$ and the bilinear mapping $\hat{e}$ are all efficiently computable.

**Definition 1 (Decisional Diffie-Hellman).** *The Decisional Diffie-Hellman (DDH) problem in $\mathbb{G}$ is defined as follows: Given a quadruple $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, decides whether $c = ab$. We say that the $(t, \epsilon)$-DDH assumption holds in $\mathbb{G}$ if no $t$-time algorithm has advantage at least $\epsilon$ in solving the DDH problem in $\mathbb{G}$.*

**Definition 2 (Decisional Linear Diffie-Hellman).** *The Decisional Linear Diffie-Hellman (DLDH) problem in $\mathbb{G}_1$ is defined as follows: Given a sextuple in the form of $(g_1, g_2, g_3, {g_1}^a, {g_2}^b, {g_3}^c) \in {\mathbb{G}_1}^6$, decides whether $c = a+b$. We say that the $(t, \epsilon)$-DLDH assumption holds in $\mathbb{G}_1$ if no $t$-time algorithm has advantage at least $\epsilon$ in solving the DLDH problem in $\mathbb{G}_1$.*

DLDH problem is proposed and proven secure in the generic group model in [4].

**Definition 3 ($q$-Strong Diffie-Hellman).** *The $q$-Strong Diffie-Hellman ($q$-SDH) problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows: Given a $(q + 2)$-tuple $(g_1, g_2,$*

*$g_2^x, g_2^{x^2}, \cdots, g_2^{x^q}) \in \mathbb{G}_1 \times \mathbb{G}_2^{q+1}$, output a pair $(A, c)$ such that $A^{(x+c)} = g_1$ where $c \in \mathbb{Z}_p^*$. We say that the $(q, t, \epsilon)$-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no t-time algorithm has advantage at least $\epsilon$ in solving the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.*

Again, $q$-SDH problem is proven secure in the generic group model [3].

**Definition 4 (eXternal Diffie-Hellman).** *The eXternal Diffie-Hellman (XDH) problem in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ is defined as solving the DDH problem in $\mathbb{G}_1$ given the follwing three efficient oracles*

1. *solving DDH problem in $\mathbb{G}_2$,*
2. *computing the isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$,*
3. *and computing the bilinear mapping of groups $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$.*

*We say that the $(t, \epsilon)$-XDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ if no t-time algorithm has advantage at least $\epsilon$ in solving the XDH problem in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$.*

The above assumption implies that the isomorphism is computationally one-way, i.e. there does not efficient way to complete $\psi^{-1} : \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The discussion on the choice of elliptic curves which can make the above assumption hold can be found in [4]. In short, the bilinear groups $(\mathbb{G}_1, \mathbb{G}_2)$ should be instantiated using the Weil or Tate pairing over MNT curves; but not supersingular curves.

## 4 Security Model of E-Cash System

### 4.1 Framework

An anonymous e-cash system consists of three parties: the bank, the user and the merchant, together with the following six algorithms.

– `Setup`. On input an unary string $1^\lambda$, where $\lambda$ is a security parameter, the algorithm outputs a master secret key $s$ and a list of publicly known system's parameter param. In an anonymous e-cash, the master secret key is owned by the bank which allows it to issue electronic coins.
– `User Setup`. On input of param, randomly outputs a key pair ($pk$, $sk$).
– `Withdrawal`. The user with input ($pk$, $sk$) withdraws a electronic coin from the bank. The bank responses with input $s$. After executing the protocol, the user obtains the coin $c$ while the bank retains certain information $\tau_w$ which allows it to trace the user should this user double-spends some coin. The bank maintains a database for this trace information.
– `Payment`. The user with input $c$ spends. The merchant response with input param. After the protocol the merchant obtains a transcript including a proof of validity $\pi$ of the coin $c$, and possibly some auxiliary information $aux$, and outputs 0/1, depending whether the payment is accepted.
– `Deposit`. The merchant submits $(\pi, aux)$ to the bank for deposit. The bank outputs 0/1, indicating whether the deposit is accepted. It is required whenever a honest merchant obtains $(\pi, aux)$ by running the `Payment` protocol with some user, there is a guarantee that this coin will be accepted by the bank. The bank adds $(\pi, aux)$ to the database of spent coins.

- `Owner tracing (of double-spender)`. Whenever a user double spent, this algorithm allows the bank to identify the double spender. Formally, on input two payment protocol transcripts from the same coin $c$, the algorithm outputs the public key $pk$ of the owner of coin $c$.
- `Coin tracing (of double-spender)`. Whenever a user double spent, this algorithm allows the bank to publish some tracing information so that all spending of the same user are identified. Formally, on input two payment transcripts from the same coin $c$ of the same owner $pk$, outputs a set of information $\{tag\}$ so that anyone with $\{tag\}$ can identify all coins from user (with public key $pk$) during the payment protocol.

We stress that the difference between fair e-cash and anonymous e-cash is that, in the former case, there exists a TTP which can revoke the anonymity of the coin and hence the privacy of the user. Whether this is desirable or not depends the application as the unconditional anonymity can be misused for illegal purposes such as money laundering or perfect blackmailing.

## 4.2 Security Definition

Security properties are described informally at first.

- *Correctness.* If an honest user runs `Withdrawal` with an honest bank and runs `Payment` with an honest merchant, the merchant accepts the coin. The merchant later runs `Deposit` with the bank, which will accept the coin.
- *Balance.* It means that no collusion of users and merchants can ever spend more coins than they withdrew. This is the most important property from the bank's point of view. We require that the adversary, after running $q_u$ `Withdrawal` protocol with the bank, cannot run the `Deposit` protocol successfully with the bank for $q_u + 1$ times. A deposit query is successful if either (1) the bank accepts the coin or (2) the bank identifies the coin is being double-spent but is unable to identify the double spender[1].
- *Identification of double-spenders.* It is required that suppose a user double spent, he must be identified.
- *Tracing of double-spenders* It is required that if a user double spent, all of his other coins can be traced regardless of it is spent honestly or not.
- *Anonymity of users* Even when the bank cooperates with any coalition of users and merchants, cannot learn anything about an honest user's spending.
- *Exculpability* An honest user cannot be accused of having double spent.

We focus on *Balance* and *Anonymity*, the two most important requirements of e-cash system. The capabilities of an adversary $\mathcal{A}$ is modeled as oracles that answers the following queries from the adversary.

[1] It is assumed that the bank holds the responsibility to charge the double-spender, so the merchant is credited even if the coin has been identified to have been double-spent. An honest merchant may not be able to detect double-spending in an off-line anonymous e-cash system. Thus, condition (2) is included in the definition of balance.

– Withdrawal queries: $\mathcal{A}$ engages in the withdrawal protocol as user and obtains a valid coin.
– Payment queries: $\mathcal{A}$ engages in the deposit protocol as a merchant.
– Hash queries: $\mathcal{A}$ can ask for the values of the hash functions for any input.

We require that the answers from the oracles are indistinguishable from the view as perceived by an adversary in real world attack.

**Balance.** The following game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ formally defines the *Balance* property.

**Definition 5 (Game Balance).**

– (Initialization Phase.) *The challenger $\mathcal{C}$ takes a large security parameter $\lambda$ and runs the* `Setup` *to generate a list of system's parameters* `param` *and also a master secret key $s$. $\mathcal{C}$ keeps $s$ to itself and sends* `param` *to $\mathcal{A}$.*
– (Probing Phase.) *The adversary $\mathcal{A}$ can perform a polynomially bounded number of queries to the oracles in an adaptive manner.*

*$\mathcal{A}$ wins the above game if the number of successful withdrawal queries plus payment queries is less than that of successful deposit queries. A deposit query is successful if either the bank accepts the deposit request or the bank identifies double-spent but is unable to identify the double spender. The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins.*

**Definition 6 (Balance).** *An e-cash game is said to have the Balance property if no adversary has a non-negligible advantage in the game Balance.*

**Anonymity.** The following game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ formally defines the *anonymity of e-cash system.*

**Definition 7 (Game Anonymity).**

– (Initialization Phase.) *The challenger $\mathcal{C}$ takes a sufficiently large security parameter $\lambda$ and runs the* `Setup` *to generate a list of system's parameters* `param` *and also the bank's secret key $s$. $\mathcal{C}$ gives $s$ and* `param` *to $\mathcal{A}$.*
– (Challenge Phase.) *The adversary $\mathcal{A}$ runs the* `withdrawal` *protocol with $\mathcal{C}$. Then $\mathcal{C}$ runs* `deposit` *protocol with $\mathcal{A}$ acting as the bank.*
– (End Game Phase.) *The adversary $\mathcal{A}$ decides if the underlying coin of the two runs are the same.*

*$\mathcal{A}$ wins the above game if it guesses correctly. The advantage of $\mathcal{A}$ is defined as the probability that $\mathcal{A}$ wins minus $\frac{1}{2}$.*

**Definition 8 (Anonymity).** *A e-cash scheme is anonymous if no adversary has a non-negligible advantage in the game Anonymity.*

## 5 Our Proposed E-Cash Systems

**Global parameters for both systems.** Let $\lambda$ be the security parameter. $(\mathbb{G}_1, \mathbb{G}_2)$ is a bilinear group pair with computable isomorphism $\psi$ as discussed. $|\mathbb{G}_1| = |\mathbb{G}_2| = p$ for some prime $p$ of $\lambda$ bits. $H : \{0,1\}^* \to \mathbb{Z}_p$ is a cryptographic hash function. We assume there exists a group $\mathbb{G}_p$ of order $p$ where DDH is hard.

### 5.1 Short E-Cash

We present a short e-cash system that supports identification and coin tracing of double-spender without the need of a TTP. We require the user to verifiably encrypt the tracing information under his own public key during the withdrawal protocol, assuming PKI is deployed. By using technique in [6], secret key of the double-spender can be extracted, and thus tracing information can be decrypted.

– `Bank Setup`. The bank's public key is $bpk = (g_1, g_2, w, h_1, h_2, h_3, u, v, h, h_t)$ and the private key $bsk = \gamma$, generated as follows.
  1. Randomly generates generator $g_2 \in \mathbb{G}_2$ and sets $g_1 = \psi(g_2)$.
  2. Randomly selects $\gamma \in_R \mathbb{Z}_p^*$ and sets $w = {g_2}^{\gamma}$.
  3. Randomly selects generators $h_1$, $h_2, h_3, u, v \in_R \mathbb{G}_1$.
  4. Randomly selects generators $h, h_t$ of $\mathbb{G}_p$.
– `User Setup`. Each user is equipped a discrete logarithm type of public and private key pair $(h^s, s) \in \mathbb{G}_p \times \mathbb{Z}_p^*$.
– `Withdrawal Protocol`. When a user with public key $y = h^s \in \mathbb{G}_p$ wants to withdraw money from the bank, the following protocol is executed.
  1. User selects $\bar{a}, \bar{b}$ such that $\bar{a}\bar{b} = s$, computes $\bar{C} = h_1^{\bar{a}} h_2^{\bar{b}} \in \mathbb{G}_1$, and a signature based on proofs of knowledge (SPK) $\Pi_1$ that $\bar{C}$ is correctly formed. User sends $(\bar{C}, \Pi_1)$ to the bank.
  2. The bank verifies that $\Pi_1$ is valid, randomly generates $r$ and sends it back to the user.
  3. User then computes $a = \bar{a}r$, $b = \bar{b}r^{-1}$, $C = {h_1}^a {h_2}^b$, and computes the encryption $R$ of ${h_t}^a$ and ${h_t}^b$ under its public key $h^s$ for coin tracing. User sends to the bank $C$, $R$ and SPK $\Pi_2$ that they are correctly formed.
  4. The bank verifies that $\Pi_2$ is valid, randomly selects $x \in_R \mathbb{Z}_p^*$ and computes $A = (g_1 C)^{\frac{1}{\gamma + x}} \in \mathbb{G}_1$. The bank sends $(A, x)$ back to the user.
  5. The bank keeps $(A, x, C, \Pi_2)$ in record and debits the user accordingly.
  6. User checks if the coin $(A, x, a, b)$ satisfies $\hat{e}(A, w g_2^x) = \hat{e}(g_1 h_1^a h_2^b, g_2)$.

  The encryption and the proof $\Pi_1$ and $\Pi_2$ are shown in the appendix.
– `Payment Protocol`. Suppose the user spends the coin $(A, x, a, b)$ to a merchant with the identity $I \in \{0,1\}^*$, the following protocol is executed.
  1. User randomly generates $\alpha, \beta \in_R \mathbb{Z}_p^*$, computes the auxiliary commitment $A_1 = u^{\alpha}$, $A_2 = v^{\beta}$, $A_3 = A{h_3}^{\alpha+\beta}$, and tracing information $B_1 = {h_t}^a$ and $B_2 = {h_t}^b$. $\{A_1, A_2, A_3\} \in \mathbb{G}_1$ and $\{B_1, B_2\} \in \mathbb{G}_p$.
  2. User computes two helper values $\delta_\alpha = x\alpha$ and $\delta_\beta = x\beta$.

3. User undertakes a proof of knowledge of values $(\alpha, \beta, x, a, b, \delta_\alpha, \delta_\beta)$ satisfying the relations: $A_1 = u^\alpha$, $A_2 = v^\beta$, $A_1^x = u^{\delta_\alpha}$, $A_2^x = v^{\delta_\beta}$, $B_1 = {h_t}^a$, $B_2 = {h_t}^b$, $\hat{e}(A_3, g_2)^x \hat{e}(h_3, g_2)^{-(\delta_\alpha+\delta_\beta)} \hat{e}(h_3, w)^{-(\alpha+\beta)} \hat{e}(h_1, g_2)^{-a} \hat{e}(h_2, g_2)^{-b} = \frac{\hat{e}(g_1,g_2)}{\hat{e}(A_3,w)}$. This proof of knowledge proceeds as follow.
   - (*Auxiliary Commitment.*) User computes $A_1, A_2, A_3, B_1, B_2$ as above.
   - (*Commitment.*) User randomly selects $r_\alpha$, $r_\beta$, $r_x$, $r_a$, $r_b$, $r_{\delta_\alpha}$, $r_{\delta_\beta} \in_R \mathbb{Z}_p^*$, computes $T_1 = u^{r_\alpha}$, $T_2 = v^{r_\beta}$, $T_3 = A_1^{r_x} u^{-r_{\delta_\alpha}}$, $T_4 = A_2^{r_x} v^{-r_{\delta_\beta}}$, $T_5 = \hat{e}(A_3, g_2)^{r_x} \hat{e}(h_3, g_2)^{-r_{\delta_\alpha}-r_{\delta_\beta}} \hat{e}(h_3, w)^{-r_\alpha - r_\beta} \hat{e}(h_1, g_2)^{-r_a} \hat{e}(h_2, g_2)^{-r_b}$, $T_6 = h_t^{(r_a)}$ and $T_7 = h_t^{(r_b)}$. $T_1, T_2, T_3, T_4$ are in $\mathbb{G}_1$, $T_5$ is in $\mathbb{G}_T$ and $T_6$, $T_7$ are in $\mathbb{G}_p$.
   - (*Challenge.*) Merchant sends the transaction information $M$ to user. User computes $c = H(A_1, A_2, A_3, B_1, B_2, T_1, T_2, T_3, T_4, T_5, T_6, T_7, M, I)$.
   - (*Response.*) User computes $s_\alpha = r_\alpha - c\alpha$, $s_\beta = r_\beta - c\beta$, $s_x = r_x - cx$, $s_{\delta_\alpha} = r_{\delta_\alpha} - c\delta_\alpha$, $s_{\delta_\beta} = r_{\delta_\beta} - c\delta_\beta$, $s_a = r_a - ca$, $s_b = r_b - cb$ and $s_t = a - cb$. User sends $(\sigma, c, s_t)$ to merchant, where $\sigma = (A_1, A_2, A_3, B_1, B_2, s_\alpha, s_\beta, s_x, s_a, s_b, s_{\delta_\alpha}, s_{\delta_\beta})$.
   - (*Verify.*) Merchant computes
     * $\tilde{T}_1 = A_1^c u^{s_\alpha}$, $\tilde{T}_2 = A_2^c v^{s_\beta}$, $\tilde{T}_3 = A_1^{s_x} u^{-s_{\delta_\alpha}}$, $\tilde{T}_4 = A_2^{s_x} v^{-s_{\delta_\beta}}$,
     * $\tilde{T}_5 = \left(\frac{\hat{e}(g_1,g_2)}{\hat{e}(A_3,w)}\right)^c \frac{\hat{e}(A_3,g_2)^{s_x}}{\hat{e}(h_3,g_2)^{(s_{\delta_\alpha}+s_{\delta_\beta})} \hat{e}(h_3,w)^{(s_\alpha+s_\beta)} \hat{e}(h_1,g_2)^{s_a} \hat{e}(h_2,g_2)^{s_b}}$,
     * $\tilde{T}_6 = B_1^c {h_t}^{s_a}$, $\tilde{T}_7 = B_2^c {h_t}^{s_b}$.

     Accepts if $c \stackrel{?}{=} H(A_1, A_2, A_3, B_1, B_2, \tilde{T}_1, \tilde{T}_2, \tilde{T}_3, \tilde{T}_4, \tilde{T}_5, \tilde{T}_6, \tilde{T}_7, M, I)$ and $B_1 \stackrel{?}{=} B_2^c {h_t}^{s_t}$ both hold and rejects otherwise.

– `Deposit Protocol`. The merchant with identity $I$ sends the payment transcript $(\sigma, c, s_t)$ and $M$ to the bank. The bank verifies the payment transcript exactly as the merchant did. In addition, the bank has to verify that $I$ is indeed the identity of the merchant and $(M, \sigma)$ is not used before by that merchant. This is to prevent colluding users and merchants submitting double spent coin (which have completely identical transcript). The bank also checks for double-spending by searching if the $(B_1, B_2)$ is already existing in some entry in the deposit database. If it is not found, $(B_1, B_2, c, s_t)$ is stored and the payment is accepted as valid. Otherwise it is a doubly-spent coin.

– `Owner Tracing`. Let the two payment transcripts are $(\sigma, c, s_t)$ and $(\sigma', c', s_t')$, the bank computes $\hat{b} = \frac{s_t - s_t'}{c' - c}$ and $\hat{a} = s_t + c\hat{b}$. The private key and the public key of the double-spender are $\hat{s} = \hat{a}\hat{b}$ and $\hat{y} = h^{\hat{s}}$ respectively.

– `Coin Tracing`. The bank decrypts the value ${h_t}^a$ and ${h_t}^b$ for all other coins issued to the double-spender by the exposed key pair.

## 5.2 Shorter E-Cash

We can further shorten our payment transcript to 192 bytes with the XDH assumption. We highlight the changes from the short e-cash system as follow.

– `Bank Setup`. Basically the same except the bank's public key is shortened to $bpk = (g_1, g_2, w, h_1, h_2, h, u, v)$.

– `User Setup`. Basically the same except the group $\mathbb{G}_p$ is replaced with $\mathbb{G}_1$.
– `Withdrawal Protocol`. The coin $(A, x, a, b)$ is generated with the same mechanism and hence $A^{x+\gamma} = g_1 h_1^a h_2^b$ still holds. But the tracing information becomes $A_1 = u^a$ and $A_3 = u^b$. To accommodate the changes, we need a new SPK $\Pi_3$ instead of the original $\Pi_2$. Again $\Pi_3$ is shown in the appendix.
– `Payment Protocol`. User spends the coin $(A, x, a, b)$ to a merchant with the identity $I \in \{0,1\}^*$ by executing the following protocol.
  1. User computes auxiliary commitment $A_1 = u^a$, $A_2 = Av^a$, $A_3 = u^b$ and a helper value $\delta = xa$.
  2. User undertakes a proof of knowledge of values $(a, b, x, \delta)$ satisfying $A_1 = u^a$, $A_1^x = u^\delta$, $A_3 = u^b$, $\hat{e}(A_2, g_2)^x \hat{e}(v, g_2)^{-\delta} \hat{e}(v, w)^{-a} \hat{e}(h_1, g_2)^{-a}$ $\hat{e}(h_2, g_2)^{-b} = \frac{\hat{e}(g_1, g_2)}{\hat{e}(A_2, w)}$. This proof of knowledge proceeds as follow.
     - (*Auxiliary Commitment.*) User computes $A_1$, $A_2$, $A_3$ as above.
     - (*Commitment.*) User randomly selects $r_a, r_b, r_x, r_\delta \in_R \mathbb{Z}_p^*$, computes
       * $T_1 = u^{r_a}$, $T_2 = A_1^{r_x} u^{-r_\delta}$,
       * $T_3 = \hat{e}(A_2, g_2)^{r_x} \hat{e}(v, g_2)^{-r_\delta} \hat{e}(v, w)^{-r_a} \hat{e}(h_1, g_2)^{-r_a} \hat{e}(h_2, g_2)^{-r_b}$.
     - (*Challenge.*) Merchant sends the transaction information $M \in \{0,1\}^*$ to user. User computes $c = H(A_1, A_2, A_3, T_1, T_2, T_3, M, I)$.
     - (*Response.*) User computes $s_a = r_a - ca$, $s_b = r_b - cb$, $s_x = r_x - cx$, $s_\delta = r_\delta - c\delta$ and $s_t = a - cb$. User sends $(\sigma, c, s_t)$ to the merchant, where $\sigma = (A_1, A_2, A_3, s_a, s_b, s_x, s_\delta)$.
     - (*Verify.*) Merchant computes $\tilde{T_1} = A_1^c u^{s_a}$, $\tilde{T_2} = A_1^{s_x} u^{-s_\delta}$ and $\tilde{T_3} = (\frac{\hat{e}(g_1, g_2)}{\hat{e}(A_2, w)})^c \hat{e}(A_2, g_2)^{s_x} \hat{e}(v, g_2)^{-s_\delta} \hat{e}(v, w)^{-s_a} \hat{e}(h_1, g_2)^{-s_a} \hat{e}(h_2, g_2)^{-s_b}$.
       Accepts if both of $c \overset{?}{=} H(A_1, A_2, A_3, \tilde{T_1}, \tilde{T_2}, \tilde{T_3}, M, I)$ and $A_1 \overset{?}{=} A_3^c u^{s_t}$ hold, rejects otherwise.
– `Deposit Protocol`. Merchant sends the payment transcript $(\sigma, c, s_t)$ to bank for deposit. In the enhanced protocol, double-spending is identified by the pair $(A_1, A_3)$ (instead of $(B_1, B_2)$).
– `Owner Tracing`. Suppose the two transcripts are $(\sigma, c, s_t)$ and $(\sigma', c', s_t')$, the bank computes $\hat{b} = \frac{s_t - s_t'}{c' - c}$ and $\hat{a} = s_t + c\hat{b}$. The private key and the public key of the double-spender are $\hat{s} = \hat{a}\hat{b}$ and $\hat{y} = h^{\hat{s}}$ respectively.
– `Coin Tracing`. The bank can decrypt the values $u^a$ and $u^b$ for all other coins issued to the double-spender for tracing.

## 5.3 Short E-Cash with TTP

In some scenario, the law enforcing agency got the knowledge of a certain criminal by non-cryptographic means, and wants to stop this user from using his coins (which has already been withdrawn). This can be achieved by incorporating a TTP in our scheme for revoking identity and coin tracing of *all* users.

For our first proposed scheme, instead of having $h_3$, $u$, $v$ generated fairly, the TTP selects $\xi_1$, $\xi_2$ such that $h_3 = u^{\xi_1} = v^{\xi_2}$. The TTP can revoke the identity of every spender by computing $A = A_3/(A_1^{\xi_1} A_2^{\xi_2})$ and identifying the spender from the withdrawal transcript. For the shorter version, TTP's private-public key pair is $(\xi, v = u^{\xi})$. To revoke the identity of the spender, TTP computes $A = A_2/(A_1^{\xi})$ for the bank to identify the spender from the withdrawal protocol.

Coin tracing can be achieved by requiring users to encrypt tracing information ($\{h_t^a, h_t^b\}$, or $\{u^a, u^b\}$ for the shorter version) under TTP's public key. In fact, coin tracing and owner tracing power can be held by different TTP, and each feature can be independently incorporated, by using different proofs in $SPK$. Due to space limitations, details can be found in the full paper.

## 5.4 Security Analysis

The security of our system is assured by the following theorems. Their proofs can be found in the full version of this paper. The security analysis of the shorter version goes in a similar way.

**Theorem 1 (Balance).** *Our proposed construction has the balance property under the q-SDH assumption, in the random oracle model.*

**Theorem 2 (Anonymity).** *Our proposed construction has the anonymity property under the DLDH assumption in $\mathbb{G}_1$ and DDH assumption in $\mathbb{G}_p$, in the random oracle model.*

## 5.5 Comparison with Compact E-Cash

We compare the bandwidth and the storage requirement of our scheme with the second scheme in [8] (which supports full coin-tracing). In the following comparison, we instantiate the CHL scheme with a 1024-bit RSA modulus. For our scheme, we take $p$ be a 170-bit prime with the families of curves described in [5]. Using the standard point compression technique, each element in $\mathbb{G}_1$ is 171-bit. Each coin consists of one element in $\mathbb{G}_1$ and three elements in $\mathbb{Z}_p^*$. The coin size is thus 681 bits. Each payment transcript contains three elements in $\mathbb{G}_1$ $(A_1, A_2, A_3)$, two elements in $\mathbb{G}_p$ $(B_1, B_2)$ and nine element in $\mathbb{Z}_p^*$, making its length 512 bytes, if we assume elements in $\mathbb{G}_p$ is representable in 1024 bits. As for the shorter version, each payment transcript contains three elements in $\mathbb{G}_1$ $(A_1, A_2, A_3)$ and six elements in $\mathbb{Z}_p^*$, making its length 192 bytes.

In the CHL scheme, the withdrawal protocol enables the user to withdraw $2^{\ell}$ coins at a time. For the payment and deposit protocols, only one coin is processed each time. The space complexity of the withdrawal, payment and deposit transcript are all of order $\ell$. In the payment protocol, the user needs to compute $7 + 9\ell$ auxiliary commitments together with $17 + 21\ell$ commitments during the SPKs, and the response takes about $20\ell$ elements. The payment transcript size is about $(24 + 50\ell) \times 1024$ bits. Taking $\ell = 10$, spending one coin requires transmission bandwidth of $1024 \times (24 + 500)$ bits, i.e. around 60 Kilobytes. In our scheme, each payment transcript is of constant size 512 bytes. Our scheme's bandwidth requirement in payment is 100 times more efficient.

The withdrawal protocol of the CHL scheme require some more investigation. Without counting the verifiable encryption, the bandwidth required for withdrawing $2^\ell$ coins is $(2+3\ell) \times 1024$ bits, which is very efficient per coin. However, the verifiable encryption is rather inefficient in itself. For a cheating probability lower than $2^k$, the user is required to perform $2k$ encryptions while the bank must perform $k$ encryptions. After this process, the bank needs to store all these $2k$ encryption transcripts later decryption. The verifiable encryption on $s$ is to be performed with relative to the Pedersen commitment $A = {g_0}^r {g_1}^u {g_2}^s {g_i}^{t_i}$ for $i = 3$ to $3\ell{+}3$. Precisely speaking, $k$ rounds of the verifiable encryption has to be done, with each round consisting of one commitment, two bilinear El Gamal encryptions, and $3\ell + 3$ responses (the $3\ell + 3$ term arise since the user has to proof that encryption on $s$ is correctly formed with respect to $A$, which contains $3 + 3\ell$ exponents). Suppose each component is of size 1024-bit, the total transcript size is $(4 + 3\ell)/8$ kilobytes for each round, making the total transmission requirement of $k(4 + 3\ell)/8$ kilobytes.

A simple trick to simplify the computation is to compute another Pedersen commitment $B = g_0^{r'} {g_1}^s$, proved that the term $s$ in both $A$ and $B$ are the same, and do the verifiable encryption with respect to $B$. In this case, each round is of size $5 \times 1024$ bits, and a total of $5k/8$ kilobytes for $k$ rounds. After that, the bank need to store this $5k/8$ kilobytes of information for later decryption. Thus, bandwidth requirement for CHL's withdrawal protocol per coin (including verifiable encryption using the improved method) is $\frac{2+3\ell+5k}{8 \cdot 2^\ell}$ kilobytes.

For a cheating probability of $0.001(k = 10)^2$ and taking $\ell = 10$ , the average storage per coin required is 10 bytes, using the improved protocol. In our scheme, this kind of inefficient verifiable encryption is not needed with the help of SPK $\Pi_2$. A total of 873 bytes is required for each coin (the number of bits required by SPK $\Pi_1$, $\Pi_2$ is 1363 and 5627 bits respectively), and the bank only needs to store 512 bytes of the encrypted information for each coin.

In short, our scheme is about 50 times less efficient per coin in the withdrawal protocol, and 100 times more bandwidth efficient per coin during the payment protocol and the deposit protocol. Withdrawal can be done by a desktop while the payment may be done in a mobile device with lower computational power and storage. We believe that our scheme is an improvement over the CHL scheme.

## 6 Conclusion

Double spender tracing is important in an anonymous e-cash system. Coin tracing may be even more important as the bank can freeze the possible misbehavior of a double-spender. Most existing systems relies on the existence of an overpowered TTP, which may identify the spender of a coin and trace all the coins by a particular spender, even the spender is an honest one who never double-spend. Recently, Camenisch, Hohenberger and Lysyanskaya proposed an e-cash system with traceable coins [8]. Once a user double-spends, his identity can be revealed and all his coins in the system can be traced, *without* resorting to TTP. Their scheme is “compact” in the sense that a user can withdraw $2^\ell$ coins in a

[2] It is worth noting that using $k = 10$ is in favor of the CHL scheme since the cheating probability of our scheme is $1/q$ with $q$ being a 170-bit prime.

single withdrawal protocol with the cost of $O(\ell \cdot k)$, and the coins can be spent in an unlinkable manner. This result is theoretically very efficient. However, we identify that the bandwidth requirements in payment and deposit protocol, and the bank's storage, may not be efficient for realistic scenario.

In this paper, we present a bandwidth-efficient off-line anonymous e-cash scheme with traceable coins. For a security level comparable with 1024-bit standard RSA signature, the payment transcript size is only 512 bytes. Security of the proposed scheme is proven under the $q$-strong Diffie-Hellman assumption and the decisional linear assumption, in the random oracle model. The transcript size of our scheme can be further reduced to 192 bytes if external Diffie-Hellman assumption is made. To the best of authors' knowledge, it is the shortest e-cash system currently available. We also show how to incorporate a TTP that is responsible for the owner-tracing and coin-tracing, if such a TTP is desired.

## References

1. Masayuki Abe and Eiichiro Fujisaki. How to Date Blind Signatures. *Advances in Cryptology - ASIACRYPT 1996*, *LNCS* 1163, pp. 244–251. Springer, 1996.
2. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. *Advances in Cryptology - CRYPTO 2000*, *LNCS* 1880, pp. 255–270. Springer, 2000.
3. Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles. *Advances in Cryptology - EUROCRYPT 2004*, *LNCS* 3027, pp. 56–73. Springer, 2004.
4. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. *Advances in Cryptology - CRYPTO 2004*, *LNCS* 3152, pp. 41–55. Springer, 2004.
5. Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. *Advances in Cryptology - ASIACRYPT 2001*, *LNCS* 2248, pp. 514–532.
6. Stefan Brands. Untraceable Off-line Cash in Wallets with Observers. *Advances in Cryptology - CRYPTO '93*, *LNCS* 773, pp. 302–318. Springer, 1994.
7. Ernie Brickell, Peter Gemmell, and David Kravitz. Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Change. *SODA '95: ACM-SIAM Symposium on Discrete Algorithms*, pp. 457–466. SIAM, 1995.
8. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-Cash. *Advances in Cryptology - EUROCRYPT 2005*, *LNCS* 3494, pp. 302–321. Springer.
9. Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups. *Advances in Cryptology - CRYPTO '97*, *LNCS* 1294, pp. 410–424.
10. Sébastien Canard and Jacques Traoré. On Fair E-cash Systems Based on Group Signature Schemes. *Information Security and Privacy, 8th Australasian Conference, ACISP 2003*, *LNCS* 2727, pp. 237–248. Springer, 2003.
11. David Chaum. Blind Signatures for Untraceable Payments. *Advances in Cryptology: CRYPTO '82*, pp. 199–203. Plenum, New York, 1983.
12. David Chaum, Amos Fiat, and Moni Naor. Untraceable Electronic Cash. *Advances in Cryptology - CRYPTO '88*, *LNCS* 403, pp. 319–327. Springer, 1990.
13. David Chaum and Eugène van Heyst. Group Signatures. *Advances in Cryptology - EUROCRYPT '91*, *LNCS* 547, pp. 257–265. Springer, 1991.
14. Niels Ferguson. Single Term Off-Line Coins. *Advances in Cryptology - EUROCRYPT '93*, *LNCS* 765, pp. 318–328. Springer, 1994.

15. Matthew K. Franklin and Moti Yung. Secure and Efficient Off-Line Digital Money (Extended Abstract). *Automata, Languages and Programming, 20th International Colloquium, ICALP '93*, *LNCS* 700, pp. 265–276. Springer, 1993.
16. Stanislaw Jarecki and Vitaly Shmatikov. Handcuffing Big Brother: An Abuse-Resilient Transaction Escrow Scheme. *Advances in Cryptology - EUROCRYPT 2004*, *LNCS* 3027, pp. 590–608. Springer, 2004.
17. Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable Signatures. *Advances in Cryptology - EUROCRYPT 2004*, *LNCS* 3027, pp. 571–589. Springer, 2004.
18. Anna Lysyanskaya and Zulfikar Ramzan. Group Blind Digital Signatures: A Scalable Solution to Electronic Cash. *Financial Cryptography 98*, *LNCS* 1465, pp. 184–197. Springer, 1998.
19. Greg Maitland and Colin Boyd. Fair Electronic Cash Based on a Group Signature Scheme. *Information and Communications Security, 3rd International Conference, ICICS 2001*, *LNCS* 2229, pp. 461–465. Springer, 2001.
20. Toru Nakanishi, Nobuaki Haruna, and Yuji Sugiyama. Unlinkable Electronic Coupon Protocol with Anonymity Control. *Information Security, Second International Workshop, ISW'99*, *LNCS* 1729, pp. 37–46. Springer, 1999.
21. Tatsuaki Okamoto. An Efficient Divisible Electronic Cash Scheme. *Advances in Cryptology - CRYPTO '95*, *LNCS* 963, pp. 438–451. Springer, 1995.
22. Tatsuaki Okamoto and Kazuo Ohta. Disposable Zero-Knowledge Authentications and Their Applications to Untraceable Electronic Cash. *Advances in Cryptology - CRYPTO '89*, *LNCS* 435, pp. 481–496. Springer, 1990.
23. Weidong Qiu, Kefei Chen, and Dawu Gu. A New Offline Privacy Protecting E-cash System with Revokable Anonymity. *Information Security, 5th International Conference, ISC 2002*, *LNCS* 2433, pp. 177–190. Springer, 2002.
24. Markus Stadler, Jean-Marc Piveteau, and Jan Camenisch. Fair Blind Signatures. *Advances in Cryptology - EUROCRYPT '95*, *LNCS* 921, pp. 209–219. Springer.
25. Jacques Traoré. Group Signatures and Their Relevance to Privacy-Protecting Off-Line Electronic Cash Systems. *Information Security and Privacy, 4th Australasian Conference, ACISP'99*, *LNCS* 1587, pp. 228–243. Springer, 1999.
26. Patrick P. Tsang and Victor K. Wei. Short Linkable Ring Signatures for E-Voting, E-Cash and Attestation. *Information Security Practice and Experience, First International Conference, ISPEC 2005*, *LNCS* 3439, pp. 48–60. Springer, 2005.
27. Sebastiaan von Solms and David Naccache. On Blind Signatures and Perfect Crimes. *Computer Security*, 11(6):581–583, 1992.

## A Signature Knowledge of Representation

A signature of knowledge allows a signer to prove the knowledge of a secret with respect to some public information non-interactively by tying his knowledge of a secret to a message begin signed. Following the notion in [9], we called these signature "Signatures based on Proofs of Knowledge" (SPK).

As an example, we denote the zero-knowledge proof of the discrete logarithm of $y$ by $SPK\{(x) : y = g^x\}(M)$, where $M$ is the hash value of the commitment.

The SPK $\Pi_1$, $\Pi_2$ and $\Pi_3$ used in our proposal are shown below.

$\Pi_1 = \mathrm{SPK}\{(\bar{a}, \bar{b}, s, r_1, \delta) : \bar{C} = h_1^{\bar{a}} h_2^{\bar{b}} \bigwedge A_1 = h_1^{r_1} h_2^{\bar{a}} \bigwedge A_1^{\bar{b}} = h_1^{\delta} h_2^{s} \bigwedge y = h^s\}(M)$ where $M = H(\bar{C}, A_1, y, h_1, h_2, )$.

For $\Pi_2$, first compute $A_1 = \bar{C}^r$ and $A_2 = \bar{C}^{r^{-1}}$ and execute the following SPK: $\Pi_2 = \mathrm{SPK}\{(\bar{a}, \bar{b}, a, b, \delta_a, \delta_b, t_a, t_b) : \bar{C} = h_1^{\bar{a}} h_2^{\bar{b}} \bigwedge A_1 = h_1^{a} h_2^{\delta_b} \bigwedge A_2 =$

$h_1^{\delta_a} h_2^b \bigwedge C = h_1^a h_2^b \bigwedge R_1 = h^{t_a} \bigwedge R_2 = y^{t_a} h_t^a \bigwedge R_3 = h^{t_b} \bigwedge y^{t_b} h_t^b \}(M)$, where $M = H(\bar{C}, A_1, A_2, C, R_1, R_2, R_3, R_4, h_1, h_2, y)$. Note that $R_1$,$R_2$,$R_3$,$R_4$ is the encryption of $h_t^a$ and $h_t^b$ under the public key $y = h^s$.

For $\Pi_3$, first compute $A_1 = \bar{C}^r$ and $A_2 = \bar{C}^{r^{-1}}$ and execute the following SPK: $\Pi_3 = \mathrm{SPK}\{(\bar{a}, \bar{b}, \delta_a, \delta_b,\ a, b, s) : \bar{C} = h_1^{\bar{a}} h_2^{\bar{b}} \bigwedge A_1 = h_1^a h_2^{\delta_b} \bigwedge A_2 = h_1^{\delta_a} h_2^b \bigwedge C = h_1^a h_2^b \bigwedge y = h^s \bigwedge R_1 = h^{t_1} \bigwedge R_2 = y^{t_1} u^a \bigwedge R_3 = h^{t_2} \bigwedge R_4 = y^{t_2} u^b \}(M)$, where $M = H(\bar{C}, C, A_1, A_2, R_1, R_2, R_3, R_4, y, h_1, h_2, u, v)$. Note that $R_1$, $R_2$, $R_3$, $R_4$ is the encryption of $u^a$ and $v^b$ under the public key $y = h^s$.

# A Universally Composable Scheme for Electronic Cash

Mårten Trolin

Royal Institute of Technology (KTH),
Stockholm, Sweden
`marten@nada.kth.se`

**Abstract.** We propose a scheme for electronic cash based on symmetric primitives. The scheme is secure in the framework for universal composability assuming the existence of a symmetric CCA2-secure encryption scheme, a CMA-secure signature scheme, and a family of one-way, collision-free hash functions. In particular, the security proof is *not* in the random-oracle model. Due to its high efficiency, the scheme is well-suited for devices such as smart-cards and mobile phones. We also show how the proposed scheme can be used as a group signature scheme with one-time keys.

## 1 Introduction

### 1.1 Background and Previous Work

The concept of electronic cash, e-cash, was introduced by Chaum et al. [6], and several subsequent schemes have been proposed [2, 8, 20, 18, 16, 14, 13, 3]. In an e-cash scheme there are three types of participants – the bank, merchants, and users. The users can withdraw coins from the bank and spend them at merchants. An e-cash scheme is *online* or *offline*. In the former case the bank is involved in every transaction, whereas in the second case payments can be performed without contacting the bank. Obviously offline schemes are preferable to online schemes. However, an electronic coin, being nothing but a string of numbers, can be copied and spent more than once, and in an offline scheme such double-spendings cannot be detected during the actual purchase. Rather than preventing double-spending, offline schemes are designed so that double-spenders are detected and identified.

Privacy is a crucial ingredient of e-cash schemes. It is desirable that merchants cannot learn the identity of the user, or even determine whether two payments were made by the same user or not. Many schemes also provide the same privacy towards the bank. However, anonymity also works in favor of criminals using the scheme for illegal activities protected by the privacy offered. To protect against such events some schemes offer the possibility for trusted third parties to trace a payment.

Most schemes require a merchant to deposit a coin after the purchase. A few schemes allow a coin to be transferred between users in several steps before it is

© Springer-Verlag Berlin Heidelberg 2005

deposited at the bank [15, 16]. Such schemes are said to have *transferable coins.* Another possible feature is divisability, i.e., that a coin may be spent only in part [16, 14, 13].

## 1.2 Group Signatures and E-Cash Schemes

Group signatures were introduced by Chaum and van Heyst [7]. In a group signature scheme there is a *group manager* and *group members.* The group manager delegates the right to generate signatures to the group members, and also publishes a *group public key.* Members can sign messages, and a signature can be verified against the group public key, but only the group manager can open a signature to learn the identity of the signer. To anyone else the signature is anonymous.

Group signatures bear many resemblances to electronic cash. Group signatures are indistinguishable to anyone but the group manager in very much the same way payments are indistinguishable in anonymous e-cash schemes. One important difference is that there is no concept of double-spending for group signatures. See, e.g., [11] for an example of an e-cash scheme based on group signatures.

## 1.3 Our Contribution

All the above schemes involve trapdoor functions such as variants of ElGamal encryption or RSA groups. A real-life electronic cash scheme would probably be implemented on a portable device with low computational power such as a smart-card or a mobile phone. For such schemes it is important that the amount of computation is low, especially on the user side. The difference between zero, one or two exponentiations in the payment protocol is significant, whereas many schemes require tens, or in some cases hundreds, of exponentiations. The merchant terminal is more comparable to a low-end PC, but also in this case it is desirable to reduce the amount of computation to one or a few exponentiations.

**Outline of Scheme.** In this paper we propose a scheme which relies on symmetric primitives such as symmetric encryption, hash functions and pseudo-random functions. The only computations performed by the user during payment is evaluation of pseudo-random functions, and the merchant verifies a signature. It is commonly believed that there exist efficient algorithms for the primitives needed, e.g., AES and SHA-256. The scheme has been implemented on a mobile platform [21]. Our scheme builds along the lines of the scheme by Sander and Ta-Shma [18].

When a user $\mathcal{U}$ withdraws a coin, the bank encrypts the identity of $\mathcal{U}$. Then $\mathcal{U}$ uses a pseudo-random function to create a list of values and sends the hash values of the pseudo-random values to the bank. The coin, consisting of the encrypted identity and the hash values, is inserted as a leaf into a Merkle hash tree. After a certain amount of time, the bank builds the tree and publishes the root. To spend a coin, the user reveals half of the preimages of the hash values together with a path from the coin up to a published root. The merchant verifies the correctness of the preimages, and verifies that the chain of hash values is valid.

If a user double-spends a coin, then she has revealed the preimage of more than half of the hash values. If this happens the bank decrypts the encrypted identity. From only the revealed preimages of a double-spent coin, it may be possible to successfully spend the coin a third time. In other words, a user double-spending a coin risks being held responsible for additional purchases. This gives additional incentive not to double-spend. The anonymity of the scheme follows from the security of the encryption scheme, and unforgeability of coins follows since the hash function is collision-free.

As an additional feature of our scheme the payment protocol is non-interactive. In other words, the user produces a coin that can only be deposited by the designated merchant. This enables a user to prepare a coin for a certain merchant. In addition, anyone can verify that the coin has been prepared for that merchant. As an example, a parent can give a coin to their child which can be spent only at a certain store.

We show security for our scheme in the framework for universal composability (UC) [4]. We stress that our proof of security is in the plain model and not in the random-oracle model. We only assume that the encryption scheme is CCA2-secure, that the hash functions are one-way and collision-free, and that the pseudo-random functions are indistinguishable from random functions. We believe that the current scheme is the first scheme for electronic cash with a security proof in the UC-model and also the first practical scheme that does not use the random-oracle model for its security proof.

Previous e-cash schemes that are secure in the plain model include [18] and [10]. The former uses zero-knowledge proofs based on general methods, and the latter is a blind signature scheme using general methods for two-party computation from which an e-cash scheme may be built. Neither of these is practical.

Our scheme does not offer the same anonymity towards the bank as many other schemes. It is an interesting question whether a scheme that does not involve trapdoor functions can offer the anonymity towards the bank in the same strong sense as, e.g., [6]. For a more thorough discussion on this, see the full version [19].

**Relations to Group Signatures.** Although proposed as a scheme for electronic cash, our scheme has some similarities with group signatures. The bank has the ability to open a coin to extract the identity in the same way the group manager can open a signature. As a matter of fact, our scheme can be seen as a group signature scheme with one-time keys. This is discussed in further detail in Section 6.

**Comparison to Sander-Ta-Shma.** As in the scheme by Sander and Ta-Shma [18], in our scheme the bank builds a hash tree and the merchant uses the published root when verifying a coin. In their scheme a zero-knowledge protocol is used by the user to prove ownership of a preimage of a hash value and a path to some certified root. Focusing on efficiency, we avoid the zero-knowledge proof by letting the user reveal preimages of $\kappa/2$ out of $\kappa$ hash values. The cost for this efficiency increase is that the bank always can identify the payer.

## 2 Notation and Definitions

### 2.1 Notation

String concatenation is denoted by $||$. For two integers $a$ and $b$ their concatenation $a||b$ is the number created by concatenating their binary representations, e.g., $a||b = 2^{k_b}a + b$, if $b$ is a $k_b$-bit number. By $s \leftarrow_R S$ we mean that $s$ is chosen independently and uniformly at random from the finite set $S$.

We define $[n] = \{1, 2, \ldots, n\}$. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_k\}$, $i_j < i_{j+1}$, be a subset of $[n]$. For a list of values $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ we define $\boldsymbol{v}_{\mathcal{I}} = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})$. Let $f$ be a function, $S = \{s_1, s_2, \ldots, s_m\}$ a set, and $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ a vector. We define $f(S) = \{f(s_1), f(s_2), \ldots, f(s_m)\}$ and $f(\boldsymbol{v}) = (f(v_1), f(v_2), \ldots, f(v_n))$.

### 2.2 Basic Definitions

A function $\varepsilon$ is *negligible* if $\varepsilon(\kappa) < 1/p(\kappa)$ for any polynomial $p(\kappa)$ and sufficiently large $\kappa$.

We define hash functions and pseudo-random functions as families of functions. The most common way to realize a family of functions is to define the function such that it depends on a key. A function is drawn from the family by generating a key.

We use hash functions that are collision-free, sometimes called collision-resistant, and one-way. Let $\mathcal{H}_\kappa$ be a family of hash functions that map values in $\{0,1\}^*$ to $\{0,1\}^\kappa$, and let $\mathcal{H} = \{\mathcal{H}_i\}_{i=1}^\infty$. Intuitively $\mathcal{H}$ is collision-free if it is infeasible to find two distinct inputs that hash to the same value and one-way if it is hard to compute a preimage of a random value for $H \leftarrow_R \mathcal{H}_\kappa$.

Let $\mathcal{R}_\kappa$ be a family of functions from $\{0,1\}^\kappa$ to $\{0,1\}^\kappa$, and let $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^\infty$. Let $\mathcal{U}_\kappa$ be the family of all functions from $\{0,1\}^\kappa$ to $\{0,1\}^\kappa$. Informally $\mathcal{R}$ is said to be pseudo-random if it is infeasible to distinguish a function from $\mathcal{R}_\kappa$ from a function from $\mathcal{U}_\kappa$.

A signature scheme $\mathcal{SS} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Vf})$ is *correct* if $\mathsf{Vf}_{\mathsf{pk}}(m, \mathsf{Sig}_{\mathsf{sk}}(m)) = 1$ for $(\mathsf{pk}, \mathsf{sk})$ generated by $\mathsf{Kg}$ and any message $m$. $\mathcal{SS}$ is secure against chosen-message attacks, CMA-secure [9], if it is infeasible to produce valid message-signature pair for *any* message, even if the adversary has access to a signing oracle $\mathsf{Sig}_{\mathsf{sk}}(\cdot)$.

A symmetric encryption scheme $\mathcal{CS} = (\mathsf{Kg}, E, D)$ is secure against a chosen cipher-text attack, CCA2-secure, if it is infeasible to distinguish between encryptions of two messages of the adversary's choice, even if the adversary is given access to an encryption oracle and a decryption oracle. This is a natural extension of CCA2-security for asymmetric encryption schemes [17].

Please see the full version [19] for precise security definitions.

## 2.3 Merkle Trees and Hash Chains

Consider the task of proving that a value belongs to a set of certified values. One way to achieve this is to create a binary tree with the values as leaves by setting the value of every inner node to the hash value of the concatenation of the values of its two children and publish the root in a certified way. This is called a *Merkle tree* [12].

From a Merkle tree a *hash chain* from each leaf up to the root of the tree can be constructed. For each step the chain contains a value and an order bit which says whether the given value should be concatenated from the left or from the right.

An example of a Merkle tree is given in Figure 1. From the tree in the figure we can construct a hash chain from $c_{121}$ up to the root as $(c_{121}, \rho, v_2, r, v_{11}, l, c_{122}, r)$. The values in the chain from $c_{121}$ have been circled in the figure. Note that $v_1$ and $v_{12}$ are not part of the chain, since these values are computed during verification.

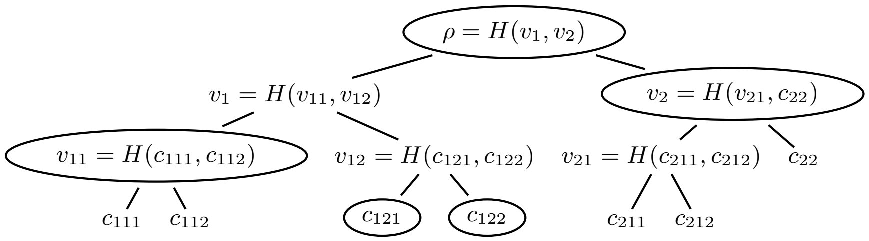
**Definition 1 (Hash chain).** *A hash chain $h$ of length $d$ is a vector $h = (v, h_0, h_1, o_1, h_2, o_2, \ldots, h_{d-1}, o_{d-1})$ where $o_i \in \{l, r\}$. A hash chain is said to be valid under a hash function $H$ if $h_0 = h'_0$, where $h'_{d-1} = v$ and*

$$h'_{i-1} = \begin{cases} H(h_i \| h'_i) \text{ if } o_i = l \\ H(h'_i \| h_i) \text{ if } o_i = r \end{cases}$$

*for $i = d-1, d-2, \ldots, 1$. This is written $\mathsf{isvalid}_H(h) = 1$, or $\mathsf{isvalid}(h) = 1$ if it is clear from the context which hash function is used. We also define $\mathsf{root}(h) = h_0$ and $\mathsf{leaf}(h) = v$.*

Once a Merkle tree has been built for a set of values and its root value has been published, constructing a hash chain for a value not in the set implies finding a collision for the hash function. Since this is assumed to be infeasible, Merkle trees give a method of proving membership.

We define the randomized function $\mathsf{buildtree}_H(S)$ with input a set $S = \{s_1, s_2, \ldots, s_n\}$ to build and output a hash tree of depth $\lceil \log_2 n \rceil$ where the leaves have values $s_1, \ldots, s_n$ in random order. When $n$ is a power of two, all



$\rho = H(v_1, v_2)$
$v_1 = H(v_{11}, v_{12})$
$v_2 = H(v_{21}, c_{22})$
$v_{11} = H(c_{111}, c_{112})$
$v_{12} = H(c_{121}, c_{122})$
$v_{21} = H(c_{211}, c_{212})$
$c_{22}$
$c_{111}$
$c_{112}$
$c_{121}$
$c_{122}$
$c_{211}$
$c_{212}$

**Fig. 1.** A Merkle tree with the values stored in the hash chain from $c_{121}$ to the root marked

leaves have equal depth $d-1$, and otherwise some leaves have depth $d-2$. The function $\mathsf{getchain}_T(s)$ returns the hash chain from the first leaf with value $s$ to the root in the tree $T$ and $\emptyset$ if no such leaf exists.

It is possible to join two Merkle trees into a new Merkle tree by creating a new root which has the two trees as children. Sander and Ta-Shma describe how the number of active roots can be reduced by joining the existing trees. In our scheme we do not join trees in this way, although the scheme could be modified to do so.

## 3 The Protocol

### 3.1 Security Parameters

Two security parameters, $\kappa_1$ and $\kappa_2$, are used in the protocol. The parameter $\kappa_1$ can be thought of as key length for the symmetric cipher, and $\kappa_2$ is the number of bits needed so that each merchant can be identified by a $\kappa_2$-bit number with $\kappa_2/2$ number of ones.

### 3.2 The Players

The players in the protocols are denoted $\mathcal{B}, P_1, \ldots, P_m$. To simplify the description we also write $P_0$ for $\mathcal{B}$. Except for the bank, any player may act as a customer, i.e., withdraw and spend coins, as well as a merchants, i.e., accept payments and deposit coins. We abuse notation and let $P_i$ represent both the identity of the player and the Turing machine taking part in the protocol.

We let $\mathcal{I}$ be a public map from identities to $[\kappa_2]$ such that $\mathcal{I}(P_i)$ has cardinality $\kappa_2/2$ and $\mathcal{I}(P_i) \neq \mathcal{I}(P_j)$ for $P_i \neq P_j$. $\mathcal{I}$ can be thought of as a collision-free hash function which maps its input to $\{0,1\}^{\kappa_2}$ with the additional property that the number of 1's in the output is always exactly $\kappa_2/2$, although it is probably more practical to realize the map with a table. We define

$$\mathsf{span}_{\mathcal{I}}\left(\{P_{i_1}, P_{i_2}, \ldots, P_{i_k}\}\right) = \left\{ P \mid \mathcal{I}(P) \subseteq \bigcup_{j=1}^{k} \mathcal{I}(P_{i_j}) \right\} .$$

Given preimages of a coin corresponding to players $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$ one can combine the preimages to spend the coin at any player in $\mathsf{span}_{\mathcal{I}}(\{P_{i_1}, P_{i_2}, \ldots, P_{i_k}\})$. It holds that $P \in \mathsf{span}_{\mathcal{I}}(S)$ if $P \in S$. Since $\mathcal{I}$ is injective $\mathsf{span}_{\mathcal{I}}(\{P\}) = \{P\}$.

## 4 The Ideal Functionality

### 4.1 Introduction

In this section we define the ideal functionality $\mathcal{F}_{\mathrm{EC}}$ and discuss why it captures the properties of an e-cash scheme.

We use a model where the ideal functionality is linked to the players through a communication network $\mathcal{C}_{\mathcal{I}}$. The communication network forwards a message $m$ from a player $P$ as $(P, m)$ to the ideal functionality. When $\mathcal{C}_{\mathcal{I}}$ receives $(P, m)$ from the functionality, it forwards the message $m$ to player $P$. Except for immediate functions, defined as a message from a player $P$ immediately followed by a response to the same player $P$, the ideal adversary $\mathcal{S}$ is informed of when a message is sent, but not of the content. The ideal adversary is allowed to delay the delivery of such a message, but not change its content.

The functionality described here has only one non-immediate function – the withdrawal protocol.

The adversary is allowed to choose an arbitrary number of players to corrupt at start-up. For further discussion on this, see the full version [19]. We do not allow the adversary to corrupt $\mathcal{B}$. It would be possible to give a functionality that allows the adversary to corrupt $\mathcal{B}$. In such a functionality even a corrupted $\mathcal{B}$ would not be able to "revoke" an issued coin. However, since this would make the functionality more complex, we describe $\mathcal{F}_{\text{EC}}$ for a trusted bank.

## 4.2 Informal Description

The ideal functionality $\mathcal{F}_{\text{EC}}$ for an e-cash scheme accepts the following messages.

– `KeyGen` to set up keys.
– `Issue Coin` to issue a coin to the designated user.
– `Tick` to build a new hash tree.
– `Prepare Coin` to mark a coin for spending at a certain merchant.
– `Verify Coin` to verify whether or not a coin can be spent at a certain merchant.
– `Open Coin` to let the bank extract the identity of the user the coin was issued to.
– `Check Doublespent` to check whether a coin has been spent more than once.

There is no separate message for depositing a coin at the bank. To deposit the merchant hands the coin to the bank, which runs the `Verify Coin` algorithm to check if the coin is valid.

## 4.3 Definition of the Ideal Functionality

The ideal functionality $\mathcal{F}_{\text{EC}}$ holds a counter $t$ that is initialized to 0 and indexed sets $C_i$ for coins that have been issued in period $i$. For convenience we let $C = \bigcup_i C_i$. For $e = (c, \cdot, k, \cdot) \in C$ we define $\mathsf{val}_H(e) = c||H(k_1)||H(k_2)|| \cdots ||H(k_{\kappa_2})$. The functionality holds a set of signed roots $T_{\text{signed}}$ and a set of coins ready to be spent $T_{\text{prepared}}$. The sets $C_i, T_{\text{signed}}, T_{\text{prepared}}$ are initialized to $\emptyset$. The tables stored by the functionality are summarized in Table 1.

The functionality must be indistinguishable from the protocol, which implies that it must output data on the same format as the real protocol, and therefore it must depend on the implementation of the real protocol. This can be achieved

**Table 1.** The tables stored by the ideal functionality $\mathcal{F}_{\mathrm{EC}}$

| Name | Content |
|---|---|
| $C_i$ | $(c, P_i, \boldsymbol{k}, h)$, where $c$ is a bit-string, $P_i$ the coin-owner, $\boldsymbol{k}$ the coin-secret and $h$ the hash chain. |
| $T_{\mathrm{prepared}}$ | Coins which are about to be spent. |
| $T_{\mathrm{signed}}$ | The certified roots. |

by querying the ideal adversary for any such output, or the functionality can produce the output itself. In the former case, the ideal adversary needs to be tailor-made for a certain implementation of the protocol, whereas in the latter case the functionality must be parameterized by the implementation. We choose the second approach in this paper.

The functionality is parameterized by a symmetric encryption scheme $\mathcal{CS} = (\mathsf{Kg}, E, D)$, a signature scheme $\mathcal{SS} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Vf})$, a family of pseudo-random functions $\mathcal{R}$ and collision-free, one-way hash functions $H$ drawn from a family of hash functions $\mathcal{H}_{\kappa_1}$. To simplify the description we assume that $\mathcal{SS}$ is correct. Instead of parameterizing the functionality it is possible to give a non-parameterized description, where the functionality is given (a description of) the function families from $\mathcal{S}$ at startup. The definition of $\mathcal{F}_{\mathrm{EC}}$ is given in Figure 2.

$\mathcal{F}_{\mathrm{EC}}$ captures some specifics of the current scheme, such as the tree update function, the specific format of a coin, the weaker anonymity, a non-interactive payment protocol, and the possibility to transfer prepared coins to other users. Therefore a generic ideal functionality for electronic cash would differ from ours.

## 4.4 On the Ideal Functionality

In this section we discuss why $\mathcal{F}_{\mathrm{EC}}$ captures the security requirements for electronic cash. The five messages `KeyGen`, `Issue Coin`, `Tick`, `Prepare Coin`, and `Check Doublespent` are all straight-forward. They manipulate tables, and use $H, \mathcal{R}, \mathcal{CS}, \mathcal{SS}$ only to produce output that has the format of a coin. When answering the `Open Coin` query, the functionality decrypts $c$ if it is not found in the table. This is so since the CCA2-security of $\mathcal{CS}$ does not prevent $\mathcal{Z}$ from producing a valid cleartext-ciphertext pair and use it to determine whether it interacts with the functionality or the real protocol.

Since the most involved message is `Verify Coin`, we discuss it in more detail. As noted in [5], messages created by corrupted players or messages created with keys that do not originate from the protocol must be verified according to the real protocol rather than rejected. Otherwise the environment $\mathcal{Z}$ could distinguish between the ideal functionality and the real protocol by creating a new pair of signature keys and sign a root with this new key pair. The same holds for a corrupted $\mathcal{U}$ which might leak its secret to $\mathcal{Z}$ to let $\mathcal{Z}$ prepare coins internally without interacting with the protocol.

When a coin is verified, Condition 1 says that it should be considered invalid if it has not been issued by $\mathcal{B}$. Condition 2 say that if the coin is being verified

**Functionality 1 ($\mathcal{F}_{\text{EC}}^{H,\mathcal{R},\mathcal{CS},\mathcal{SS}}$).** Until $(\mathcal{B}, \texttt{KeyGen})$ is received all messages except $(\mathcal{B}, \texttt{KeyGen})$ are ignored.

- Upon reception of $(P_i, \texttt{KeyGen})$ proceed as follows:
  1. If $P_i = \mathcal{B}$, set $\mathsf{key} \leftarrow \mathsf{Kg}(\kappa_1)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kg}(\kappa_1)$, and return $(\mathcal{B}, \texttt{KeyGen}, \mathsf{pk})$.
  2. Else record $P_i$ in the member list and draw $U^i$ from the family $\mathcal{U}_{\kappa_1}$ and return $(P_i, \texttt{KeyGen})$.
- Upon reception of $(\mathcal{B}, \texttt{Issue Coin}, P_i)$, verify that $P_i$ is in the member list. If not, return $(\mathcal{B}, \texttt{Not A Member})$ and quit. Set

$$c \leftarrow E_{\mathsf{key}}(0),\ k_j \leftarrow (U^i(c\|j))_{j=1}^{\kappa_2},\ \boldsymbol{z} \leftarrow H(\boldsymbol{k}) ,$$

  where $\boldsymbol{k} = (k_1, k_2, \ldots, k_{\kappa_2})$. Add $(c, P_j, \boldsymbol{k}, \emptyset)$ to $C_t$. Hand $(\mathcal{S}, \texttt{New Coin}, P_i)$ and $(P_i, \texttt{New Coin}, c, \boldsymbol{z})$ to $\mathcal{C}_\mathcal{I}$.
- Upon reception of $(\mathcal{B}, \texttt{Tick})$, set $T \leftarrow \mathsf{buildtree}_H(\mathsf{val}_H(C_t))$ and modify each $e = (c, P_i, \boldsymbol{k}, \emptyset) \in C_t$ into $(c, P_i, \boldsymbol{k}, \mathsf{getchain}_T(\mathsf{val}_H(e)))$. Set $\sigma \leftarrow \mathsf{Sig}_{\mathsf{sk}}(\mathsf{root}(T))$ and add $\mathsf{root}(T)$ to $T_{\text{signed}}$. Return $(\mathcal{B}, \texttt{Tick}, T, \sigma)$ to $\mathcal{C}_\mathcal{I}$. Set $t \leftarrow t + 1$.
- Upon reception of $(P_i, \texttt{Prepare Coin}, c, \boldsymbol{z}, P_j)$, find $\boldsymbol{k}$ such that $(c, P_i, \boldsymbol{k}, \cdot) \in C$. If no such $\boldsymbol{k}$ exists, then hand $\mathcal{C}_\mathcal{I}$ the message $(P_i, \texttt{Reject Prepare Coin}, c)$ and quit. Otherwise set $\tilde{\boldsymbol{k}} \leftarrow \boldsymbol{k}_{\mathcal{I}(P_j)}$, return $(P_i, \texttt{Prepared Coin}, c, \tilde{\boldsymbol{k}})$ to $\mathcal{C}_\mathcal{I}$ and store $(c, P_j)$ in $T_{\text{prepared}}$.
- Upon reception of $(P_i, \texttt{Verify Coin}, c, \boldsymbol{z}, \tilde{\boldsymbol{k}}, P_j, h', \sigma, \mathsf{pk}')$, find $P_l, \boldsymbol{k}, h$ such that $(c, P_l, \boldsymbol{k}, h) \in C$. Return $(P_i, \texttt{Verify Coin}, c, P_j, \texttt{invalid})$ to $\mathcal{C}_\mathcal{I}$ if at least one of the following holds:
  1. No such entry exists.
  2. $\mathsf{pk} = \mathsf{pk}'$ and $\mathsf{root}(h) \notin T_{\text{signed}}$.
  3. $\mathsf{Vf}_{\mathsf{pk}'}(\mathsf{root}(h), \sigma) = 0$.
  4. $h' \neq h$.
  5. $P_l$ is not corrupted, and

$$(\tilde{\boldsymbol{k}} \neq \boldsymbol{k}_{\mathcal{I}(P_j)}) \vee (P_j \notin \mathsf{span}_\mathcal{I}(\{P \mid (c, P) \in T_{\text{prepared}}\})) .$$

  6. $P_l$ is corrupted and $H(\tilde{\boldsymbol{k}}) \neq \boldsymbol{z}_{\mathcal{I}(P_j)}$.

  Otherwise return $(P_i, \texttt{Verify Coin}, c, P_j, \texttt{valid})$ to $\mathcal{C}_\mathcal{I}$.
- Upon reception of $(\mathcal{B}, \texttt{Open Coin}, c)$, find a value $(c, P, \cdot, \cdot)$ in $C$. If no such entry exists, then set $P \leftarrow D_{\mathsf{sk}}(c)$. Return $(\mathcal{B}, \texttt{Open Coin}, c, P)$.
- Upon reception of $(P_l, \texttt{Check Doublespent}, c, \boldsymbol{z}, \tilde{\boldsymbol{k}}_{\boldsymbol{1}}, \tilde{\boldsymbol{k}}_{\boldsymbol{2}}, h, \sigma, P_{j_1}, P_{j_2})$ from $\mathcal{C}_\mathcal{I}$, execute $(\texttt{Verify Coin}, c, \boldsymbol{z}, \tilde{\boldsymbol{k}}_{\boldsymbol{i}}, h, \sigma, P_{j_i})$ for $i = 1, 2$.
  1. If at least one execution returns $(\texttt{Verify Coin}, c, P_{j_i}, \texttt{invalid})$, then return $(P_l, \texttt{Check Doublespent}, c, \texttt{invalid})$ to $\mathcal{C}_\mathcal{I}$.
  2. If $P_{j_1} = P_{j_2}$ then return $(P_l, \texttt{Check Doublespent}, c, \texttt{no})$, otherwise return $(P_l, \texttt{Check Doublespent}, c, \texttt{yes})$ to $\mathcal{C}_\mathcal{I}$.

**Fig. 2.** The definition of $\mathcal{F}_{\text{EC}}^{H,\mathcal{R},\mathcal{CS},\mathcal{SS}}$

with the correct key public key, then it is valid only if $\mathcal{B}$ actually signed the root, and Condition 3 ensures a correct answer when the coin is verified with a different public key. Because of the correctness of $\mathcal{SS}$, Condition 3 always holds for $\mathsf{pk} = \mathsf{pk}'$ if the root has been signed. Condition 4 says that the correct path must be given. Condition 5 says that if the coin owner is not corrupted, the coin must have been prepared for the designated receiver $P_j$. (Recall that $\mathsf{span}_I(\{P\}) = \{P\}$.) Alternatively if the coin has been prepared more than once, then $P_j$ must be in the span of the set of receivers. Condition 6 says that for a corrupt coin owner, the coin is accepted if the given preimages actually hash to the correct values.

**Anonymity.** In the ideal protocol, $c$ is an encryption of 0, and thus the coin does not contain any information about the owner. The only information that is disclosed to the merchant is to which tree the coin belongs. The amount of information this contains depends on the size of the tree. The larger the tree, i.e., the longer the interval between `Tick` messages, the smaller the amount of information released to the merchant.

**Fairness.** By fairness we mean that if a player (or coalition of players) prepares $l + 1$ coins that pass `Verify Coin` after withdrawing only $l$ coins, at least one withdrawn coin will be detected as double-spent. Since the only coins that can be successfully spent are the coins in the database $C$, and the only way to have a coin being added to the database is to engage in the withdrawal protocol, by the pigeon hole principle at least one coin has been prepared twice in this case. The implementation of the double-spending detection in the ideal functionality guarantees that the double-spender is revealed.

**Non-frameability.** A coalition of players should not be able to spend coins withdrawn by someone outside of the coalition. Since the `Prepare Coin` algorithm checks that it is called by the coin owner, this requirement is fulfilled.

**Detection of double-spenders.** A user that spends a coin at two different merchants will by construction have to produce two different sets of $k_i$'s and will always be detected by the bank.

Since double-spending at a single merchant will not be detected by the bank, it is the responsibility of the merchant to detect such actions, holding a list of the coins spent at that merchant. However, a simple modification of the scheme allows the merchant to remember only the coins spent the same day. To achieve this, we include the date in the computation of the index set. In the other words, rather than disclosing the list $\boldsymbol{k}_{\mathcal{I}(P_i)}$, the list $\boldsymbol{k}_{\mathcal{I}(P_i, date)}$ is disclosed.

**Correctness.** An e-cash scheme is correct if a coin withdrawn by an honest player always, or almost always, can be spent at an honest merchant and the merchant can deposit the coin at the bank. It is immediate from the construction that this property holds for the ideal functionality provided that the signature scheme $\mathcal{SS}$ is correct.

## 5 The Real Protocol

### 5.1 Definition of the Protocol

We give the definition of the protocol in the $\mathcal{F}_{\mathrm{SIG}}$-hybrid model. The ideal signature functionality $\mathcal{F}_{\mathrm{SIG}}$ [1, 5] accepts messages $\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify}$ to set up keys, sign a message, and verify a signature. We use the definition of $\mathcal{F}_{\mathrm{SIG}}^{\mathcal{SS}}$ given in Figure 3, slightly modified from [5] in that the functionality is parameterized by the signature scheme, and that $\mathcal{SS}$ is assumed to be correct.

**Functionality 2** ($\mathcal{F}_{\mathrm{SIG}}^{\mathcal{SS}}$ **[5]).**

- Upon reception of $(\mathcal{B}, \mathtt{KeyGen})$, set $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kg}$. Hand $(\mathcal{B}, \mathsf{pk})$ to $\mathcal{C}_{\mathcal{I}}$.
- Upon reception of $(\mathcal{B}, \mathtt{Sign}, m)$, set $\sigma \leftarrow \mathsf{Sig}_{\mathsf{sk}}(m)$, store $m$, and hand $(\mathcal{B}, \mathtt{Signature}, m, \sigma)$ to $\mathcal{C}_{\mathcal{I}}$.
- Upon reception of $(P_i, \mathtt{Verify}, m, \sigma, \mathsf{pk}')$, set $f = 0$ if $\mathcal{B}$ is uncorrupted, $m$ is not stored, and $\mathsf{pk} = \mathsf{pk}'$. Otherwise set $f = \mathsf{Vf}_{\mathsf{pk}'}(m, \sigma)$. Hand $(P_i, \mathtt{Verify}, m, f)$ to $\mathcal{C}_{\mathcal{I}}$.

**Fig. 3.** The definition of $\mathcal{F}_{\mathrm{SIG}}^{\mathcal{SS}}$

We are now ready to define the protocol $\pi_{\mathrm{EC}}^{H,\mathcal{R},\mathcal{CS}}$.

**Protocol 1** ($\pi_{\mathrm{EC}}^{H,\mathcal{R},\mathcal{CS}}$**).**

- The bank $\mathcal{B}$ acts as follows:
  - Upon reception of $(\mathtt{KeyGen})$, $\mathcal{B}$ creates and stores a symmetric key $\mathsf{key} \leftarrow \mathsf{Kg}(\kappa_1)$, requests $\mathsf{pk}$ from $\mathcal{F}_{\mathrm{SIG}}$, sets $C \leftarrow \emptyset$ and returns $(\mathtt{KeyGen}, \mathsf{pk})$.
  - Upon reception of $(\mathtt{Issue\ Coin}, P_i)$, $\mathcal{B}$ initiates the following protocol with $P_i$:
    1. $\mathcal{B}$ computes $c \leftarrow E_{\mathsf{key}}(P_i)$ and sends $(\mathtt{Withdrawal\ Request}, c)$ to $P_i$.
    2. $P_i$ sets $k_j \leftarrow R^i(c||j), \boldsymbol{z} \leftarrow H(\boldsymbol{k})$. Then it outputs $(\mathtt{New\ Coin}, c, \boldsymbol{z})$ and hands $(\mathtt{Withdrawal\ Response}, c, \boldsymbol{z})$ to $\mathcal{B}$.
    3. $\mathcal{B}$ stores $(c||z_1||z_2||\dots||z_{\kappa_2})$ in $C$.
  - Upon reception of $(\mathtt{Open\ Coin}, c)$, $\mathcal{B}$ returns $(\mathtt{Open\ Coin}, c, D_{\mathsf{key}}(c))$.
  - Upon reception of $(\mathtt{Tick})$, $\mathcal{B}$ computes a new hash tree $T$ from all stored values, i.e., sets $T = \mathsf{buildtree}_H(C)$. It acquires a signature $\sigma$ on $\mathsf{root}(T)$ from $\mathcal{F}_{\mathrm{SIG}}$, sets $C = \emptyset$, and returns $(\mathtt{Tick}, T, \sigma)$.
- A non-bank player $P_i$, i.e., $i > 0$, acts as below:
  - Upon reception of $(\mathtt{KeyGen})$, $P_i$ creates and stores a pseudo-random function $R^i \leftarrow_R \mathcal{R}_{\kappa_1}$ and returns $(\mathtt{KeyGen})$.
  - Upon reception of $(\mathtt{Prepare\ Coin}, c, \boldsymbol{z}, P_j)$, $P_i$ sets $k_l \leftarrow R^i(c||l)$ for $l = 1, \dots, \kappa_2$ and verifies that $\boldsymbol{z} = H(\boldsymbol{k})$. If this does not hold, it outputs the message $(\mathtt{Reject\ Prepare\ Coin}, c)$ and quits. Otherwise it sets $\tilde{\boldsymbol{k}} = \boldsymbol{k}_{\mathcal{I}(P_j)}$ and outputs $(\mathtt{Prepared\ Coin}, c, \tilde{\boldsymbol{k}})$.
  - Upon reception of $(\mathtt{Withdrawal\ Request})$, $P_i$ acts as described above.

– In addition to the above, any player $P_i$, including the bank, acts as follows.
  - Upon reception of $(\texttt{Verify Coin}, c, \boldsymbol{z}, \tilde{\boldsymbol{k}}, P_j, h, \sigma, \mathsf{pk})$, $P_i$ proceeds as follows:
    1. $P_i$ sends $(\texttt{Verify}, \mathsf{root}(h), \sigma, \mathsf{pk})$ to $\mathcal{F}_{\mathrm{SIG}}$. If $\mathcal{F}_{\mathrm{SIG}}$ returns 0, then $P_i$ returns $(\texttt{Verify Coin}, c, P_j, \texttt{invalid})$ and quits.
    2. $P_i$ verifies that $H(c, \boldsymbol{z}) = \mathsf{leaf}(h)$ and that $\mathsf{isvalid}_H(h) = 1$. If this is not the case, then $P_i$ returns $(\texttt{Verify Coin}, c, P_j, \texttt{invalid})$ and quits.
    3. $P_i$ verifies that $H(\tilde{\boldsymbol{k}}) = \boldsymbol{z}_{\mathcal{I}(P_j)}$ . If this is not the case, then it returns $(\texttt{Verify Coin}, c, P_j, \texttt{invalid})$ and quits.

    $P_i$ returns $(\texttt{Verify Coin}, c, P_j, \texttt{valid})$.
  - Upon reception of $(\texttt{Check Doublespent}, c, \boldsymbol{z}, \tilde{\boldsymbol{k}}_{\boldsymbol{1}}, \tilde{\boldsymbol{k}}_{\boldsymbol{2}}, h, \sigma, P_{j_1}, P_{j_2})$, $P_i$ executes $(\texttt{Verify Coin}, c, \boldsymbol{z}, \tilde{\boldsymbol{k}}_{\boldsymbol{i}}, h, \sigma, P_{j_i})$ for $i = 1, 2$.
    1. If at least one execution returns $(\texttt{Verify Coin}, c, P_{j_i}, \texttt{invalid})$, then $P_i$ returns $(\texttt{Check Doublespent}, c, \texttt{invalid})$ and quits.
    2. If $P_{j_1} = P_{j_2}$ then $P_i$ returns $(\texttt{Check Doublespent}, c, \texttt{no})$, otherwise it returns $(\texttt{Check Doublespent}, c, \texttt{yes})$.

### 5.2 On the Real Protocol

The protocol relies on the existence of an ideal signature functionality. Such a functionality can be implemented with a CMA-secure signature scheme [1, 5]. It is possible that a merchant will verify several coins from the same tree. In these cases the merchant can save time by only verifying the signature once.

The scheme relies on the roots being constructed after a certain amount of time, and therefore coins may not be immediately usable. The scheme can also be used without this delay by constructing a tree of size one for each coin issued and returning the signature to the user immediately. This increases coin size since there is a separate signature for each coin, but does not increase the amount of computation the user has to perform. This modification also eliminates linkability issues when coins with same owner are placed in the same tree.

### 5.3 Security of the Real Protocol

In the full version [19] we prove the following theorem:

**Theorem 1.** *The protocol* $\pi_{\mathrm{EC}}^{H,\mathcal{R},\mathcal{CS}}$ *securely realizes* $\mathcal{F}_{\mathrm{EC}}^{H,\mathcal{R},\mathcal{CS},\mathcal{SS}}$ *in the* $\mathcal{F}_{\mathrm{SIG}}^{\mathcal{SS}}$*-hybrid model if* $H$ *is drawn from a collection* $\mathcal{H}$ *of one-way collision-free hash functions,* $\mathcal{R}$ *is a collection of pseudo-random functions, and* $\mathcal{CS}$ *is a* CCA2*-secure encryption scheme.*

## 6 Comparison to Group Signatures

Our scheme is in some ways similar to group signatures. A coin can be viewed as a signature on the identity of the merchant. Signatures by different users are indistinguishable to merchants, but not to the bank. This corresponds to a group signatures scheme where the bank acts as group manager.

A user can only sign once for every coin she withdraws. For electronic cash this is a fundamental property, but it differs, of course, from ordinary group signatures. Also when used a group signatures scheme, there is no exculpability against the group manager. In other words the group manager can frame a group member.

Some group signature schemes offer revocation. When converting our scheme into a group-signature like scheme this can be achieved by publishing the coins issued to the revoked player. When verifying a signature, the verifier first checks the coin against the revocation list.

## 7 Conclusions

We have given a scheme for electronic cash that is based on symmetric primitives. The construction is efficient, and can be implemented on mobile phones or smart-cards without a cryptographic co-processor. We have also showed how to convert the scheme into a group signature scheme with one-time keys.

## Acknowledgments

I would like to thank Johan Håstad and Gustav Hast for discussions about the construction. I would also like to thank Douglas Wikström for suggesting improvements in the ideal functionality.

## References

1. M. Backes and D. Hofheinz. How to break and repair a universally composable signature functionality. In *Information Security Conference – ISC 2004*, volume 3225 of *Lecture Notes in Computer Science*, pages 61–74. Springer Verlag, 2004. Full version at `http://eprint.iacr.org/2003/240`.
2. S. Brands. Untraceable off-line cash in wallets with observers. In *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer Verlag, 1994.
3. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer Verlag, 2005. Full version at `http://eprint.iacr.org/2005/060`.
4. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd IEEE Symposium on Foundations of Computer Science – FOCS*. IEEE Computer Society Press, 2001. Full version at `http://eprint.iacr.org/2000/067`.
5. R. Canetti. Universally composable signature, certification, and authentication. In *17th IEEE Computer Security Foundations Workshop (CSFW)*, pages 219–235. IEEE Computer Society Press, 2004. Full version at `http://eprint.iacr.org/2003/239`.

6. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer Verlag, 1990.
7. D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer Verlag, 1991.
8. N.T. Ferguson. Single term off-line coins. In *Advances in Cryptology – EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 318–328. Springer Verlag, 1993.
9. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
10. A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures. In *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164. Springer Verlag, 1997.
11. A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Financial Cryptography'98*, volume 1465 of *Lecture Notes in Computer Science*, pages 184–197. Springer Verlag, 1998.
12. R. Merkle. Protocols for public key cryptosystems. In *1980 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1980.
13. T. Nakanishi, M. Shiota, and Y. Sugiyama. An efficient online electronic cash with unlinkable exact payments. In *Information Security Conference – ISC 2004*, volume 3225 of *Lecture Notes in Computer Science*, pages 367–378. Springer Verlag, 2004.
14. T. Nakanishi and Y. Sugiyama. Unlinkable divisible electronic cash. In *Information Security Workshop – ISW 2000*, volume 1975 of *Lecture Notes in Computer Science*, pages 121–134. Springer Verlag, 2000.
15. T. Okamoto and K. Ohta. Disposable zero-knowledge authentication and their application to untraceable electronic cash. In *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 481 – 496. Springer Verlag, 1990.
16. T. Okamoto and K. Ohta. Universal electronic cash. In *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 324–337. Springer Verlag, 1992.
17. C. Rackoff and D.R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer Verlag, 1992.
18. T. Sander and A. Ta-Shma. Auditable, anonymous electronic cash. In *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 555–572. Springer Verlag, 1999.
19. M. Trolin. A universally composable scheme for electronic cash (full versoin). Cryptology ePrint Archive, Report 2005/341, 2005. `http://eprint.iacr.org/2005/341`.
20. V. Varadharajan, K.Q. Nguyen, and Y. Mu. On the design of efficient RSA-based off-line electronic cash schemes. *Theoretical Computer Science*, 226:173–184, 1999.
21. C. Zamfir, A. Damian, I. Constandache, and V. Cristea. An efficient ecash platform for smart phones. In *E_COMM_LINE 2004*, pages 5–9, 2004. Also available at `http://linux.egov.pub.ro/~ecash/`.

# Energy-Privacy Trade-Offs in VLSI Computations

Akhilesh Tyagi

Dept. of Electrical and Computer Engineering,
Iowa State University, Ames, IA 50011
tyagi@iastate.edu

**Abstract.** VLSI circuits are open to sidechannel attacks which disclose information about its internal state to an adversary. Privacy is a design attribute to quantify the circuit's resistance and resilience to sidechannel attacks. There has been some recent work in cryptography to capture the notion of privacy in circuits. Several constructions to transform a circuit into a private circuit have also been proposed. In this paper, we quantify the energy cost of providing privacy. We use the classical area-time-energy VLSI complexity theory techniques to prove lower bounds on the energy of any VLSI computation for a given function $f$ parametrized by its privacy $P$ (Privacy $P$ or a $P$-private circuit implies that at least $P$ bits of the circuit need to be observed to derive a single bit of information about an internal node). The main result establishes a lower bound of $\Omega\left(t^2n^2\right)$ on the $E$ or $ET$ or $AT^2$ product of any $t$-private computation of an $n$-bit multiplier or shifter. Incidentally, the privacy transformation proposed by Ishai et al [6] will generate $n$-bit multiplier and shifter with matching energy, energy-time, and $AT^2$ characteristics establishing that these lower bounds are tight. The privacy of the base design, without any privacy enhancement techniques, is $t = 1$. Hence this demonstrates that the privacy comes at a quadratic multiplicative factor energy cost, which can be significant for portable, energy-starved applications such as Smart card. We further introduce the notion of information splitting secret sharing based privacy enhancement techniques. The lower bound on the energy for this case improves to $\Omega\left(Pn^2\right)$, a factor $P$ improvement.

## 1 Overview

Sidechannel attacks [9], [8] are known to be serious threats to crypto hardware. The examples of such attacks include differential power attacks, timing attacks, EM radiation measurement based attacks, and direct observation attacks. Moreover, the intellectual property protection even for the non-cryptographic hardware faces the same sidechannel information leakage issues. Inclusion of TPM [2] in most computing systems will bring forth hardware privacy issues in every day VLSI designs. Several methods have been developed to mask the system behavior with respect to its energy/power profile [4], [5]. Most recently, however, Ishai, Sahai and Wagner [6] provided the most comprehensive treatment

© Springer-Verlag Berlin Heidelberg 2005

of privacy enhancing constructions for an arbitrary VLSI computation (which is based on the prior work of Chari et al. [5] and Messerges [10].

Specifically, a general construction guarding against any type of sidechannel adversary creates multiple shares of each original bit in the logic. These shares can be made to be un-correlated in the secret-splitting scheme, which gives the additional privacy in an intuitive sense. The disclosure of any subset of these shares should reveal as little information about the original bit as possible (0 if possible). This is consistent with our notion of privacy. For instance, Messerges [10] creates a random bit/variable $r_x$ for each random bit/variable $x$. The two shares created from $x$ then are $r_x$ and $r_x \oplus x$. Note that the two shares carry enough information about $x$ so that it can be uniquely decoded at the other end. Additionally, the revelation of $r_x$ or $r_x \oplus x$ does not reveal any information about $x$ (both shares are needed to decode $x$). Hence if the adversary can only get one of the two shares, complete privacy is guaranteed. Ishai et al. [6] generalize this scheme further by creating $t+1$ shares with $t$ random shares $r_{x_1}, r_{x_2}, \ldots, r_{x_t}$ and the $t+1$st share $x \oplus r_{x_1} \oplus r_{x_2} \oplus \ldots, \oplus r_{x_t}$ (their construction uses $m = 2t+1$ shares which can be refined to $t+1$ shares). Their $t$-private circuits yield no information to the adversary for an adversary limited to up to $t$ observations per cycle. This is a very robust design style.

We demonstrate in this paper that such privacy guarantees come at a substantial energy cost (and area cost). Specifically, the energy consumption of a $t$-private VLSI computation goes up by a factor $t^2$. The energy $E$ (for a pipelined computation) or $ET$ product for a sequential implementation is bounded by $\Omega\left(P^2 I(f,n)^2\right)$ for privacy level $P$ ($P$-private circuits in [6] notation). Such results provide valuable insights about the VLSI implementation costs related to privacy issues. Most of the building blocks of cryptographic hardware, integer multiplication, shifting, DFT have information complexity $I(f,n)$ of order of $n$ bits. This indicates that such blocks will consume energy proportional to $n^2P^2$ to attain privacy $P$ as opposed to energy proportional to $n^2$ for a base implementation. For one, this implies that the energy cost of achieving privacy levels equivalent of the order of $n$ bits (even if almost all of the input bits are seen by the adversary, no sidechannel information compromising the secrecy of results would have leaked) is fairly prohibitive at $\Omega\left(n^4\right)$. Given that privacy is a larger issue in portable systems, which are more amenable to sidechannel attacks, and yet are energy limited; the energy cost of privacy enhancement becomes an important parameter. A battery operated device has finite amount of energy (per battery charge), and hence is designed to perform maximal computations within this energy budget. For such a system, a privacy level of even 10 results in an energy overhead of a factor equal to approximately 100! We use this observation to consider an alternative way to derive the privacy enhancing shares. These *information splitting* shares reduce switching per shared bit in proportion to the number of shares. In such information splitting privacy schemes, the energy cost can be shown to be proportional to $n^2P$, a $P$-fold reduction.

We first introduce the VLSI computing, energy and privacy model in Section 2. the energy model. Some basic definitions and preliminaries are established in Section 3. Section 4 contains the energy-privacy trade-off results. Section 5 introduces the information-splitting privacy model and sketches the energy bounds proofs. Section 6 concludes the paper.

## 2 Model

We introduce the VLSI model followed by the privacy model.

### 2.1 VLSI and Energy Model

**Grid and I/O Ports:** The model of VLSI computation is essentially the same as the one described by Thompson [12]. A computation is abstracted as a communication graph. A communication graph is very much like a flow graph with the primitives being some basic operators that are realizable as electrical devices. Two communicating nodes are adjacent in this graph. A layout can be viewed as a convex embedding of the communication graph in a Cartesian grid (a $\lambda$ separated grid derived from the $\lambda$ based design rules).

**Energy Model:** The following is an enhancement of this model to account for energy. It is more or less similar to the model proposed by Kissin [7] except for the multiswitch energy case. We assume that whenever a wire of length $l$ changes state, it consumes $\Theta(l)$ switching energy. This has the following justification for CMOS technology. The switching energy required to switch a node of capacitance $C$ is $CV^2/2$. The capacitance $C$ of the wire equals $\frac{\epsilon A}{d}$, where $\epsilon$ is the permittivity of the dioxide, $A = w\,l$ is the area of the wire, and $d$ is the depth of the dioxide. Then the switching energy is $CV^2/2 = \frac{V^2 \epsilon w}{2d}\, l$. In a typical layout, the widths of all the wires are within a constant factor of the minimum metal width for a process. For a given process, $d$ and $\epsilon$ are constant. The supply voltage $V$, currently is in the range .8-2 Volts and is not expected to be lowered significantly in order for a chip to remain noise-immune. Thus dioxide depth $d$, wire width $w$, permittivity $\epsilon$ and the supply voltage $V$ can all be absorbed into a proportionality constant permitting us to conclude that *the energy needed to switch a wire of length $l$ is $\Theta(l)$*. When a bit is stored on-chip for $k$ time units, we charge $k$ units of energy. The logic is broadly classified into two design styles — combinational and sequential. The design style determines the number of times a wire can switch in response to a new input, which has a strong bearing on the effectiveness of the lower bound proof techniques. For this reason, the following two cases are considered.

**Combinational/Uniswitch Model:** In a combinational circuit, each wire is limited to switch at most once in the absence of race conditions. These are acyclic circuits with the restriction that for each input instance no signal can switch more than once. The output value is determined solely by the input combination. This model was named the Uniswitch Model (USM) by Kissin [7]. In this paper, to avoid confusion, we will use the term *combinational* which covers both USM and MSM (multiswitch model) as defined by Kissin.

**Sequential/Multiswitch Model:** The other design style is *sequential* where combinational blocks are used repeatedly under the control of finite state machines. The switching of the wires in sequential circuits cannot be bounded by one. Tyagi [13] used the term Multiswitch Model (MSM) to refer to this model. However, to avoid ambiguity with Kissin's MSM model, we will refer to this model as *sequential model.* Note that the multiple switching can occur due to presence of either cycles or race conditions or both. The race conditions without cycles constitute combinational circuits. The other two cases fall in the domain of sequential circuits.

## 2.2 Privacy Model

We adopt a variant of Agrawal and Aggarwal [1] who provide an entropy based definition of privacy.

**Definition 1.** *Privacy of a single variable $x$ is defined as the entropy of $x$, $h(x)$, given by $\int_{\Omega_x} f_x(i) \log f_x(i) di$. Note that $\Omega_x$ is the domain of $x$ and $i$ is a value in $\Omega_x$. This is the classical information theoretic definition of entropy for a variable $x$ viewed as a random variable.*

If this variable $x$'s privacy were to be enhanced by applying a perturbing function $r_x$, we can capture conditional entropy of $x$ as follows.

**Definition 2.** *Conditional privacy of a single variable $x$ perturbed by a function $r_x$ is defined as the conditional entropy of $x$, $h(x|r_x)$, given by $\int_{\Omega_{x,r_x}} f_{x,r_x}(i,j)$ $\log f_{x,r_x}(i,j) di\, dj$.*

The loss of privacy for $x$ resulting from the exposure of $r_x$ is the key definition of privacy developed in Agrawal and Aggarwal [1].

**Definition 3.** *The privacy loss for variable $x$ resulting from the exposure of a perturbing variable $r_x$ is defined as $1 - 2^{h(x|r_x)}/2^{h(x)}$.*

Note that if $r_x$ is a random variable chosen independently from $x$ (as is the case in [10] and [6]), the privacy loss is 0 since $h(x|r_x) = h(x)$. Now we can define the notion of *privacy* as used in Ishai et al. [6].

**Definition 4.** *A variable $x$ is designed to be $t$-private if it is perturbed by at least $k \geq t$ variables $r_{x_1}, r_{x_2}, \ldots, r_{x_k}$ and the privacy loss for $x$ resulting from the exposure of any subset of up to $t$ perturbing variables is 0.*

Note that this definition of privacy insists on maintaining 0 correlation between the protected variable $x$ and any subset of its perturbing variables. In these schemes, $x$ is represented by at least $t + 1$ physical variables, also known as its *shares* $xs_0, xs_1, \ldots, xs_t$. In other words, almost all the shares of $x$ carry 0 information about $x$ in these schemes. We call such privacy schemes *information isolating* schemes or *information isolating* shares. Later in the paper (Section 5), we will introduce a variant called *information-splitting* shares or privacy schemes.

Once again, to recap Messerges [10] and Ishai et al. [6] schemes, Messerges splits each variable $x$ into two shares $r_x$ (a random bit) and $r_x \oplus x$. He calls this scheme a masking scheme. He also introduces a similar arithmetic masking variant. Ishai et al. generalize this scheme to split $x$ into $t+1$ shares $xs_0 = r_{x_1}, xs_1 = r_{x_2}, \ldots, xs_{t-1} = r_{x_t}, xs_t = r_{x_1} \oplus r_{x_2} \oplus \cdots \oplus r_{x_t} \oplus x$. They then provide a transformation for the Boolean basis of a "not" gate and an "and" gate where each operand is a $t+1$ bit value.

**Privacy Adversary:** ¿From privacy perspective, our adversary is similar to the one assumed in Ishai, Sahai, Wagner [6]. It is a $t$-limited, interactive adversary. The adversary can choose the inputs to the circuit to be asserted for each round, and can choose an arbitrary set of up to $t$ internal nodes to observe/sample. The adversary is allowed to choose an input and $t$ internal nodes in each round $i$ on the basis of its observations in the preceding rounds. In order to accommodate power attacks, the adversary can also choose to observe the power nodes, *i.e.*, some of these $t$ observations are power nodes. For a power node, the adversary is not interested in a Boolean value. It observes the current profile of that node for a certain duration $\Delta t$ in that round. If the power node is internal, such an observation is feasible through EM radiation measurement. For an external power pin, the current can be measured directly.

## 3 Preliminaries

In this section, we briefly review the most relevant concepts from VLSI complexity theory – information or communication complexity of a function. We also state and prove a lemma to bound the switching incurred in transmitting $k$ bits of information. Finally, a key cutting lemma to separate a chip into two parts is presented. In the following, log and ln refer to the base 2 and natural logarithms respectively and $e$ is the inverse of natural logarithm.

### 3.1 Information

Many proof techniques in VLSI complexity theory argue about the amount of information exchanged between two disjoint regions on a chip. The information content is measured in terms of number of bits, which is also consistent with the information-theoretic notion of information based on the entropy function. Hence the number of bits required to transmit $k$ distinct events with probabilities $p_1, p_2, \ldots, p_k$ is given by $\lceil H(P) \rceil$ where $H(P) = \sum_{i=1}^{k} p_i \log \left( \frac{1}{p_i} \right)$.

In most of the proofs in this paper, we will distinguish between **physical bits** and **information bits**. When two disjoint regions on a chip need to exchange $k$ bits of information; we are referring to the information-theoretic definition of *bits*. We will use **information bit** to refer to the information content of data. On the other hand, the term bit is also used to refer to the concept of *binary digits*; which is the number of physical bits in the data. Note that in order to communicate $k$ bits of information (or $k$ information bits), at least $k$ physical bits have to be used. We can, of course, always use $k' > k$ physical bits to encode $2^k$ distinct events ($k$ information bits), however the most efficient communication occurs when the number of information bits is the same as the number of physical bits. In this paper, we use the term *bits* to default to *physical bits*.

Note that we assume, without loss of generality, that the information is *signaled* by *switching* a wire. It may seem, at first glance, that synchronous circuits can get around this assumption by transmitting information through the signal level and clock cycle count, *i.e.*, if a wire retains signal level 1 from clock cycle 0 through clock cycle $k$ then the event $k$ is communicated. Note, however, that in such a scenario, the clock wire is providing implicit communication by switching twice during each clock period. Hence the basic mode of signaling is still switching. The following lemma formalizes this argument for combinational circuits.

**Lemma 1.** *Let $k$ combinational wires, which switch at most once, be the only wires connecting two disjoint regions $R_1$ and $R_2$ in a VLSI circuit. The amount of information that can be exchanged between $R_1$ and $R_2$ is at most $k$ information bits.*

*Proof.* Note that each wire can switch at most once and hence it carries a binary event. The entropy function $H(p)$ for a binary probability distribution has a maximum value 1. This argument holds assuming that the single switch of the wire is not decoded along with some timing information in order to generate more information bits. For instance, the time slot when the wire $w$ switches conveys $T$ events for a circuit operating for $T$ time slots resulting in $\log T$ information bits. In this scenario, $k$ wires can carry up to $k \log T$ information bits. How can such timing information be provided? The clock wires are precluded in combinational model since they need to switch more than once. The other option consists of two regions $R_1$ and $R_2$ maintaining their own clocks that are consistent. But this requires $T$ synchronization events in our VLSI model as we discussed in Section 2. In combinational model, each synchronization event takes one wire with one switch. Hence the transmission of $k \log T$ information with explicit time information requires $k + T$ combinational wires, which is even less efficient. $\square$

Now we introduce the notion of *information complexity* of a function $f(x_n x_{n-1} \ldots x_1) = y_m y_{m-1} \ldots y_1$. Let $C_f = (V, W, \mathcal{I}, \mathcal{O})$ be a circuit to compute $f$. Consider a partition $\pi = \{\pi^L = \{x_{i_1}, x_{i_2}, \ldots x_{i_{\lceil n/2 \rceil}}\}, \pi^R = \{x_{j_1}, x_{j_2}, \ldots x_{j_{\lfloor n/2 \rfloor}}\}\}$ that divides the set of input bits into two equal-sized sets. The chip $C_f$ can possibly be partitioned in such a way that one partition contains the input ports corresponding to $\pi^L$ and the other one contains the input ports for $\pi^R$. Let $I(f, C_f, \pi, \boldsymbol{x}, n)$ be the number of bits that need to be exchanged between $\pi^L$ and $\pi^R$ when the input bits are assigned values according to $\boldsymbol{x} \in \{0,1\}^n$. Note that $I(f, C_f, \pi, \boldsymbol{x}, n)$ is $\infty$ for all the partitions $\pi$ that cannot be realized in the chip $C_f$. The information complexity of $f$, $I(f, n)$, is the minimum number of bits exchanged between any two almost equal-sized partitions of the input bits over all implementations, which is $\min_{C_f} \min_{\pi \in \Pi_n} \max_{\boldsymbol{x}} I(f, C_f, \pi, \boldsymbol{x}, n)$. $\Pi_n$ is the set of all approximately equal sized partitions of $n$ bits such that each set in the partition contains at least $n/c$ bits for a constant $c > 1$. We will use $I$ to denote $I(f,n)$ in the following, whenever the function $f$ is implicit. Many techniques were developed to derive lower bounds on $I(f,n)$ for specific functions [[14], [12], [16]]. A particularly interesting class of functions: *transitive functions*, was introduced by Vuillemin [15]. A transitive function embeds a transitive permutation group computation. Some examples of transitive functions include shifting, integer multiplication and linear transforms. Vuillemin also showed that $I(f,n)$ for a transitive function is $\Omega(n)$.

## 3.2 Switching Lemma

We wish to show that when $k$ information bits are encoded with $k' > k$ physical bits and transmitted across $w$ wires; they induce a significant amount of switching. The main difficulty in proving lower bounds on energy consumption arises from the observation that expanding information can reduce switching and hence reduce energy. An extreme scenario expands the information exponentially. For instance, a chip could transmit $2^k$-bit long strings to convey $k$ information bits. Only the strings from $\{0^*10^*\} \cap \{0,1\}^{2^k}$ (strings of length $2^k$ with only one 1) are used to transmit the information. The string $0^{i-1}10^{2^k-i}$ conveys the $i$th event. Note that the switching in such a string is limited to only two, at the borders of the bit 1. The switching lemma bounds the average switching by $\Omega\left(k/\log(k'/k)\right)$. Let us first introduce the notion of *alternation*.

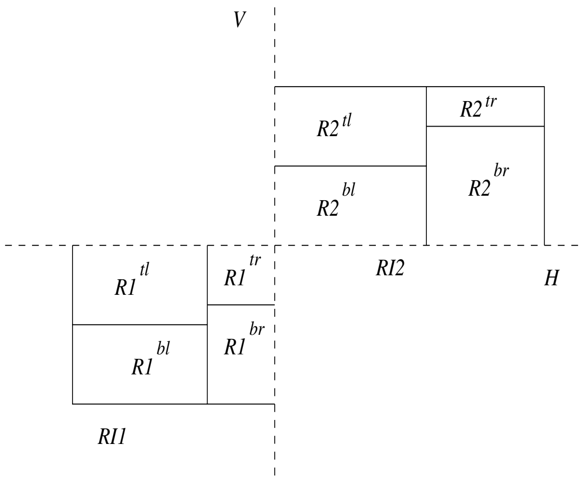


**Fig. 1.** An Illustration of the Cutting Method

We say that in a bit sequence $a_1, a_2, \ldots, a_l$, there is an *alternation* at position $j$ if $a_j \neq a_{j+1}$ for $1 \leq j \leq l-1$. Let $\delta_j$ be 1 if $a_j \neq a_{j+1}$ and 0 otherwise. Then the *total alternation* for a $l$-bit sequence is given by $\sum_{j=1}^{l-1} \delta_j$.

**Lemma 2.** *Let $k'$ and $k$ be two positive integers such that $k' \geq k$. The average number of alternations in transmitting a $k'$ bit encoding of $k$ information bits, $A(k',k) \geq k/4\log(4k'/k)$.*

Note that if a $k'$ bit sequence is sent on a single wire then the average switching is identical to the average alternation of the bit sequence. Lemma 2 proves that a $k' > k$ bit encoding of $k$ information bits ($2^k$ distinct events) has average alternation at least $k/4 \log(4k'/k)$. Now let us consider the case when the $k'$-bit encodings are sent on $l > 1$ wires.

**Lemma 3.** *Let $k$ information bits be encoded with $k' > k$ physical bits. The average switching in transmitting $k'$ bits on $l > 0$ wires is at least $k/4 \log(4k'/k)$.*

## 3.3 Cutting Lemma

The basic technique to derive VLSI lower bounds (introduced by Thompson [12]) first cuts the chip into parts such that each part has approximately half the input bits. For area-time lower bounds, a simple variant of a line cutting the rectangular chip is sufficient. For energy lower bounds, a cut needs to satisfy several additional conditions, as will become evident later.

We first prove a key cutting lemma. Note that we have not used the floor function to denote the integer value of a fraction $\left\lfloor \frac{x}{y} \right\rfloor$ in the following lemma in order to avoid the clutter of notation. The use of floor function would not have changed the results.

**Lemma 4.** *Let $C = (V, W, \mathcal{I}, \mathcal{O})$ be a circuit to compute $f$. Let us assume that no input port reads more then $n/81$ input bits. There exist two sets of input ports $I_1 \subset \mathcal{I}$ and $I_2 \subset \mathcal{I}$ such that $I_1$ and $I_2$ each read at least $n/81$ input bits. Let $R_{I_1}$ and $R_{I_2}$ denote the smallest rectangles containing $I_1$ and $I_2$ respectively. Let $p_1$ ($p_2$) be the perimeter of $R_{I_1}$ ($R_{I_2}$). Let $h$ ($v$) be the horizontal (vertical) distance between $R_{I_1}$ and $R_{I_2}$. Then at least one of the following statements holds true.*

1. $h \geq \min(p_1/4, p_2/4)$.
2. $v \geq \min(p_1/4, p_2/4)$.

*Proof.* Since no input port reads more than $n/81$ input bits, we can find a vertical line $V$ bisecting the chip into two halves such that each half reads at least $n/3$ input bits. The argument for it is similar to that in Ullman [[14] , page 49], which states that moving an input port from one side to the other can move at most $n/3$ bits. Note that such a line need not be a straight line. It could be a line with one jog as shown in Thompson's cutting argument [12]. Let us concentrate on the right-hand side of $V$. Once again, we can find a horizontal line $H$ that cuts the right-hand side of $V$ into two parts such that each part reads at least $n/9$ input bits. The left-hand side of $V$ is also cut into two parts by the line $H$. At least one of these parts reads $n/6$ input bits. Without loss of generality, let the bottom part read greater than or equal to $n/6$ input bits. Let $R_{I_1}$ and $R_{I_2}$ be the smallest rectangles containing the input ports in the bottom-left and top-right quadrants respectively, as shown in Figure 1.

Now cut both $R_{I_1}$ and $R_{I_2}$ first vertically and then horizontally as before. This gives rise to four rectangles $R_1^{tl}$, $R_1^{tr}$, $R_1^{bl}$ and $R_1^{br}$ each reading at least

$n/54$ input bits. We also get $R_2^{tl}$, $R_2^{tr}$, $R_2^{bl}$ and $R_2^{br}$ each reading at least $n/81$ input bits. Let $R_1^{tl}$, $R_1^{tr}$, $R_1^{bl}$, $R_1^{br}$, $R_2^{tl}$, $R_2^{tr}$, $R_2^{bl}$ and $R_2^{br}$ have perimeter $p_1^{tl}$, $p_1^{tr}$, $p_1^{bl}$, $p_1^{br}$, $p_2^{tl}$, $p_2^{tr}$, $p_2^{bl}$ and $p_2^{br}$ respectively. Without loss of generality, let a rectangle in $R_{I_1}$ have the smallest perimeter. Let this rectangle be $R_1^{tr}$ with the smallest perimeter $p = p_1^{tr}$. Since we will be separating this rectangle from one of the rectangles in $R_{I_2}$, any other rectangle in $R_{I_1}$ can only have a larger separation. Consider the rectangle $R_2^{bl}$. It has perimeter $p_2^{bl} \geq p$. Then either its width $w \geq p/4$ or its height $h \geq p/4$. If the width of $R_2^{bl}$ is greater than or equal to $p/4$ then $R_2^{br}$ is horizontally separated from $R_1^{tr}$ by distance at least $p/4$ where the perimeter of $R_1^{tr}$ is $p$. Otherwise $R_2^{tl}$ is vertically separated from $R_1^{tr}$ by distance at least $p/4$. All the other pairs of rectangles have an even larger separation. This proves the assertion. □

## 4 Energy Privacy Trade-Off

In this section, we show that the combinational energy and combinational & sequential $ET$ product have a lower bound of $\Omega\left(t^2 I^2\right)$ for a $t$-private implementation, where $I$ is the information complexity of the function, $I(f,n)$. The sequential energy can be shown to be $\Omega\left(t^{3/2} I^{3/2}\right)$. In particular, these results apply to transitive functions such as shifting, multiplication and DFT. Since the information complexity of transitive functions is known to be $\Omega(n)$, these lower bounds translate into: combinational and sequential $ET$ product, $\Omega(t^2 n^2)$; combinational $E$, $\Omega(t^2 n^2)$; sequential $E$, $\Omega(t^{3/2} n^{3/2})$. Let us consider the worst case switching energy first.

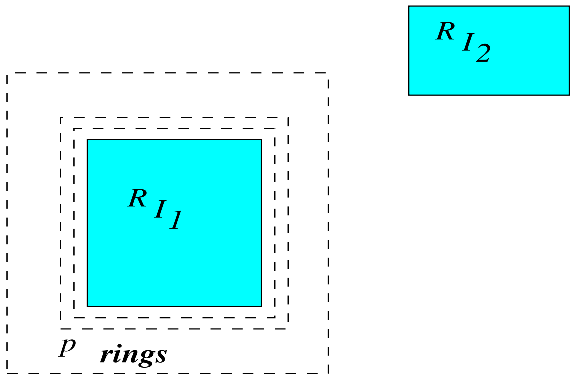


**Fig. 2.** The Rings Used in $ET$ Lower Bound

### 4.1 Worst Case Switching Energy

The strategy for the proof is as follows. We cut the given chip according to the procedure outlined in the cutting lemma (Lemma 4) so that there are two widely separated sections of the chip containing a high number of input bits ($n/c$ for a constant $c > 1$) each. By definition, the information complexity $I$ is the minimum number of information bits that must be exchanged between two regions containing at least $n/c$ input bits each for a constant $c > 1$. Due to privacy enhancements, in a $t$-private circuit, these sections exchange at least $tI$ bits. A cut is a bounding box around these input bit regions. The switching lemma from Section 3 provides a lower bound on the switching energy consumed at such a cut. This energy is summed over several such disjoint cuts leading to a lower bound. Our lower bounds along with the lower bounds on $I$ result in energy lower bounds parametrized by $n$ and $t$. We start with the uniswitch/pipelined energy case.

**Combinational Energy**

Recall that no wire in a combinational circuit can switch more than once. Hence no input port can read more than one input bit. Given that all the $2^n$ input values are uniformly distributed, any input bit has probability 1/2 of differing from another input bit. If an input port did read two or more input bits, it will switch more than once at least for some input assignments, once between initial state and the first input bit, and then between the first and second input bits. This implies that a combinational circuit $C = (V, W, \mathcal{I}, \mathcal{O})$ must contain $n$ input ports.

**Theorem 1.** *A combinational $t$-private VLSI computation of a function $f$ with information complexity $I$ has an energy-time product $ET = \Omega(t^2I^2)$. The combinational energy consumption is also $\Omega(t^2I^2)$. Hence the ET product and the switching energy of $t$-private transitive function implementations are $\Omega(t^2n^2)$.*

*Proof.* Note that the proof of Lemma 1 also implies that in combinational circuits, every input port reads at most one input bit. Thus we can use Lemma 4 to derive two rectangles $R_{I_1}$ and $R_{I_2}$ satisfying the following conditions. $R_{I_1}$ and $R_{I_2}$ each contain at least $tn/81$ input ports (or bits derived from the input decoder of [6]). Their horizontal or vertical separation is greater than or equal to $\min(p_1/4, p_2/4)$ where $p_1$ and $p_2$ are their respective perimeters. Without loss of generality, let this separation be $p_1/4$. We build $p_1/4$ new bisections in the following way. Let $W(0)$ be the set of all the horizontal or vertical unit length wire segments crossing the perimeter of $R_{I_1}$. Let $W(i)$ be defined recursively as the set of unit length wires adjacent to the wires in $W(i-1)$ and not in $W(0), \ldots, W(i-1)$ and not inside $R_{I_1}$. These are the rings around $R_{I_1}$ as shown in Figure 2. Note that if the chip boundaries do not allow some of these rings to expand along some directions then only the wires in the expansion towards $R_{I_2}$ constitute the bisection 'rings'. Let us consider the rings $W(0), W(1), \ldots W((p_1/4) - 1)$. Note that $W(0)$ contains at most $p_1 + 4$ unit length wire segments that link it to $W(1)$. It can be seen that the perimeter of $W(i)$ for $1 \leq i \leq (p_1/4 - 1)$ is $p_1 + 8i$ and at most $p_1 + 8i + 4$ unit length wire segments are contained in the bisection of $W(i)$ and $W(i + 1)$. Hence the number of unit length wire segments connecting $W(i)$ to $W(i+1)$ for $0 \leq i \leq (p_1/4 - 1)$ has an upper bound of $3p_1$. The combinational model dictates the uniswitch requirement which allows each wire to switch at most once. Lemma 1 shows that each wire in a combinational circuit can carry at most one bit of information during the course of computation. Note that the race conditions induced switching, omitted from the combinational model, does not carry any extra information. Hence, the total number of bits transmitted across a ring, $m$, cannot exceed $3p_1$. If the majority of the output bits are generated inside a ring $W(i)$, then the information about the input bits read in $R_{I_2}$ needs to come into the ring. Otherwise, the information about the input bits read inside the ring needs to go out. In either case, since $R_{I_1}$ and $R_{I_2}$ initially contain information on at least $n/81$ input ports, the number of information bits traveling across these rings is $\Omega(tI)$ due to $t$-privacy. Let $m_{W_i}$ physical bits cross $W(i)$. Note that since the maximum number of physical bits that can be transmitted across a ring is $3p_1$, $m_{W_i} \leq 3p_1$, $\forall i$. Hence by Lemma 3 the switching at a ring $W(i)$ is at least $\frac{tI}{4\log(4m_{W_i}/tI)} \geq \frac{tI}{4\log(12p_1/tI)}$. The length of each wire segment in a ring $W(i)$ is one and hence the energy consumption at a ring equals the switching count. Summed up over $p_1/4$ rings, the energy $E$ is $\frac{tIp_1}{16\log(12p_1/tI)}$. Note that for the correctness of the computation the total number of physical bits crossing a ring must at least equal the number of privacy enhanced information bits $tI$, *i.e*, $3p_1 \geq m_{W_i} \geq tI$. Also observe that for $p_1 > tI/3$, $\frac{p_1}{16\log(12p_1/tI)} > tI/96$. Thus the combinational energy is $\Omega(t^2I^2)$. Since the time taken is at least one time unit: $T \geq 1$, $ET$ product is also $\Omega(t^2I^2)$. □

## Sequential Energy

In a sequential circuit, the principal difficulties in proving a good lower bound are as follows. An input port can be multiplexed to read many input bits. A wire can be used to multiplex many bits, unlike the combinational/uniswitch case where a wire carries at most one bit. In the extreme, a wire may transmit a bit for each time unit. Or to make things even more complicated, a hand-shaking protocol might induce a few periods of activity on an otherwise idle wire. These conditions in conjunction with information expansion make it harder to get a handle on sequential energy consumption.

Now we are ready to prove a lower bound on the sequential $ET$ product, which uses an argument similar to the one used in Theorem 1.

**Theorem 2.** *A sequential $t$-private VLSI implementation of a function $f$ with information complexity $I$ has an energy-time product $ET = \Omega(t^2I^2)$. Hence the $ET$ product of a $t$-private implementation of a transitive function is $\Omega(t^2n^2)$.*

Next, we derive a $\Omega\left(t^{3/2}n^{3/2}\right)$ lower bound on the sequential energy of $t$-private implementations of transitive functions. The key intuition to support an $tn\sqrt{tn}$ lower bound on sequential energy is the relationship between the perimeter and area of a two-dimensional region. A region with perimeter $p$ can have area at most $p^2$. An encoding of $tn$ information bits into $m > tn$ physical bits creates a bottle-neck at the perimeter.

We use a lemma based on Baudet's ideas [3] to establish that even if an input port reads many input bits, these bits would have to be remembered before an output bit can be asserted for a transitive function. Thus before any output bit can be generated, the information about almost all the $n$ input bits should either be stored on the chip or should be read at that time. Hence, at least $\Omega(tn)$ storing nodes are needed for the computation of a transitive function.

We will use the cutting lemma, Lemma 4, in proving the sequential energy lower bound of $\Omega(t^{3/2}n^{3/2})$. However, Lemma 4 guarantees the two well separated rectangles only when no input port reads more than $n/81$ input bits. The $\Omega\left(t^{3/2}n^{3/2}\right)$ sequential energy can *either* be attributed to the storage cost of the input bits for the duration the computation needs to retain them *or* it can be attributed to the routing energy of information. We prove both the scenarios in the following theorem to illustrate the role of both storage and routing energy.

**Theorem 3.** *The sequential energy of a $t$-private implementation of a transitive function is $\Omega(t^{3/2}n^{3/2})$.*

## 5 Information-Splitting Privacy Schemes

The redundant information resulting from privacy enhancement always results in increased switching energy consumption. If one were to encode the $n$ information bits with $m > n$ redundant bits (as is done in many of the information-theoretic codes), the average switching per wire to transmit the $m$ bits can be bounded (from below) by $n/\log(2m/n)$. The wire lengths (and in turn the switched capacitances), however, increase in proportion to $m$ in this scenario. Hence the resulting switching energy scales as $mn/\log(2m/n)$ which always exceeds $\Omega(n^2)$ given by the non-redundant encoding with $m = n$.

Unlike the information-theoretic coding techniques that strive towards redundant codes either for fault-tolerance (error correction) or for reduced switching, the secret splitting introduces $m$ shares that all switch with high activity factor. A random bit switches with probability .5 and contains full information (entropy) with $P(x = 1) = .5$ and $P(x = 0) = .5$. Hence, in the privacy transformations, we end up paying for the increased wire capacitance without deriving the benefits of the reduced switching! This is the motivation behind information-splitting privacy schemes.

**Definition 5.** *A variable $x$ is designed to be $t$-private if it is perturbed by at least $k \geq t$ variables $r_{x_1}, r_{x_2}, \ldots, r_{x_k}$ and the privacy loss for $x$ resulting from the exposure of any one of the $t$ perturbing variables does not exceed $h(x)/t$. In other words, the revealed/leaked information is at most linear in the number of exposed bits.*

In other words, the original information in $x$ is split into $t$ equal buckets of information $h(x)/t$ each. Moreover, the leaked information is linearly proportional to the number of exposed bits. This can be achieved in many ways. Qualifying variables that split the cube set of $x$ [11] into a $t$-partition will have such an information-splitting property. Admittedly, this model is less robust than the information-isolating privacy schemes. However, the energy savings are significant (a multiplicative factor $t$) as indicated by the following theorem.

**Theorem 4.** *A combinational t-private information-splitting VLSI computation of a function f with information complexity I has an energy-time product* $ET = \Omega(tI^2)$*. The combinational energy consumption is also* $\Omega(tI^2)$*. Hence the ET product and the switching energy of t-private information-splitting transitive function implementations are* $\Omega(tn^2)$*.*

Proof omitted.
Note that we are using a more information-theoretic notion of privacy as opposed to the one based on the simulation of the adversary's view as in Ishai et al. [6]. Also observe that a statistical privacy variant of information-splitting can also be obtained. In fact, an information-splitting $t$-private circuit itself can be shown to provide $\sqrt{t}$ level of statistical privacy. In the following, we motivate a logic synthesis scenario for introducing information-splitting privacy in an adder.

Consider the generate-propagate paradigm for addition of two $n$-bit numbers $X = x_n x_{n-1} \ldots x_1$ and $Y = y_n y_{n-1} \ldots y_1$. Let generate signal $g_i = x_i.y_i$ denoting the conditions under which carry is generated at the $i$th bit position. Similarly, let $p_i$ denote the carry propagation condition given by $x_i \oplus y_i$. Then the carry computation is given by $c_i = g_i + p_i.c_{i-1}$. This schema already points to many natural shares of $c_i$. Specifically, two shares are given by $g_i$ and $p_i.c_{i-1}$. In fact, up to $i$ shares of $c_i$ can be extracted by the following expression: $c_i = g_i + p_i.g_{i-1} + p_i.p_{i-1}.g_{i-2} + \ldots + p_i.p_{i-1}.\ldots.p_2.g_1 + p_i.p_{i-1}.\ldots.p_2.p_1.c_{in}$. Note that these shares are additive with *or* rather than *ex-or*. One can develop logic where the shares are mutually exclusive. Also note that not all these shares carry equal amount of information (same switching probability). This example is only for illustration purposes. In a logic synthesis system (for random logic), as much balance in information between shares as possible can also be incorporated as a goal. We are currently working on modifications to SIS [11], a multi-level logic synthesis system, to incorporate such privacy.

## 6 Conclusions

The sidechannel attacks on VLSI implementations are becoming commonplace. Many privacy schemes to tolerate an observation aperture of up to $t$ variables (called a $t$-private circuit) have been proposed. In this paper, we demonstrate that the switching energy cost of any $t$-private implementation of a function such as $n$-bit integer multiplication, $n$-bit shifter, or $n$-point DFT is at least a constant multiple of $t^2n^2$. These bounds are tight in as much as implementations matching these bounds exist. This is a $t^2$ fold increase in switching energy just to support $t$-privacy. We then propose an alternate schema for privacy enhancement called information-splitting privacy. The energy costs of such schemas can be shown to be only $tn^2$, or $t$ times more energy efficient than the original information-isolating schemas. This is the first work of its kind incorporating the newly emerging security and privacy attributes into VLSI complexity theory.

## References

1. Dakshi Agrawal and Charu C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Symposium on Principles of Database Systems*, 2001.
2. Trusted Computing Platform Alliance. Trusted platform module, 2003. http://www.trustedcomputing.org/.
3. G. M. Baudet. On the Area Required by VLSI Circuits. In *Proceedings of CMU Conference on VLSI – VLSI Systems and Computations*, pages 100–107. CMU, Computer Science Press, 1981.
4. Johannes Blomer, Jorge Merchan, and Volker Krummel. Provably secure masking of aes. In *Proceedings of Workshop on Selected Areas in Cryptography (SAC 2004)*, 2004.
5. S. Chari, C. Jutla, J. R. Rao, , and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *CRYPTO '99, LNCS 1666*, pages 398–412. Springer-Verlag, 1999.
6. Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In *Proceedings of CRYPTO 2003*, 2003.
7. G. Kissin. Measuring Energy Consumption in VLSI Circuits: a Foundation. In *Proceedings of ACM Symposium on Theory of Computing*, pages 99–104. ACM-SIGACT, 1982.
8. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. *Lecture Notes in Computer Science, Crypto '99*, 1666:388–397, 1999.
9. Paul C. Kocher. Timing attacks on implementations of die-hellman, rsa, dss, and other systems. In *In N. Koblitz, editor, Advances in Cryptology, CRYPTO '96*, volume 1109, pages 104–113. Lecture Notes in Computer Science, SpringerVerlag, 1996.
10. Thomas S. Messerges. Securing the aes finalists against power analysis attacks. In *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, Lecture Notes in Computer Science, pages 150–164. Springer, 2000.
11. E. M. Sentovich, K. J. Singh, L. Lavango, C. Moon, R. Muragi, A. Saldhana, H. Savoj, P. Stephen, R. Brayton, and A. Sangiovanni-Vincentelli. SIS: A System for Sequential Circuit Synthesis. Technical Report Memorandum Number UCB/ERL M92/41, Electronics Research Laboratory, Dept. of EECS, University of California, Berkeley, 1992.
12. C.D. Thompson. Area-Time Complexity for VLSI. In *Proceedings of ACM Symposium on Theory of Computing*, pages 81–88. ACM-SIGACT, 1979.
13. A. Tyagi. Energy-Time Trade-Offs in VLSI Computations. In *Proceedings of the Ninth Conference on Foundations of Software Technology & Theoretical Computer Science*, pages 301–311. Lecture Notes in Computer Science #405, Springer-Verlag, 1989. submitted to IEEE TC.
14. J.D. Ullman. *Computational Aspects of VLSI.* Computer Science Press, Rockville, Md., 1984.
15. J. Vuillemin. A Combinatorial Limit to the Computing Power of VLSI Circuits. *IEEE Transactions on Computers*, C-32:294–300, March 1983.
16. A.C. Yao. Some Complexity Questions Related to Distributed Computing. In *Proceedings of ACM Symposium on Theory of Computing*, pages 209–213. ACM-SIGACT, 1979.

# Modified Serial Multipliers for Type-IV Gaussian Normal Bases*

Chang Han Kim[1], Yongtae Kim[2], Nam Su Chang[3], and IlWhan Park[4]

[1] Dept. of Information and Security, Semyung Univ., Jecheon, Korea
chkim@semyung.ac.kr
[2] Dept. of Mathematics Education,
Gwangju National Univ. of Education, Gwangju, Korea
ytkim@gnue.ac.kr
[3] Center for Information Security Technologies(CIST),
Korea Univ., Seoul, Korea
ns-chang@cist.korea.ac.kr
[4] National Security Research Istitute(NSRI), Daejeon, Korea
ilhpark@etri.re.kr

**Abstract.** The curves recommended by NIST are defined over finite fields $GF(2^m)$ with $m = 163, 233, 283, 409, 571$. Among them $GF(2^{163})$ and $GF(2^{409})$ have type-IV Gaussian normal bases. Using the Reyhani-Masoleh and Hasan's serial multiplier for type-I optimal normal basis, in this paper, we propose a new serial multiplier for $GF(2^m)$ with type-$IV$ Gaussian normal basis, which reduces the critical XOR path delay of the best known Reyhani-Masoleh and Hasan's serial multiplier by 25% and the number of XOR gates of Kwon et al.'s multiplier by 2. Therefore our proposed multiplier can be applicable to implementing the protocols related to the area including ECC under in ubiquitous computing.

**Keywords:** Finite fields, Massey-Omura multiplier, serial multiplier, Gaussian Normal Basis, ECC.

## 1 Introduction

Finite fields are important to cryptography and coding theory and especially to public key cryptography such as in ECC, XTR and ElGamal type cryptosystems, thus many researchers devote their attentions to efficient finite field arithmetic [10,12]. Finite field arithmetic depends on the basis representation and an element of the finite field is usually represented with respect to polynomial basis [6,8,19], normal basis [1,4,9,13,14,15,16,17,18] and the nonconventional basis [7] sometimes. In hardware implementation, the merit of the normal basis representation is that the result of squaring an element is simply the right cyclic shift of its coordinates. In particular, the type-I optimal normal basis generated

* This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment).

© Springer-Verlag Berlin Heidelberg 2005

by an irreducible All One Polynomial is the best known efficient among the normal bases implementation [4,7,15,16]. Massey-Omura [11] invented the serial multiplier which has a long path delay with parallel input and serial output. Agnew et al. [1] proposed a Sequential Multiplier with Parallel Output (SMPO) by improving Massey-Omura's serial multiplier. Recently, Reyhani-Masoleh and Hasan [13,17] proposed a SMPO with the lower area complexity and a little higher path delay than that of Agnew et al. In 2004, Kwon et al. [9] proposed a SMPO improving that of Agnew et al. whose path delay is unchanged and the area complexity is equal to or higher than that of Reyhani-Masoleh and Hasan according to type-II or otherwise. On the other hand, Yang et al. [20] proposed a SMPO which has the same path delay as that of Kwon et al. by reconstructing the multiplication matrix of Reyhani-Masoleh and Hasan over type-II optimal normal basis. Although the arithmetic operations in finite fields are the most efficient in case they have type-I optimal normal bases, two curves recommended by NIST are defined over the finite fields with type-IV Gaussian normal bases. Finite fields $GF(2^m)$ with type-IV Gaussian normal bases for odd $m$ can be embedded in finite fields with type-I optimal normal bases[see Lemma 1]. In this regard, we, in this paper, propose a new architecture for SMPO which transforms the Gaussian normal basis multiplication in $GF(2^m)$ into the type-I optimal normal basis multiplication in $GF(2^{4m})$ based on Reyhani-Masoleh and Hasan's SMPO over $GF(2^m)$ having a type-IV Gaussian normal basis. Our SMPO reduces the critical XOR path delay of the serial multiplier of Reyhani-Masoleh and Hasan by 25% and has the same critical path delay as that of Kwon et al. The number of XOR gates of our proposed serial multiplier is the same as that of Reyhani-Masoleh and Hasan and is fewer than that of Kwon et al. by 2.

## 2 Type-k Gaussian Normal Bases for $GF(2^m)$

In this section, we give some preliminaries for the normal basis representation of the finite field element and introduce a normal basis of low complexity. It is well known that there is always a normal basis for the finite field $GF(2^l)$ over $GF(2)$ for any positive integer $l$ [10,12]. If there exists an element $\beta$ of $GF(2^l)$ such that the set $N = \{\beta, \beta^2, \cdots, \beta^{2^{l-1}}\}$ is a basis for $GF(2^l)$ over $GF(2)$, then $N$ is called the normal basis for $GF(2^l)$ over $GF(2)$ and $\beta$ is called the normal element. Then any $A \in GF(2^l)$ can be represented as follows.

$$A = \sum_{i=0}^{l-1} a_i \beta^{2^i}, a_i \in GF(2).$$

For brevity, the normal basis representation of $A$ will be denoted by

$$A = (a_0, a_1, \cdots, a_{l-1}).$$

Also the matrix representation of $A$ will be

$$A = \overline{a} \times \overline{\beta}^T = \overline{\beta} \times \overline{a}^T,$$

where $\overline{a} = [a_0, a_1, \cdots, a_{l-1}], \overline{\beta} = [\beta, \beta^2, \cdots, \beta^{2^{l-1}}]$ and $T$ denotes the vector transposition. The merit of the normal basis representation is that the result of squaring an element $A$ is simply the right cyclic shift(RCS) of its coordinates. That is,

$$A^2 = (a_{l-1}, a_0, a_1, \cdots, a_{l-2}).$$

Let $A = \sum_{i=0}^{l-1} a_i \beta^{2^i}, B = \sum_{i=0}^{l-1} b_i \beta^{2^i} \in GF(2^l)$, where $a_i, b_i \in GF(2)$ and $C = AB = \sum_{i=0}^{l-1} c_i \beta^{2^i}$. Then

$$C = (\overline{a} \times \overline{\beta}^T) \times (\overline{\beta} \times \overline{b}^T) = \overline{a} M \overline{b}^T,$$

where the multiplication matrix $M$ is defined as

$$M = \overline{\beta}^T \times \overline{\beta} = (\beta^{(2^i + 2^j)}), 0 \le i, j \le l-1.$$

If each $\beta^{2^i + 2^j}$ is represented with respect to the normal basis, then we have $M = M_0 \beta + M_1 \beta^2 + \cdots + M_{l-1} \beta^{2^{l-1}}$, where $M_i$ is an $l$ by $l$ matrix over $GF(2)$. Using the property of squaring an element with normal basis representation, the coefficients of $C$ is obtained as below,

$$c_i = \overline{a} M_i \overline{b}^T = \overline{a}^{(i)} M_0 \overline{b}^{(i)^T},$$

where $\overline{a}^{(i)} = [a_i, a_{i+1}, \cdots, a_{i-1}], \overline{b}^{(i)} = [b_i, b_{i+1}, \cdots, b_{i-1}]$. From this result, we can show that the numbers of 1s in $M_i, 0 \le i \le l-1$ are the same. The number of 1s in each $M_i$ is called the complexity of the normal basis and denoted by $C_N$. Gao et al. proved that $C_N \ge 2l-1$ [3, 12]. Throughout this paper, each element of the finite field will be represented with respect to a normal basis. Now we describe a construction of normal bases of low complexity. Let $m, k$ be positive integers, $n = mk$ and $n+1 (\ne 2)$ prime. Let $\tau$ be an element of $GF(n+1)^*$ of order $k$ and $\gamma \in GF(2^n)$ be a primitive $n$-th root of unity. Suppose that $\gcd(n/e, m) = 1$, where $e$ is the order of 2 modulo $n+1$. Then $\beta = \gamma + \gamma^\tau + \gamma^{\tau^2} + \cdots + \gamma^{\tau^{k-1}}$ generates a normal basis for $GF(2^m)$ over $GF(2)$ [2,5,12], that is, $N = \{\beta, \beta^2, \beta^{2^2}, \cdots, \beta^{2^{m-1}}\}$ is a normal basis of $GF(2^m)$ over $GF(2)$. Then $\beta$ is called Gaussian period of type $(m, k)$ over $GF(2)$ [2,5,9]. If $C_N = 2m-1$, then $N$ is called the optimal normal basis of type-I or type-II according to $k = 1$ or 2 respectively. A polynomial whose coefficient are all 1s is called All-One-Polynomial(AOP), e.g. $x^n + x^{n-1} + \cdots + x + 1$. It is well known that $GF(2^n)$ has a type-I optimal normal basis over $GF(2)$ if and only if $n+1$ is prime and $GF(n+1)^* =< 2 >$, the fnite field generated by 2 modulo $n+1$ [12].

Throughout this paper, we assume that $m$ is odd and $GF(2^m)$ has a type-IV Gaussian normal basis. Then, by the following lemma, it is obvious that the finite field $GF(2^m)$ is a subfield of $GF(2^{4m})$.

**Lemma 1.** *Suppose that $m$ is odd and $GF(2^m)$ has a type-IV Gaussian normal basis, then $GF(4m+1)^* =< 2 >$, i.e. $GF(2^{4m})$ has a type-I optimal normal basis.*

*Proof.* Since $GF(2^m)$ has a type-IV Gaussian normal basis, $4m+1$ is prime and $(4m/e, m) = 1$, where $e$ is the order of 2 in $GF(4m+1)^*$. Therefore $e$ is equal to one of $m, 2m$ or $4m$. Moreover, if $GF(4m+1)^* =< g >$, where $2 = g^t, 0 \le t < 4m$, then $e = 4m/(t, 4m)$. Therefore $t$ is even in case $e = m, 2m$, and thus 2 is a quadratic residue of $4m+1$. On the other hand, $4m+1$ is the form of $8k+5$ since $m$ is odd. Therefore 2 is a quadratic nonresidue of $4m+1$, which leads to the contradiction. Consequently the order of 2 is $4m$ in $GF(4m+1)^*$.

Now, we like to find the methods for representing an element of $GF(2^m)$ as an element of $GF(2^n), n = 4m$. In the sequel, any element $A$ of $GF(2^n)$ will be denoted by the vector representation $A = (a_0, a_1, \cdots, a_{n-1})$. From the fundamental property of the finite field, for any element $B$ of $GF(2^{4m})$, $B$ is also an element of $GF(2^m)$ if and only if $B^{2^m} = B$. Therefore, using the property of squaring an element with respect to a normal basis, we obtain the following.

**Theorem 1.** *Let $B = (b_0, b_1, \cdots, b_{4m-1})$ be an element of $GF(2^{4m})$. Then $B$ is an element of $GF(2^m)$ if and only if $b_i = b_t$, where $i \equiv t \mod m$ for $0 \le i, t < 4m$, that is, $B = (b_0, b_1, \cdots, b_{m-1}, \cdots, b_0, b_1, \cdots, b_{m-1})$.*

Let $n = 4m$. Then $n+1$ prime and $GF(n+1)^* =< 2 >$. If $\gamma$ is a primitive $n+1$-th root of unity, then $\gamma$ generates a type-I optimal normal basis of $GF(2^n)$ over $GF(2)$ [12] and thus we have $\beta = \gamma + \gamma^{2^m} + \gamma^{2^{2m}} + \gamma^{2^{3m}}$ since $\tau = 2^m$. Therefore each element $A$ of $GF(2^m)$ can be represented as

$$A = A_0\beta + A_1\beta^2 + A_2\beta^{2^2} + \cdots + A_{m-1}\beta^{2^{m-1}},$$

where each $A_i \in GF(2)$. Since $GF(2^m)$ is a subfield of $GF(2^n)$, $A$ is represented with respect to normal bases in two ways as follows.

$$A = A_0\gamma + A_1\gamma^2 + A_2\gamma^{2^2} + \cdots + A_{m-1}\gamma^{2^{m-1}} + A_0\gamma^{2^m} + A_1\gamma^{2^{m+1}} + A_2\gamma^{2^{m+2}} + \cdots + A_{m-1}\gamma^{2^{2m-1}} + \cdots + A_0\gamma^{2^{3m}} + A_1\gamma^{2^{3m}} + A_2\gamma^{2^{3m+1}} + \cdots + A_{m-1}\gamma^{2^{4m-1}},$$

and

$$A = a_0\gamma + a_1\gamma^2 + a_2\gamma^{2^2} + \cdots + a_{4m-1}\gamma^{2^{4m-1}}.$$

We thus have

$$A_i = a_{i+tm}, 0 \le i \le m-1, 0 \le t \le 3. \quad \cdots \quad (*)$$

## 3 Serial Multiplier of Reyhani-Masoleh and Hasan Using AOP

In this section we revisit the structure of the serial multiplier proposed by Reyhani-Masoleh and Hasan [13,17] using AOP. For brevity, we use two notations $(\!(i)\!) \equiv i \bmod m$ and $\langle\!\langle i \rangle\!\rangle \equiv i \bmod n$ from now on. Let $GF(2^n)$ be the finite field generated by the AOP $x^n + x^{n-1} + \cdots + x + 1$ and $\gamma$ be a root of the AOP, then $\gamma$ generates the type-I optimal normal basis. Let $\delta_i = \beta^{1+2^i}$, then $n$ becomes even and $\delta_i = \gamma^{1+2^i}$, where $i = 1, 2, \cdots, v = n/2$, since $\beta = \gamma$. Then we have the following lemma since $\gamma$ is a root of the AOP.

**Lemma 2.**

$$\delta_i = \begin{cases} \gamma^{2^{k_i}}, & 1 \le i \le n/2-1, \\ 1 = \sum_{j=0}^{n-1} \gamma^{2^j}, & i = n/2, \end{cases}$$

*where $k_i$ satisfies the congruence $2^i + 1 \equiv 2^{k_i} \bmod n+1$.*

Reyhani-Masoleh and Hasan [13,17] proved the following lemma by substituting all the entries of the the multiplication matrix $M = (\beta^{2^i+2^j})$ by the element of the form $\beta^{2^j}$, $0 \le j \le n-1$.

**Lemma 3.** *Let $GF(2^n)$ be a finite field having a type-I optimal normal basis, $\gamma$ a generator of the optimal normal basis, $A, B \in GF(2^n)$, $C = AB$, and $g \in \{0,1\}$. Then*

$$C = \sum_{j=0}^{n-1} a_{\langle\langle j-g\rangle\rangle} b_{\langle\langle j-g\rangle\rangle} \gamma^{2^j} + \sum_{i=1}^{v-1} (\sum_{j=0}^{n-1} x_{j,i}\gamma^{2^j})^{2^{k_i}} + \sum_{j=0}^{n-1}(\sum_{i=1}^{v-1} x_{i,v})\gamma^{2^j}, v = n/2,$$

*where* $x_{j,i} = \begin{cases} a_j b_{\langle\langle i+j\rangle\rangle} + a_{\langle\langle i+j\rangle\rangle} b_j & \textit{if g=1} \\ (a_j + a_{\langle\langle i+j\rangle\rangle})(b_j + b_{\langle\langle i+j\rangle\rangle}) & \textit{if g=0.} \end{cases}$

## 4 The Type-IV Gaussian Normal Bases Multipliers

### 4.1 A New Serial Multiplier

Let $GF(2^m)$ has a type-IV Gaussian normal basis and m is odd, $n = 4m$. Then $n+1$ prime and $GF(n+1)^* =< 2 >$ . Now, We like to embed the elements $A, B \in GF(2^m)$ to the elements in $GF(2^n)$ with respect to type-I optimal basis and then construct a new multiplier for the finite subfield $GF(2^m)$ to calculate $C = AB$ modifying Reyhani-Masoleh and Hasan multiplier described in section 3. For the later, we now define an exponent.

**Definition 1.** *Suppose that $n = 4m, n+1$ prime, $GF(n+1)^* =< 2 >$ and $k_i$ is the exponent defined in Lemma 2. For $1 \le i_0 \le u = (m-1)/2$ and $i \in \{i_0, m-i_0, m+i_0, 2m-i_0\}$, we will define $\varsigma_i$ as follows.*

$$\varsigma_i = \begin{cases} ((k_i)), & i \equiv i_0 \bmod m, \\ ((k_i + i_0)), & i \equiv -i_0 \bmod m. \end{cases}$$

Then we have the following.

**Theorem 2.** *Assume that $GF(2^m)$ has a Gaussian normal basis of type-IV, $n = 4m$, $u = (m-1)/2$, and $g \in \{0,1\}$. If $A, B$ are belong to $GF(2^m)$ and $C$ is the product of $A$ and $B$, then*

$$C = \sum_{j=0}^{m-1} A_{((j-g))} B_{((j-g))} \beta^{2^j} + \sum_{j=0}^{m-1} (\sum_{i_0=1}^{u} x_{j,i_0} (\sum_{w=0}^{1} \beta^{2^{\varsigma_{wm+i_0}}} + \sum_{w=1}^{2} \beta^{2^{\varsigma_{wm-i_0}}}))^{2^j} ,$$

*where* $x_{j,i} = \begin{cases} A_j B_{((i+j))} + A_{((i+j))} B_j & \textit{if g=1,} \\ (A_j + A_{((i+j))})(B_j + B_{((i+j))}) & \textit{if g=0.} \end{cases}$

*Proof.* Without loss of generality, we prove the theorem for $g = 1$. Let $A, B \in GF(2^m) \subset GF(2^n)$. Thus, by Lemma 3

$$C = AB = \sum_{j=0}^{n-1} a_{\langle\langle j-1\rangle\rangle} b_{\langle\langle j-1\rangle\rangle} \gamma^{2^j} + \sum_{i=1}^{v-1}(\sum_{j=0}^{n-1} x_{j,i}\gamma^{2^j})^{2^{k_i}} + \sum_{j=0}^{n-1}(\sum_{i=1}^{v-1} x_{i,v})\gamma^{2^j}, v = 2m,$$

and $a_j = A_{((j))}, b_j = B_{((j))}, 0 \le j \le n-1$, from the equation $(*)$. Therefore we calculate only $A_iB_i$ for $i = 0, 1, 2, \cdots, m-1$. Next, if $i = wm, 1 \le w \le 2$, then

$$x_{j,i} = a_j b_{\langle\langle i+j\rangle\rangle} + a_{\langle\langle j+i\rangle rlangle} b_j = A_{((j))}B_{((i+j))} + A_{((i+j))}B_{((j))} = 0.$$

Lastly,

$$\begin{aligned} x_{j,i} &= a_j b_{\langle\langle j+i\rangle\rangle} + a_{\langle\langle j+i\rangle\rangle} b_j \\ &= A_j B_{((j+i)} + A_{((j+i))} B_j \\ &= x_{((j)),((i))}. \end{aligned}$$

Therefore, for $1 \le i \le (m-1)/2$,

$$\sum_{j=0}^{n-1} x_{j,i}\gamma^{2^j} = \sum_{t=0}^{3}(\sum_{j_0=0}^{m-1} x_{j_0,i}\gamma^{2^{j_0}})^{2^{tm}} = \sum_{j_0=0}^{m-1}(x_{j_0,i}\sum_{t=0}^{3}\gamma^{2^{tm}})^{2^{j_0}} = \sum_{j_0=0}^{m-1} x_{j_0,i}\beta^{2^{j_0}}.$$

Thus,

$$\sum_{i=1}^{v-1}(\sum_{j=0}^{n-1} x_{j,i}\gamma^{2^j})^{2^{k_i}} = \sum_{i=1}^{v-1}(\sum_{j_0=0}^{m-1} x_{j_0,i}\beta^{2^{j_0}})^{2^{k_i}} = \sum_{j_0=0}^{m-1}(\sum_{i=1}^{v-1} x_{j_0,i}\beta^{2^{k_i}})^{2^{j_0}}.$$

For $1 \le i_0 \le u = (m-1)/2$, we divide $i$s into two classes as follows.
(1) $i = wm + i_0, 0 \le w \le 1$,
(2) $i = wm - i_0, 0 \le w \le 2$.
For (1),

$$\begin{aligned} x_{j,i} &= x_{j,wm+i_0} \\ &= a_j b_{\langle\langle j+wm+i_0\rangle\rangle} + a_{\langle\langle j+wm+i_0\rangle\rangle} b_j \\ &= A_{((j))}B_{((j+i_0))} + A_{((j+i_0))}B_{((j))} \\ &= x_{((j)),i_0}. \end{aligned}$$

For (2),

$$\begin{aligned} x_{j,i} &= x_{j,wm-i_0} \\ &= a_j b_{\langle\langle j+wm-i_0\rangle\rangle} + a_{\langle\langle j+wm-i_0\rangle\rangle} b_j \\ &= A_{((j))}B_{((j-i_0))} + A_{((j-i_0))}B_{((j))} \\ &= x_{((j-i_0)),i_0}. \end{aligned}$$

Therefore,

$$\begin{aligned} C = AB &= \sum_{j=0}^{m-1} A_{((j-1))}B_{((j-1))}\beta^{2^j} + \sum_{j=1}^{m-1}(\sum_{i=0}^{v-1} x_{j,i}\beta^{2^{k_i}})^{2^j} \\ &= \sum_{j=0}^{m-1} A_{((j-1))}B_{((j-1))}\beta^{2^j} + \sum_{j=0}^{m-1}(\sum_{i_0=1}^{u} x_{j,i_0}(\sum_{w=0}^{1}\beta^{2^{k_{wm+i_0}}} + \sum_{w=1}^{2}\beta^{2^{k_{wm-i_0}+i_0}}))^{2^j}. \end{aligned}$$

This completes the proof.

From Theorem 2, if

$$G_j(A,B) = A_{((j-g))}B_{((j-g))}\beta + \sum_{i_0=1}^{u} x_{j,i_0}(\sum_{w=0}^{1}\beta^{2^{\varsigma_{wm+i_0}}} + \sum_{w=1}^{2}\beta^{2^{\varsigma_{wm-i_0}}}),$$

then

$$C = ((G_{m-1}^2 + G_{m-2})^2 + \cdots + G_1)^2 + G_0,$$

where $G_{m-t}(A,B) = G_{m-1}(A^{2^{t-1}}, B^{2^{t-1}})$. Moreover, since $\beta^{2^j}$ appears 4 times for each $j$, there can occur at most $4u+1$ terms in the right hand side of the last equation . However, that $\beta^{2^i}$ appears twice for each $i_0$ means the same value is added twice. Therefore such terms can be neglected. Consequently, the number of XOR gates of the serial multiplier is

$$M = |\{\beta\}| + \sum_{i_0=1}^{u} |\{\beta^{2^{\varsigma_{i_0}}}, \beta^{2^{\varsigma_{m-i_0}}}, \beta^{2^{\varsigma_{m+i_0}}}, \beta^{2^{\varsigma_{2m-i_0}}}\}| + \epsilon u,$$

where

$$\epsilon = \begin{cases} 1\ if & g=1, \\ 2\ if & g=0. \end{cases}$$

And if we set $l = Max_{j=0}^{m-1} M_j + 1$, where $M_j = |\{wm+i_0|\varsigma_{wm+i_0} = j, 0 \le w \le 1\}| + |\{wm-i_0|\varsigma_{wm-i_0} = j, 1 \le w \le 2\}| + t$ and

$$t = \begin{cases} 1\ if j = 0, \\ 0\ if j \neq 0, \end{cases}$$

then $l$ determines the critical path delay of the XOR gates. To reduce the value of $l$, we need the following.

**Corollary 1.** *Assume that $GF(2^m)$ has a type-IV Gaussian normal basis,$n = 4m$, and $g \in \{0,1\}$. If $A, B \in GF(2^m) \subset GF(2^n)$ and $C = AB$, then*

$$C = \sum_{j=0}^{m-1}(A_{((j+j_0-g))}B_{((j+j_0-g))}\beta^{2^{j_0}})^{2^j}$$
$$+ \sum_{j=0}^{m-1}(\sum_{i_0=1}^{u} x_{((j+j_{i_0})),i_0}(\sum_{w=0}^{1}\beta^{2^{\varsigma_{wm+i_0}+j_{i_0}}} + \sum_{w=1}^{2}\beta^{2^{\varsigma_{wm-i_0}+j_{i_0}}}))^{2^j},$$

*where* $x_{j,i} = \begin{cases} A_jB_{((i+j))} + A_{((i+j))}B_j & \text{if g=1,} \\ (A_j + A_{((i+j))})(B_j + B_{((i+j))}) & \text{if g=0.} \end{cases}$

*Proof.* If we perform $j_{i_0}$-fold right cyclic shift on each basic element $\beta^{2^j}$ appeared in $C$ in Theorem 2 for each $i_0$, then

$$C = \sum_{j=0}^{m-1}(A_{((j-g))}B_{((j-g))}\beta^{2^{j_0}})^{2^{j-j_0}}$$
$$+ \sum_{j=0}^{m-1}(\sum_{i_0=0}^{u} x_{j,i_0}(\sum_{w=0}^{1}\beta^{2^{\varsigma_{wm+i_0}+j_{i_0}}} + \sum_{w=1}^{2}\beta^{2^{\varsigma_{wm-i_0}+j_{i_0}}}))^{2^{j-j_{i_0}}}.$$

Substitute the value of $\beta$ which has earlier derived and matches the bits, then we obtain the result.

Therefore if we define

$$G'_j(A,B) = A_{((j+j_0-g))}B_{((j+j_0-g))}\beta^{2^{j_0}} + \sum_{i_0=1}^{u} x_{((j+j_{i_0})),i_0}\Big(\sum_{w=0}^{1}\beta^{2^{\varsigma_{wm+i_0}+j_{i_0}}} + \sum_{w=1}^{2}\beta^{2^{\varsigma_{wm-i_0}+j_{i_0}}}\Big),$$

then

$$C = ((G'^2_{m-1} + G'_{m-2})^2 + \cdots + G'_1)^2 + G'_0,$$

where $G'_{m-t}(A,B) = G'_{m-1}(A^{2^{t-1}}, B^{2^{t-1}})$ by Corollary 1.

Let $l' = Max_{j=0}^{m-1} M'_j + 1$, where $M'_j = |\{wm + i_0 | \varsigma_{wm+i_0} + j_{i_0} = j, 0 \le w \le 1\}| + |\{wm - i_0 | \varsigma_{wm-i_0} + j_{i_0} = j, 1 \le w \le 2\}|$. Then the process of calculating $C$ in Corollary 1 is as follows. Firstly, for each $i_0$, we seek for $j_{i_0}$ such that $l'$ becomes to be optimal and choose $j_0$ being different from $l'$. Next, $G'_j(A,B)$ is obtained by $j_0$-fold and $j_{i_0}$-fold right shift of the first term and the second term respectively in the equation defining $G_j(A,B)$. Lastly, $C$ is calculated by $G'_j$s and thus the XOR path delay for calculating $C$ can be reduced to $\lceil \log_2 l' \rceil$. Therefore the critical XOR path delay is determined by $l'$.

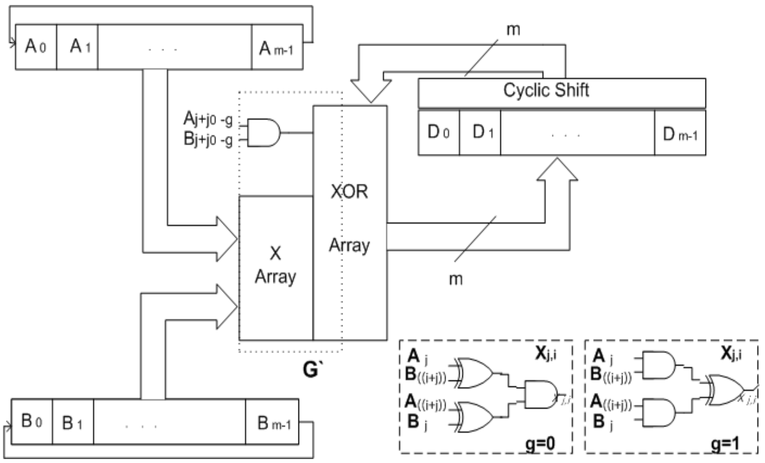


**Fig. 1.** The structure of the proposed multiplier over $GF(2^m)$

The serial multiplier given in Fig.1 is constructed from Corollary 1. The Z array calculates $x_{j,i}$ for each $1 \le i \le u = (m-1)/2$. $G'_{m-1}$ is calculated in the block $G'$, and the block $X_{j,i}$ indicates the methods of operations according to $g = 0$ or $g = 1$.

## 4.2 Optimization

In implementing our proposed multiplier, for a fixed $1 \le i_0 \le u = (m-1)/2$, if some

$$\varsigma_i, \quad i = i_0, m - i_0, m + i_0, 2m - i_0,$$

in Theorem 2 are the same, then so are the outputs of $\beta^{2^{\varsigma_i}}$. Therefore the result is not changed if we discard them. Thus we can reduce the number of XOR gates and path delay. In this regard, the number of XOR gates and path delay can be reduced by confirming whether the values of $\varsigma_i$ coincide for some $i$.

**Lemma 4.** *Assume that $m$ odd, $4m+1$ prime and $GF(4m+1)^* =< 2 >$. Then one of $k_1 = k_2 + u \bmod m$ or $k_3 = k_4 + u \bmod m$ holds for $u = (m-1)/2$ if, in $GF(4m+1)^* =< 2 >$,*

$$\begin{cases} 2^u + 1 = 2^{k_1}, & 2^{m-u} + 1 = 2^{k_2}, \\ 2^{m+u} + 1 = 2^{k_3}, & 2^{2m-u} + 1 = 2^{k_4}. \end{cases}$$

*Proof.* All the following equations are modulo $4m+1$. Since $(1+2^m)^2 = 2^{m+1} = (2^{u+1})^2$, we have $1+2^m = \pm 2^{u+1}$. From this fact, we will first prove $2^{m+u} - 2^u = \pm 1$. Using $2^{2m} = -1$,

$$\begin{aligned} 2^{m+u} - 2^u &= 2^{m+u} + 2^{2m+u} \\ &= 2^{m+u}(1+2^m) \\ &= \begin{cases} -1 & \text{if } 1+2^m = 2^{u+1}, \\ 1 & \text{if } 1+2^m = -2^{u+1}. \end{cases} \end{aligned}$$

Next, we prove this lemma by dividing into two cases i.e. $1+2^m = 2^{u+1}$ and $1+2^m = -2^{u+1}$.
(1) If $1+2^m = -2^{u+1}$, then $2^u + 1 = 2^{m+u}$. Therefore

$$\begin{aligned} 2^{m-u} + 1 &= 2^{u+1} + 1 \\ &= -(2^m + 1) + 1 \\ &= -2^m = 2^{3m}. \end{aligned}$$

(2) If $1+2^m = 2^{u+1}$, then $2^{m+u} + 1 = 2^u$. Thus we have

$$2^{2m-u} + 1 = 2^{m+u+1} + 1 = 2^m 2^{u+1} + 1 = 2^m(1+2^m) + 1 = 2^m.$$

This completes the proof.

By Lemma 4, if $GF(2^m)$ have a type-IV Gaussian normal basis and $m$ be odd then for $u = (m-1)/2$, either $\varsigma_u = \varsigma_{m-u}$ or $\varsigma_{m+u} = \varsigma_{2m-u}$. Thus, for $g = 1$, there need $M = (5m-7)/2$ XOR gates.

Now, we like to give an algorithm in order to find a method for calculating the optimal $l'$ by determining $j_0, j_{i_0}$ satisfying Corollary 1 with respect to each type-IV Gaussian normal basis. First of all, let us define three sets and a matrix for $1 \le i_0 \le u$, as follows.

$T_{i_0} = \{i_0, m - i_0, m + i_0, 2m - i_0\}$, $S_{i_0} = \{\varsigma_t | t \in T_{i_0}\}$, $S'_{i_0} = S_{i_0} - \{\varsigma_i | \varsigma_i = \varsigma_j, i \neq j \in T_{i_0}\}$, $S = (s_{ij})_{1 \leq i \leq u, 0 \leq j \leq m-1}$, where $s_{ij} = \begin{cases} 1 \text{ if } j \in S'_i, \\ 0 \text{ otherwise.} \end{cases}$

**Algorithm 1.** (Algorithm for finding $S'$)

```
Input : $S, l' - 1 = 3, m$
Output : $S'$
1. $l = l'; j = 0; u = (m-1)/2; mmm = m;$
2. $ttt = l; ss = 0;$
3. While($ttt > l - 1$)
    3.1 $j = 0;$
    3.2 While($j < mmm$)
        3.2.1 $i = 0; ttt = 0;$
        3.2.2 While($i < u$)
            3.2.2.1 If $s_{ij} = 1$, then $ttt = ttt + 1;$
            3.2.2.2 If $ttt = l$, then $ss = i; i = u; j = mmm;$
            3.2.2.3 $i = i + 1$
        3.2.3 $j = j + 1;$
    3.3 If $ttt > l - 1$, then $S = S_{ss} >> 1$.
4. Return($S$).
```

Here, $S_{ss} >> 1$ means 1-fold right cyclic shift(RCS) of $ss$-th row in $S$. There needs a total of $2m - 4$ XOR gates to calculate the optimal value of $l'$. In other words, $S$ has $\sum_{i=1}^{u} M_i - 1 = 2m - 4$ nonzero entries. We like to find the target matrix $S'$ containing at most $(l' - 1)$ nonzero entries in each column using Algorithm 1. Although the lower bound of $l' - 1$ is equal to 2 in this case, it is sufficient to seek for $S'$ containing three 1s (note that $l' - 1 = 3$) in each column since the value of the critical XOR path delay is $1 + \lceil \log_2 4 \rceil$.

*Remark 1.* For $m \leq 1000$, we can easily find the target matrix $S'$ with $l' - 1 = 3$ for $GF(2^m)$ using Algorithm 1.

## 5 Complexity

In this chapter, we calculate the complexities of the serial multiplier constructed in Theorem 2 and Corollary 1 of section 4.1.

**Theorem 3.** *The maximum complexities of the multiplier of Theorem 2 and Corollary 1 are*
*a) $m$ AND gates, $(5m - 7)/2$ XOR gates in case $g = 1$, $(m + 1)/2$ AND gates, $3m - 4$ XOR gates in case $g = 0$ and*
*b) $T_A + 3T_X$ path delay, where $T_A$ and $T_X$ denote AND delay and XOR delay respectively.*

*Proof.* For a), if $g = 1$, then there need one AND gate in order to calculate $A_{j-1}B_{j-1}$ and the total number of AND gates to generate $x_{j,i_0}$ is $m - 1$ since, for each $1 \leq i_0 \leq u = (m-1)/2$, there need two AND gates in order to calculate $x_{j,i_0}$. Therefore we need a total of $m$ AND gates. Since the number of XOR

**Table 1.** Comparison of Sequential Type-IV Gaussian Normal Basis Multipliers over $GF(2^m)$

| Multipliers | # AND | # XOR | Critical Path Delay | flip-flop |
|---|---|---|---|---|
| MO[11] | $4m-7$ | $\leq (4m-8)$ | $T_A + \lceil \lg(4m) \rceil T_X$ | $2m$ |
| Agnew et.al. [1] | $m$ | $\leq 4m-7$ | $T_A + 3T_X$ | $3m$ |
| Reyhani- Masoleh and Hasan [17] | $m$ | $(5m-7)/2$ | $T_A + 4T_X$ | $3m$ |
| Reyhani- Masoleh and Hasan [17] | $(m+1)/2$ | $3m-4$ | $T_A + 4T_X$ | $3m$ |
| Kwon et.al. [9] | $m$ | $(5m-3)/2$ | $T_A + 3T_X$ | $3m$ |
| Proposed $g=1$ | $m$ | $(5m-7)/2$ | $T_A + 3T_X$ | $3m$ |
| Proposed $g=0$ | $(m+1)/2$ | $3m-4$ | $T_A + 3T_X$ | $3m$ |

gate to generate each $x_{j,i_0}$ is 1 for each $1 \leq i_0 \leq u = (m-1)/2$, there needs a total of $(m-1)/2$ XOR gates. There need $1 + 2(m-1)$ XOR gates to calculate $G_j(A,B)^2 + G_{j-1}(A,B)$ except $x_{j,i_0}$, and thus we need a total of $5(m-1)/2+1$ XOR gates. By the way, for $i_0 = u$, two $\varsigma_{i_0}$ coincide by Lemma 4. Therefore the optimized total number of XOR gates is $(5m-7)/2$. For $g=0$, there need $(m-1)/2$ and 1 AND gates to generate $x_{j,i_0}$ and $A_jB_j$ respectively and thus we need a total of $(m+1)/2$ AND gates. And there need a total $3m-4$ XOR gates in case $g=0$ similarly.
For b), it is immediately that both of the number of path delay of AND gates and that of XOR gates to generate $x_{j,i_0}$ are equal to 1. Thus the critical XOR path delay is $1 + \lceil \log_2 4 \rceil$ since the number of the upper bound of the number of XOR gates to generate the basic element $\beta^{2^j}$ in $G_j(A,B)^2 + G_{j-1}(A,B)$ except $x_{j,i_0}$ is 4. This completes the proof.

Thus, we can obtain the following table 1.

## 6 Example

1) For $g=1, m=7$, we obtain the matrices $S$ and $S'$ from Algorithm 1 as follows.

$$S = \begin{pmatrix} 1\,0\,1\,0\,0\,1\,1 \\ 0\,1\,0\,1\,1\,1\,0 \\ 0\,0\,1\,0\,0\,1\,0 \end{pmatrix}, \qquad S' = \begin{pmatrix} 1\,0\,1\,0\,0\,1\,1 \\ 0\,1\,0\,1\,1\,1\,0 \\ 0\,0\,0\,1\,0\,0\,1 \end{pmatrix}.$$

In this case $l'-1=2$. Then $j_1 = 0,\ j_2 = 0,\ j_3 = 1$, and $G'_{m-1} = A_5B_5\beta + x_{6,1}(\beta+\beta^{2^2}+\beta^{2^5}+\beta^{2^6}) + x_{6,2}(\beta^2+\beta^{2^3}+\beta^{2^4}+\beta^{2^5}) + x_{0,3}(\beta^{2^3}+\beta^{2^6})$, where $x_{6,1} = A_6B_0 + A_0B_6,\ x_{6,2} = A_6B_1 + A_1B_6,\ x_{0,3} = A_0B_3 + A_3B_0$. The corresponding serial multiplier for $GF(2^7)$ is given in Fig.2.
2) If $g=1, m=37$, then $n=148$ and $GF(149)^* =< 2 >$. Since

$$G_j(A,B) = A_{j-1}B_{j-1}\beta + \sum_{i_0=1}^{u} x_{j,i_0}\left(\sum_{w=0}^{1} \beta^{2^{\varsigma_{wm+i_0}}} + \sum_{w=1}^{2} \beta^{2^{\varsigma_{wm-i_0}}}\right),$$

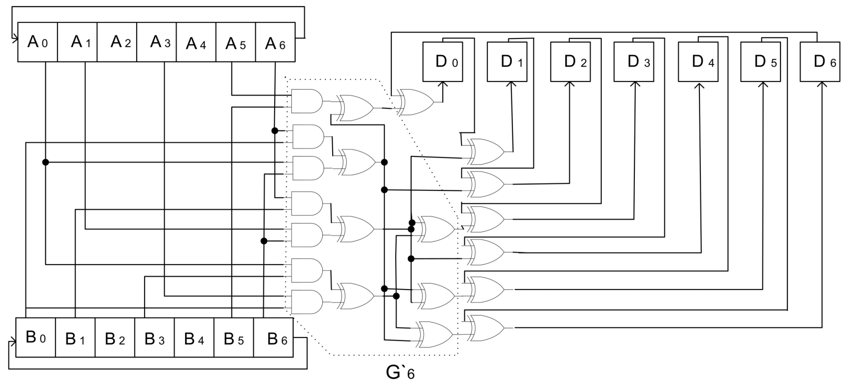


**Fig. 2.** The serial multiplier for $GF(2^7)$

$$S=\left(\begin{array}{*{37}{c}}
1&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1\\
0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1\\
0&1&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0\\
\hdashline
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0
\end{array}\right)$$

$$S'=\left(\begin{array}{*{37}{c}}
1&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1\\
0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1\\
0&1&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0
\end{array}\right)$$

**Fig. 3.** The matrices $S$ and $S'$ for $GF(2^{37})$

the entries of $j$-th column of the matrix $S$ are 1 if $j \in \{\varsigma_{i_0}, \varsigma_{m-i_0}, \varsigma_{m+i_0}, \varsigma_{2m-i_0}\}$ for $1 \leq i_0 \leq u$ and 0 otherwise. By twice right cyclic shift of 16-th row, we can obtain the target matrix $S'$ having at most three 1s in each column. The matrices $S$ and $S'$ are given in Fig.3.

## 7 Conclusion

In Lemma 1, we proved that $GF(4m+1)^* =< 2 >$ if $GF(2^m)$ has a type-IV Gaussian normal basis and $m$ odd. From Lemma 1 and using the fact that if $GF(2^m)$ having type-IV Gaussian normal basis and $m$ is odd, then $GF(2^m)$ is a subfield of $GF(2^{4m})$ having a type-I optimal normal basis, in this paper, we propose a new architecture for SMPO, which transforms the Gaussian normal basis multiplication in $GF(2^m)$ into the type-I optimal normal basis multiplication in $GF(2^{4m})$ based on Reyhani-Masoleh and Hasan's SMPO over $GF(2^m)$

having a type-IV Gaussian normal basis. From Table 1, we can confirm that our proposed SMPO reduces the critical XOR path delay of the serial multiplier of Reyhani-Masoleh and Hasan by 25% and has the same critical path delay as that of Kwon et al. And the number of XOR gates of our proposed serial multiplier is the same as that of Reyhani-Masoleh and Hasan and is fewer than that of Kwon et al. by 2. Therefore we expect our proposed serial multiplier will be efficiently applied to hardware implementations in the related application areas.

## References

1. G.B. Agnew, R.C. Mullin, I. Onyszchuk and S.A. Vanstone, *An implementation for a fast public key cryptosystem,* J. Cryptography, vol.3, pp.63-79, 1991.
2. ANSI X 9.63, *Public key cryptography for the financial sevices industry: Elliptic curve key agreement and transport protocols,* draft,1998.
3. S. Gao Jr. and H.W. Lenstra, *Optimal normal bases,* Designs, Codes and Cryptography, vol. 2, pp.315-323, 1992.
4. M.A. Hasan, M.Z. Wang, and V.K. Bhargava, *A modified Massey-Omura parallel multiplier for a class of finite fields,* IEEE Trans. vol.42, no.10, pp. 1278-1280, Oct, 1993.
5. IEEE P1363, *Standard specifications for public key cryptography,* Draft 13, 1999.
6. T. Itoh and S. Tsujii, *Structure of parallel multipliers for a class of fields,* Information and Computation, vol.83, pp. 21-40, 1989.
7. C.H. Kim, S. Oh, and J. Lim, *A new hardware architecture for operations in $GF(2^n)$,* IEEE Trans. vol.51, no.1, pp. 90-92, Jan. 2002.
8. C.K. Koc and B. Sunar, *Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields,* IEEE Trans. vol.47, no.3, pp. 353-356, Mar, 1998.
9. S. Kwon, K. Gaj, C.H.Kim and C.P. Hong, *Efficient Linear Array for Multiplication in $GF(2^m)$ Using a Normal Basis for Elliptic Curve Cryptography ,* CHES 2004, LNCS 3156, pp.76-91, 2004.
10. R. Lidl and H. Niederreiter, *Introduction to finite fields and its applications,* Cambridge Univ. Press, 1994.
11. J.L. Massey and J.K. Omura, *Computational method and apparatus for finite field arithmetic,* US Patent NO. 4587627, 1986.
12. A.J. Menezes, I.F. Blake, X. Gao, R.C. Mullin, S.A. Vanstone, and T. Yaghoobian, *Applications of finite fields,* Kluwer Academic, 1993.
13. A. Reyhani-Masoleh and M.H. Hasan, *Low complexity sequential normal basis multipliers over $GF(2^m)$,* 16th IEEE Symposium on Computer Arithmetic, vol.16, pp. 188-195, 2003.
14. A. Reyhani-Masoleh and M.H. Hasan, *Efficient Digit-Serial Normal Basis Multipliers over Binary Extension Fields,* ACM Trans. on Embedded $m(m+1)/2$ ed Computing Systems(TECS), Special Issue on Embedded Systems and Security, pp. 575-592, vol.3, Issue 3, August 2004.
15. A. Reyhani-Masoleh and M.H. Hasan, *A new construction of Massey-Omura parallel multiplier over $GF(2^m)$,* IEEE Trans. vol.51, no.5, pp. 512-520, May, 2002.
16. A. Reyhani-Masoleh and M.H. Hasan, *Efficient multiplication beyond optimal normal bases,* IEEE Trans. vol.52, no.4, pp. 428-439, April, 2003.

17. A. Reyhani-Masoleh and M.H. Hasan, *Low Complexity Word-Level Sequential Normal Basis Multipliers,* IEEE Trans. vol.54, no.2, pp. 98-110, February, 2005.
18. C.C Wang, T.K. Truong, H.M. Shao, L.J. Deutsch, J.K. Omura, and I.S. Reed, *VLSI architectures for computing multiplications and inverses in* $GF(2^n)$*,*
19. H. Wu and M.A. Hasan, *Low Complexity bit-parallel multipliers for a class of finite fields,* IEEE Trans. vol.47, no.8, pp. 883-887, Aug., 1998.
20. D.J.Yang, C.H.Kim, Y.Park, Y.Kim and J.Lim, *Modified sequential Normal Basis Multipliers for Type II Optimal Normal Basis*, ICCSA 2005, LNCS 3481, pp.647-656, 2005.

# Scalar Multiplication on Elliptic Curves Defined over Fields of Small Odd Characteristic

Christophe Negre

Equipe DALI,
Laboratoire LP2A, Université de Perpignan,
52 Avenue Alduy, 66860 Perpignan, France

**Abstract.** This paper explores elliptic curve cryptosystems over fields of small odd characteristic $p = 3$ or 5. We establish formulas multiplying by $p$ a random point on an ordinary curve defined over $\mathbb{F}_{p^n}$, thereby improving scalar multiplication on random and special curves for $p = 3$ or 5 using a $p$-Multiply-and-Add method. We study the complexity of our method and compare it to other schemes.

**Keywords:** Elliptic curve, cryptography, scalar multiplication, small characteristic.

## 1 Introduction

Elliptic curve cryptography (ECC), proposed independently by Koblitz [11] and Miller [13] in 1985, is a more efficient and secure alternative to the RSA [19] cryptosytem. ECC security is based on the difficulty of the discrete logarithm problem in the group of points of elliptic curves.

The most important operation of ECC protocols is the scalar multiplication of a point on a curve. Specifically, given an integer $k$ and a point $Q$ on the curve, we have to compute $kQ$. Scalar multiplication is usually done by a chain of double and add using the so-called *Double-and-Add* method. Doubling and Adding points on the curve require several field multiplications, additions, and one eventual inversion of the point coordinates. Hence, the efficiency of the protocols relies deeply on the arithmetic of the curve and the arithmetic of the underlying field.

Until now, research efforts have focused only on large prime fields and binary fields to provide efficient and secure implementations of ECC protocols. Other types of finite fields, for example fields of small odd characteristic, have received little attention: only few works concerning supersingular curves on $\mathbb{F}_{3^n}$ have been done, motivated by the existence of curves suitable for identity-based cryptosystems (cf. the work of Boneh *et al.* [2]) and few works on Koblitz curve which concern a small set of ellitptic curve (cf. [21]).

In this paper we focus on ordinary curves defined over $\mathbb{F}_{p^n}$ when $p = 3, 5$. We noticed that the multiplication of a point by $p$ can be done efficiently when $p$ is small. This fact comes from an underlying property of morphisms between curves: the morphism $Q \mapsto pQ$ splits into the composition of the Frobenius

© Springer-Verlag Berlin Heidelberg 2005

morphism and another curve morphism, which is simpler to evaluate than $pQ$. To improve the efficiency of scalar multiplication on the curve, we propose replacing the Double-and-Add method with a *p-Multiply-and-Add* method, which exploits the efficiency of multiplication by $p$.

The remainder of this paper is organized as follows: in the first section we review some basic facts on elliptic curves over finite fields, finite field arithmetic and point representation. Then we establish the general property of the $p$-multiplication in curves $E(\mathbb{F}_{p^n})$. We present the $p$-Multiply-and-Add method to do scalar multiplication. In the fifth section we give the formulas for computing $3Q$ in characteristic 3, and study the global cost of scalar multiplication. In the sixth section we do the same for the characteristic 5 case. We finish with a brief conclusion.

## 2 Background

Given a finite group with underlying difficult discrete logarithm problem (DLP) and efficient group law, one could use this group to implement cryptographic protocols such as ElGamal encryption [7] or Diffie-Hellman key exchange [6].

Elliptic curves have a group structure which provides efficient group arithmetic and difficult DLP. Indeed, let $\mathbb{F}_{p^n}$ be a finite field, with $p > 3$ (resp. $p = 3$), an elliptic curve $E$ over $\mathbb{F}_{p^n}$ is the set of pairs $(x, y) \in \mathbb{F}_{p^n} \times \mathbb{F}_{p^n}$, satisfying an equation of the form $Y^2 = X^3 + aX + b$ (resp. $Y^2 = X^3 + aX^2 + b$ when $p = 3$), plus a point at infinity $\mathcal{O}$. We have a group law on the set of points of $E$: given two points $Q_1 = (x_1, y_1)$ and $Q_2 = (x_2, y_2)$ on $E$, we compute the addition $Q_3 = (x_3, y_3) = Q_1 + Q_2$ by

$$\text{when } p > 3$$
$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_3 - x_1) + y_1 \end{cases} \text{where } \lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \text{ if } Q_1 \neq \pm Q_2 \\ \frac{3x_1^2 + a}{2y_1} \text{ if } Q_1 = Q_2 \end{cases}$$

$$\text{when } p = 3$$
$$\begin{cases} x_3 = \lambda^2 - a - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases} \text{where } \lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \text{ if } Q_1 \neq \pm Q_2 \\ \frac{ax_1}{y_1} \text{ if } Q_1 = Q_2 \end{cases}$$

Before proceeding, we note that the formulas of the group law need inversion in the field, which is a relatively expensive operation compared to multiplication. It is possible to avoid inversion in addition and doubling formulas when using projective coordinates. A projective point can be set in an affine form by one inversion and several multiplications.

There are different types of projective coordinates which yield different efficiencies for the addition and doubling formulas, the most used are the Jacobian coordinates. For more on these coordinates, we refer to the work of Cohen *et al.* [4]. In the same paper, Cohen *et al.* pointed out that the use of mixed of coordinates, affine and Jacobian, could speed-up scalar multiplication.

Efficient scalar multiplication requires fast arithmetic in the field $\mathbb{F}_{p^n}$. For now, efficient arithmetic can be done by using a special representation of $\mathbb{F}_{p^n}$:

polynomial basis representation [1], or normal basis representation [15]. When we use special polynomial basis, or normal basis, the powering to $p$ in $\mathbb{F}_{p^n}$ will be markedly cheaper compared to the cost of multiplication. For example, when $p = 3, 5$ the exponentiation to $p$ has a cost of roughly $2n$ additions in $\mathbb{F}_p$ using suitable polynomial basis, and is free of computation for normal basis.

We refer to the papers [9, 16] of Smart and Page for a practical algorithm for multiplication and cubing in fields $\mathbb{F}_{3^n}$. These methods can be extended to field $\mathbb{F}_{p^n}$ for small odd $p$.

Using elliptic curve cryptosystems on these types of fields necessitates special requirements for security. Specifically, as stated by Diem in [5] we have to employ prime degree $n \geq 11$ for the degree of $\mathbb{F}_{p^n}$ to prevent the Weil Decent attack. The other requirements are more usual: to prevent Pohlig-Hellman's attack [17] and other generic attacks on DLP such as the Pollard rho method [18], we have to use a curve whose cardinality is either a big prime number or a product of a small integer with a big prime number.

## 3 Computing *pQ* in Characteristic *p*

Let $E$ be an elliptic curve defined over the field $\mathbb{F}_{p^n}$ by the equation $Y^2 = X^3 + aX + b$ when $p > 3$ (but the same could be done for $Y^2 = X^3 + aX^2 + b$ when $p = 3$). The multiplication by $p$ on the curve $E$ inducesa morphism of curve $[p]\colon\ E \to E$ defined by $Q \mapsto pQ$.

We will split this morphism to get a simpler expression of $pQ$. For this, first we denote $\sigma$ the Frobenius of $\mathbb{F}_{p^n}$ over $\mathbb{F}_p$ defined by $u \mapsto u^p$.

Next, if we define $E^\sigma$ the elliptic curve given by the equation $Y^2 = X^3 + \sigma(a)X + \sigma(b)$, we can construct a morphism $F$ from $E$ to $E^\sigma$ where $F$ is defined by $F((x, y)) = (\sigma(x), \sigma(y))$. From [20] we have the following splitting of $[p]$

$$\begin{tikzcd} E \arrow[rr, "{[p]}"] \arrow[dr, "F"'] & & E \\ & E^\sigma \arrow[ur, "\phi"'] & \end{tikzcd} \tag{1}$$

where the morphism $\phi\colon E^\sigma \to E$ expresses as $\phi(x, y) = \left(\frac{P_x(x)}{P_z(x)}, \frac{P_y(x)y}{P_z(x)}\right)$ where $P_x, P_y$ and $P_z$ are polynomials such that $\deg P_x = p$, $\deg P_y = p+2$ and $\deg P_z = (p-1)/2$. The splitting of $[p]$ for a point $Q = (x, y)$ of $E$ implies

$$pQ = \phi(F(Q)) = \left(\frac{P_x(x^p)}{P_z(x^p)}, \frac{P_y(x^p)y^p}{P_z(x^p)}\right). \tag{2}$$

Consequently, to multiply a point $Q$ by $p$ , we first compute $F(Q)$ which is cheap because of the low cost of powering to $p$ in $\mathbb{F}_{p^n}$; secondly we evaluate $\phi$ at $F(Q)$. Now, the cost of the multiplication by $p$ of the point $Q$ is roughly equal the the cost of the evaluation $\phi$ at the point $F(Q)$. When $p$ is small, the polynomials $P_x, P_y$ and $P_z$ have small degrees. The evaluation of $\phi$ at the point $F(Q)$ should be in this case, easy to compute thereby obtaining an efficient multiplication by $p$ on the curve $E$.

## 4 Scalar Multiplication with a $p$-Multiply-and-Add Method

We can use this efficient multiplication by $p$ to improve scalar multiplication on the curve. The most common method to multiply a point $Q$ by an integer $k$ is the Double-and-Add method, with a classical binary representation of the integer $k$, or with a signed representation (e.g. the Non-Adjacent Form [12]). Other methods use memory storage, and differ on whether the point $Q$ is fixed or not. For a good survey on these methods, refer to the paper by Hankerson *et al.* [12].

In this paper we will study only scalar exponentiation methods for non-fixed point $Q$. The Signed Window Radix-$p$ representation ($SWR_{p,w}$) is a version of sliding signed binary window representations (cf. the $NAF_w$ given in [12]) for the case of radix-$p$ representation with $p > 2$. Specifically, for an integer $k$, an $SWR_{p,w}$ representation of $k$ is given by the coefficients of the expression of $k$ in a signed radix-$p$ expansion $k = \sum_{i=0}^{\ell-1} k_i p^i$, where each $|k_i| < \frac{p^w}{2}, k_i \not\equiv 0 \mod p$ and between the two non-zero $k_i$ there is at least $w-1$ zeros (unless one of the two non-zero $k_i$ is the leading coefficient $k_{\ell-1}$).

**Algorithm 1.** Signed window in radix-$p$

```
Require: An integer $k \geq 0$, a prime $p$ and a window length $w$
Ensure: The $SWR_{p,w}$ of $k$
  $i \leftarrow 0$
  while $k > \frac{p^w}{2}$ do
    if $k \equiv 0 \mod p$ then
      $k_i \leftarrow 0, k \leftarrow k/p$
    else
      $k_i \leftarrow k \mod_s p^w,\ k \leftarrow (k - k_i)/p$
    end if
    $i \leftarrow (i+1)$
  end while
  $k_i \leftarrow k, \ell \leftarrow (i+1)$
  Return $(k_{\ell-1}, \ldots, k_0)$
```

It is not difficult to see that the length of the representation of $k > 0$ is smaller than $\lfloor \log_p(k) \rfloor + 1$. Moreover the number of non-zero $k_i$ in $SWR_{p,w}$ is equal to $\frac{\ell}{w+\frac{1}{p-1}}$ since in $(p-1)w+1$ consecutive coefficients, there is on average, exactly $(p-1)$ non-zero coefficients. We can use this representation to do scalar multiplication on the elliptic curve $E(\mathbb{F}_{p^n})$. We use a modified version of the Double-and-Add method: the *p-Multiply-and-Add* method. The integer $k$ is given by its $SWR_{p,w}$ representation $SWR_{p,w}(k) = (k_{\ell-1}, \ldots, k_0)_p$.

The general complexity of this scalar multiplication method is then equal to $\ell\text{Mult}_p + \frac{\ell}{w+\frac{1}{p-1}}\text{Add} + \textit{Precomputations}$. To get a complete evaluation of the complexity of this method, we have to explain how we do the precomputations.

**Algorithm 2.** $p$-Multiply-and-Add

```
Require: A curve $E(\mathbb{F}_{p^n})$, a point $Q \in E$ and the $SWR_{p,w} = (k_{\ell-1}, \dots, k_0)$ representation of an integer $k$ .
  Precomputations. for $i = 0, \dots, \frac{p^w-1}{2}$ do $T[i] \leftarrow iQ$.
  $R \leftarrow \mathcal{O}$
  for $i = \ell - 1$ to $0$ do
    $R \leftarrow pR$
    if $k_i \geq 0$ then
      $R \leftarrow R + T[k_i]$
    else
      $R \leftarrow R - T[-k_i]$
    end if
  end for
Ensure: $R$
```

**Algorithm 3.** Radix-$p$ precomputations

```
Require: The size of the window $w$, the elliptic curve $E$ and a point $P$
Ensure: A table $T$ containing $iP$ for $1 < i \leq \frac{p^w-1}{2}$ and $i \not\equiv 0 \mod p$
  $C[1] \leftarrow P, T[0] \leftarrow \mathcal{O}, T[1] \leftarrow P$
  for $i = 1$ to $(p-1)$ do
    $C[i+1] \leftarrow C[i] + P$ { Computation of $C_0$ and $\mathcal{S}'_0 = \mathcal{C}_0$}
    $T[i+1] \leftarrow C[i+1]$
  end for
  for $t = 1$ to $w - 2$ do
    for $i = 1$ to $(p-1)$ do
      $C[i] \leftarrow pC[i]$ { Computation of $p^t C_0$}
      for $j = 0$ to $(p^t - 1)$, $j \not\equiv 0 \mod p$ do
        $T[j + p^t i] \leftarrow T[j] + C[i]$ { Computation of $\mathcal{S}_{t+1} = \mathcal{S}_t + p^t C_0$}
      end for
    end for
  end for
  for $i = 1$ to $\frac{p-1}{2}$ do
    $C[i] \leftarrow pC[i]$ { we compute $p^{w-1}\mathcal{C}_0$}
    for $j = 0$ to $(p^{w-1} - 1)$, $j \not\equiv 0 \mod p$ and $j + p^{w-1} i \leq \frac{p^w-1}{2}$ do
      $T[j + p^{w-1} i] \leftarrow T[j] + C[i]$ { Computation of $\mathcal{S}_w$}
    end for
  end for
  return $T$
```

Since the number of computed points grows exponentially with $w$, the length of the window, and the cost of the precomputation could be not negligible. A straightforward method consists of computing the points of $\mathcal{S}_w$ by adding $P$ repetitively.

Here we will use a more refined method inspired by the method presented by Cohen *et al.* [14]. This method uses the multiplication by $p$ on the curve (instead of the curve doubling used in the original Cohen's method). Specifically, this method iteratively computes the sets

$$\mathcal{S}'_i = \left\{ iP, \text{ such that } 0 \le i < (p^i - 1) \text{ and } i \ne 0 \mod p \right\}.$$

This set is computed using the recursive relation $\mathcal{S}'_{i+1} = \mathcal{S}'_i + p^i \times \mathcal{C}_0$ where $\mathcal{C}_0 = \{iP, \text{ such that } 0 < i < (p-1)\}$. At the end we get $\mathcal{S}_w$ by computing the points $iQ$ in $\mathcal{S}'_{w-1} + p^{w-1}\mathcal{C}_0$ such that $i \le \frac{p^w-1}{2}$. The algorithm 3 uses this strategy to do the precomputations.

Let us evaluate the complexity of this algorithm. Since we do only one addition to compute each element of the table $T$, the complexity of this algorithm is equal to

$$p^{w-1}(\frac{p-1}{2})\text{Add} + ((w-2)(p-1) + \frac{p-1}{2})\text{Mult}_p. \tag{3}$$

When the points of $\mathcal{S}_w$ are expressed in affine coordinates, which is generally the best choice (cf [4]), each point addition requires one inversion, which is costly. In this situation, the algorithm 3 becomes really interisting by using Montgomery's algorithm (for example cf. [3], Algorithm 10.3.4) to do simultaneous inversions in the computation $\mathcal{S}'_i + (p^{i-1}\mathcal{C}_0)$. Specifically, Montgomery's algorithm replaces $m$ inversions with one inversion and $(3m-3)$ multiplications, so we will use this algorithm to simultaneously compute the inversions of $T[j] + C[i]$ for $j = 0, \ldots, p^{t-1}$.

## 5 Scalar Multiplication on Curves Defined Over the Field $\mathbb{F}_{3^n}$

Interest in the characteristic three case has grown recently, because of the existence of supersingular curves suitable for identity-based cryptosystems. Here we will deal with non-supersingular curves. We will establish tripling formulas in affine and Jacobian coordinates. We will then study scalar multiplication for curves $Y^2 = X^3 + aX^2 + b$ with *random* or *sparse* coefficients $a, b$.

Recently Smart and Westwood in [22], have studied the arithmetic of elliptic curves over characteristic three fields which are defined by this type of equation $Y^2 = X^3 + X^2 + 1$, i.e., with $a = 1$. They showed that these curves correspond to curves with points of order three, and are in correspondence with curves of Hessian form $Y^3 + X^3 = XYd$ with $d \in \mathbb{F}_{3^n}$. They established formulas for doubling in Hessian form which are cheap compared to doubling in Weierstrass form, and tripling formulas in Weierstrass form in a form close to (2).

We have slightly modified the tripling formulas of Smart and Westwood. We state below our formulas for every curve $Y^2 = X^3 + aX^2 + b$. But in the case $a = 1$ these formuas improve by one multiplication in Jacobian coordinates and two multiplications and one squaring in affine coordinates the cost of the formulas of Smart and Westwood.

### Tripling in Affine Coordinates

We begin with the tripling formula when the points are given in affine coordinates. Let $Q = (x, y)$ be a point on $E$, the coordinates of $Q_3 = 3Q$, where $Q_3 = (x_3, y_3)$, are given by $x_3 = y^6/(a^2(x^3+b)^2) - x^3a/(x^3+b)$, and $y_3 = y^9(a^3(x^3+b)^3) - y^3/(x^3+b)$. The details of the computation are given below.

**Table 1.** Tripling in affine coordinates

$$\begin{array}{c} x_3 = D^2 - B',\ (S) \\ y_3 = D^3 - B,\ \ (C) \end{array}$$

$$\text{with} \quad \begin{cases} X = x^3, & (C) \\ Y = y^3, & (C) \\ A = 1/(X+b), & (I) \end{cases} \text{and} \begin{cases} B = YA, & (M) \\ B' = aXA, & (2M) \\ D = B/a, & (M) \end{cases}$$

The cost of tripling in affine coordinates is equal to $3C + 4M + S + I$ where $C$ represents the cubing in $\mathbb{F}_{3^n}$, $M$ the multiplication, $S$ the squaring and $I$ the inversion.

## Tripling in Jacobian Coordinates

The Jacobian coordinates offer the cheapest tripling among coordinates which avoid inversion. Let $Q = (x, y, z)$ be a point on $E$ in Jacobian coordinates, we compute Jacobian coordinates of $3Q$, where $3Q = (x_3, y_3, z_3)$, as follows

**Table 2.** Tripling in Jacobian coordinates

$$\begin{array}{ll} x_3 = D_2 - aAF & (2M) \\ y_3 = D_1^3 - YF^2 & (M+S+C) \\ z_3 = F & \end{array}$$

$$\text{with} \quad \begin{cases} A = (xz)^3 & (M+C) \\ B = bz^9 & (M+2C) \\ D_1 = y^3/a & (M+C) \end{cases} \text{and} \begin{cases} D_2 = D^2 & (S) \\ F = A + B & \end{cases}$$

The cost of the tripling formula in Jacobian coordinates is equal to $6M + 2S + 5C$.

## On the Use of Sparse Coefficients

The tripling formulas in Tables 1 and 2 in affine and Jacobian coordinates, require several constant multiplications. Indeed, in the case of affine coordinates, we multiply by $a$ in the computation of $B'$ and by $\frac{1}{a}$ in the computation of $D$. For the Jacobian coordinates, we multiply by $b$ in $B$, by $\frac{1}{a}$ in $D$ and by $a$ in $x_3$.

If we choose the constants $a$ and $b$ of the curve equation with sparse representation in $\mathbb{F}_{3^n}$, the multiplication by such constants can be implemented efficiently. If we use the formulas in Tables 1 and 2 generally we cannot have $a$ and $\frac{1}{a}$ simultaneously sparse. In order to get a maximum sparse constant, we slightly modified the formulas to avoid multiplication by $\frac{1}{a}$.

For tripling in affine coordinates we propose the formulas in Table 9 in the appendix. The cost of tripling is equal to $3C + 5M + S + I$ when the constants are not sparse, but when $a, b$ and $a^2$ are sparse the cost of tripling becomes $3C + 2M + S + I$. For the Jacobian coordinates, we suggest the formulas in Table 10 in the appendix. The total cost of tripling is equal to $7M + 2S + 5C$ but when the constants are sparse the cost become $3M+2S+5C$. In both cases, the gain is not negligible.

## Scalar Multiplication

For now the elliptic curve $E$ is given by the equation $Y^2 = X^3 + aX^2 + b$ where $a$ and $b$ are random or sparse elements in $\mathbb{F}_{3^n}$. In Table 3 below we provide the cost of each elementary operation in each coordinate system in terms of cubing (C), squaring (S), multiplication (M), and inversion (I) in $\mathbb{F}_{3^n}$.

**Table 3.** Cost of the operations on the curve

| Operation | Coordinates system | Cost | |
|---|---|---|---|
| | | Random coeff. | Sparse coeff. |
| Addition | Affine | $S + 2M + I$ | $S + 2M + I$ |
| Mixed addition | Jacobian | $2C + 3S + 8M$ | $2C + 3S + 7M$ |
| Doubling | Affine | $S + 3M + I$ | $S + 2M + I$ |
| Doubling | Jacobian | $3C + 2S + 7M$ | $3C + 2S + 5M$ |
| Tripling | Affine | $3C + S + 4M + I$ | $3C + S + 3M + I$ |
| Tripling | Jacobian | $5C + 2S + 6M$ | $5C + 2S + 3M$ |

Now using the cost of each elementary operation we can compute the global cost of scalar multiplication. For simplification, we will suppose that the cost of squaring is roughly equal to the cost of multiplication, and that the cost of cubing is negligible compared to multiplication. We have computed the global cost of scalar multiplication in Table[1] 4 first using the Double-and-Add method (DA) with an integer $k$ which is given by its $NAF_w$ of length $\ell$. The precomputations are done using Cohen's method [14]. And we have also computed the cost of the scalar multiplication using Triple-and-Add method (TA) (Algorithm 2 with $p = 3$) with the integer $k$ given by its $SWR_{3,w}$ representation of the integer $k$ with length $\ell'$. The precomputations are done with Algorithm 3 and by using Mongomery's trick for inversions.

When $w = 1$, i.e., in the case of negligible precomputations, we remark that in general, the Triple-and-Add method is better than the Double-and-Add method

[1] The R or S corresponding to Coeff in the first column of the Table means R=Random,S=Sparse

**Table 4.** Cost of scalar multiplication with precomputations in $E(\mathbb{F}_{3^n})$

| Coord., Method, Key, Coeff. | Cost |
|---|---|
| Aff., DA, $NAF_w$, R. | $(\ell(4+\frac{3}{w+1})+3\cdot 2^{w-1}+4w-18)M+(\ell(1+\frac{1}{w+1})+w)I$ |
| Aff., DA, $NAF_w$, S. | $(3\ell(1+\frac{1}{w+1})+3\cdot 2^{w-1}+4w-15)M+(\ell(1+\frac{1}{w+1})+w)I$ |
| Jac., DA, $NAF_w$, R. | $(\ell(9+\frac{11}{w+1})+3\cdot 2^{w-1}+4w-23)M+(w+1)I$ |
| Jac., DA, $NAF_w$, S. | $(\ell(7+\frac{10}{w+1})+3\cdot 2^{w-1}+3w-19)M+(w+1)I$ |
| Aff., TA, $SWR_{3,w}$, R. | $(\ell\log_3(2)(5+\frac{3}{w+\frac{1}{2}})+2\cdot 3^w+10w-20)M$ $+(\ell\log_3(2)(1+\frac{1}{w+\frac{1}{2}})+2w-1)I$ |
| Aff., TA, $SWR_{3,w}$, S. | $(\ell\log_3(2)(4+\frac{3}{w+\frac{1}{2}})+2\cdot 3^w+8w-16)M$ $+(\ell\log_3(2)(1+\frac{1}{w+\frac{1}{2}})+2w-1)I$ |
| Jac., TA, $SWR_{3,w}$, R. | $(\ell\log_3(2)(8+\frac{11}{w+\frac{1}{2}})+2\cdot 3^w+10w-23)M+2wI$ |
| Jac., TA, $SWR_{3,w}$, S. | $(5\ell\log_3(2)(1+\frac{2}{w+\frac{1}{2}})+2\cdot 3^w+8w-17)M+2wI$ |

by about 20%, and that the use of sparse coefficients provides an important increase in speed using scalar multiplication in the Triple-and-Add method.

Let us study the practical case $\ell = 160$, and compute the best window size in each situation. To simplify we will suppose that the cost of inversion is roughly equal to $10M$. We obtain

- for the Triple-and-Add method, minimal computation is reached in Jacobian coordinates: it is equal to $1186M + 6I$ when $w = 3$ for random coefficients, and $855M + 6I$ when $w = 3$ for sparse coefficients;
- for the Double-and-Add method, minimal computation is reached in Jacobian coordinates: it is equal to $1779M + 6I$ when $w = 5$ for random coefficients, and $1531M + 6I$ when $w = 5$ for sparse coefficients.

In each situation, i.e., with sparse and random coeffcient, we note that Triple-and-Add method gives a real improvement in the cost of scalar multiplication. Specifically, the use of sparse coefficients provides an important increase in speed employing the Triple-and-Add method.

## 6 Scalar Multiplication on Ordinary Elliptic Curve Defined Over $\mathbb{F}_{5^n}$

Let $E$ be an elliptic curve defined by $Y^2 = X^3 + aX + b$, with $a, b \in \mathbb{F}_{5^n}$ and $Q = (x, y)$ a point on $E$. The equation (2) of section (3) has in this situation the following form

$$5Q = \left(\frac{P_x(x^5)}{P_z^2(x^5)}, \frac{y^5 P_y(x^5)}{P_z(x^5)}\right),$$

where $\deg P_x = 5, \deg P_y = 6$ and $\deg P_z = 2$. We modified this expression of $5Q$ to increase the efficiency of mulitplication by 5. We obtain the following formulas, first in the case of affine coordinates, and then in Jacobian coordinates.

**Table 5.** Multiplication by 5 in affine coordinates

$$\begin{array}{l} x_5 = 4M_xA/M_z^2, \ (S+2M+I) \\ y_5 = 2M_y/M_z^3, \qquad (2M) \end{array}$$

$$\begin{cases} X = x^5, & (Exp) \\ Y = y^5, & (Exp) \\ A = X + ab, & \\ B = x^2, & (S) \end{cases} \text{ and } \begin{cases} C = A^2, & (S) \\ E^\dagger = (B+a)^2 - xb + 2a^2, & (S+M) \\ F = X + 3ab, & \end{cases}$$

$$\text{and} \quad \begin{cases} M_x^\dagger = (B+2a)^{10} + 3abY^2, & (Exp+2S+M) \\ M_y^\dagger = YE^5(abF - a^5) + Y^5, & (2Exp+3M) \\ M_z^\dagger = aC + 3(a^3+3b^2)^2. & (M) \end{cases}$$

The total cost of the multiplication by 5 in affine coordinates is equal to $5Exp + 6S + 10M + I$, where $S$ represents squaring, $I$ the inversion, $M$ the multiplication, and $Exp$ the exponentiation to 5 in the field $\mathbb{F}_{5^n}$.

**Table 6.** Multiplication by 5 in Jacobian coordinates

$$\begin{array}{l} x_5 = -(C^{10} + 3DY^2)(X+D), \ (Exp+S+2M) \\ y_5 = 2(YF^5(DG - B^5) + Y^5), \quad (3Exp+3M) \\ z_5^\dagger = (aE + 3(a^3+3b^2)^2A^5)Z, \quad (Exp+3M) \end{array}$$

$$\text{with} \begin{cases} x^2, y^2, z^2, & (3S) \\ X = x^5, Y = y^5, & (2Exp) \\ Z = z^5, & (Exp) \\ A = z^4, & (S) \\ B^\dagger = az^4 & (M) \end{cases} \quad \textit{and} \quad \begin{cases} C = x^2 + 2B, & \\ D^\dagger = ab(z^2)^5, & (M) \\ E = (X+D)^2, & (S) \\ F = 2C^2 - xy^2, & (S+M) \\ G = X + 3D. & \end{cases}$$

The total cost of the multiplication by 5 in Jacobian coordinates of the point $Q$ is equal to $8Exp + 7S + 11M$.

### On the Use of Sparse Coefficients

As in the case of characteristic three, we remark that in the formulas above several constant multiplications are needed († indicates where such multiplications are required). Specifically, there are three constant multiplications in the Jacobian formula and four in the affine formula. We can exploit this fact by using sparse curve coefficients $a$ and $b$. This provides a way to speed-up the multiplication by 5 by decreasing the number of multiplications by three or four depending on the coordinate system used.

Consequently, we will divide the study of the complexity of scalar multiplication into two cases: 1) where $a$ and $b$ are random elements of $\mathbb{F}_{5^n}$, and 2) where $a$ and $b$ are sparse elements of $\mathbb{F}_{5^n}$.

## Complexity

In Table 7 below, we provide the cost of different operations on the curve in the affine, Jacobian and modified Jacobian coordinate systems (for additional details on this, see the paper by Cohen *et al.* [4]). The cost of each operation is given in terms of exponentiation to 5 ($Exp$), squaring ($S$), multiplication ($M$) and inversion ($I$) in $\mathbb{F}_{5^n}$. Now, using the cost of the elementary operations on

**Table 7.** Curve Operations

| Operation | Coordinate System | Cost | |
|---|---|---|---|
| | | Random coeff. | Sparse coeff. |
| Addition | Affine | $S + 2M + 1I$ | $S + 2M + I$ |
| Mixed Addition | Jacobian | $3S + 8M$ | $3S + 8M$ |
| Mixed Addition | Modified Jacobian | $5S + 9M$ | $5S + 8M$ |
| Doubling | Affine | $2S + 2M + 1I$ | $2S + 2M + I$ |
| Doubling | Jacobian | $6S + 4M$ | $5S + 4M$ |
| Doubling | Modified Jacobian | $4S + 4M$ | $4S + 4M$ |
| Multiply by 5 | Affine | $5Exp + 6S + 10M + I$ | $5Exp + 6S + 6M + I$ |
| Multiply by 5 | Jacobian | $8Exp + 7S + 11M$ | $8Exp + 7S + 8M$ |

curve $E$, we can determine the total cost of scalar multiplication on the curve. This means that for integer $k$ and point $Q$, we are going to determine the total cost to compute $kQ$. We study the two following situations.

First if we fix $k$ to be an $\ell$ bit length integer, using the Double-and-Add method (DA) with an integer $k$ given by its $NAF_w$ [8], the total cost of scalar multiplication is equal to $\ell$Double + $\frac{\ell}{w+1}$Add + *Precomputations*. The precomputations are done using the method given in [14].

Secondly, for the 5-Multiply-and-Add method (5-MA), if $\ell'$ is the length of the representation in basis 5 of $k = \sum_{i=0}^{\ell'-1} k_i 5^i$ the total cost of the 5-Multiply-and-Add method is equal to $\ell'$Mult$_5$ + $\left(\frac{\ell'}{w+\frac{1}{4}}\right)$ Add + *Precomputations*, where Mult$_5$ represents the cost of the multiplication by 5 and $w$ is the size of the window. We have $\ell' \cong \ell \frac{\log 2}{\log 5} = \ell \log_5(2)$.

If we suppose that the cost of a squaring is equal to the cost of multiplication and if we neglect the cost of exponentiation to 5 in the field $\mathbb{F}_{5^n}$, we obtain the overall cost of the sclalar multiplication in each coordinate system in Table[2] 8.

When $w = 1$, i.e., in the case of negligible precomputations, we remark that in general, the 5-Multiply-and-Add method is better than the Double-and-Add method by about 9%, and that the use of sparse coefficients provides an important increase in speed using scalar multiplication in the 5-Multiply-and-Add method.

[2] The R or S corresponding to Coeff in the first column of the Table means R=Random,S=Sparse.

**Table 8.** Cost of scalar multiplication in $E(\mathbb{F}_{5^n})$

| Coord.,Method,Key, Coeff. Type | Cost |
|---|---|
| Aff., DA, $NAF_w$, R. | $(\ell(4+\frac{3}{w+1})+3\cdot 2^{w-1}+4w-17)M$ $+(\ell(1+\frac{1}{w+1})+w)I$ |
| Aff., DA, $NAF_w$, S. | $(\ell(4+\frac{3}{w+1})+3\cdot 2^{w-1}+4w-17)M$ $+(\ell(1+\frac{1}{w+1})+w)I$ |
| Mod. Jac., DA, $NAF_w$, R. | $(\ell(8+\frac{14}{w+1})+3\cdot 2^{w-1}+4w-21)M+(w+1)I$ |
| Mod. Jac., DA, $NAF_w$, S. | $(\ell(8+\frac{13}{w+1})+3\cdot 2^{w-1}+4w-21)M+(w+1)I$ |
| Aff., 5-MA, $SWR_{5,w}$, R. | $(\ell\log_5(2)(16+\frac{3}{w+\frac{1}{4}})+5^{w-1}\cdot 12+64w-114)M$ $+(\ell\log_5(2)(1+\frac{1}{w+\frac{1}{4}})+4w-2)I$ |
| Aff., 5-MA, $SWR_{5,w}$, S. | $(\ell\log_5(2)(12+\frac{3}{w+\frac{1}{4}})+5^{w-1}\cdot 12+48w-86)M$ $+(\ell\log_5(2)(1+\frac{1}{w+\frac{1}{4}})+4w-2)I$ |
| Jac., 5-MA, $SWR_{5,w}$, R. | $(\ell\log_5(2)(18+\frac{11}{w+\frac{1}{4}})+5^{w-1}\cdot 12+64w-116)M$ $+(4w-1)I$ |
| Jac., 5-MA, $SWR_{5,w}$, S. | $(\ell\log_5(2)(15+\frac{11}{w+\frac{1}{4}})+5^{w-1}\cdot 12+48w-89)M$ $+(4w-1)I$ |

If we study the practical situation $\ell = 160$ we remark that, under the condition that the cost of inversion is of $10M$ (cf. for example [10])

- for the 5-Multiply-and-Add method, minimal computation is reached in jacobian coordinates: it is equal to $1650M + 7I$ when $w = 2$ for random coefficients, and $1185M + I$ when $w = 1$ for sparse coefficients;
- for the Double-and-Add method, minimal computation is reached in jacobian coordinates: it is equal to $1705M + 5I$ when $w = 5$ for random coefficients, and $1674M + 5I$ when $w = 5$ for sparse coefficients.

Thus in the case of random coefficients, we note that 5-Multiply-and-Add method provides only a small gain compared to the Double-and-Add method. Conversely, the use of sparse coefficients provides an important increase in speed employing the 5-Multiply-and-Add method when $w = 2$. This clearly justifies the use of scalar multiplication on the curve in these methods.

## 7 Conclusion

We have presented a new way to multiply a point $Q$ by 3 and 5 on curves defined over a field of characteristic 3 and 5 respectively. We used a $p$-Multiply-and-Add method to do scalar multiplication on the curve. We saw that this approach provides a scalar multiplication method which is less costly than the classic method of Double-and-Add.

Theoretically this method extends for bigger characteristics $p \geq 7$. But in these cases we did not obtain multiplication by $p$ as efficiently as necessary due to the high degree of morphism $\phi$ defined in equation (1).

## References

1. D.V. Bailey and C. Paar. Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms. In *Advances in cryptology, CRYPTO'98, 6th International Conference, (AAECC-6)*, volume 1462, pages 472–485. Springer-Verlag, 1998.
2. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *Lecture Notes in Computer Science*, 2139, 2001.
3. H. Cohen. *A cours in computational algebraic number theory*, volume 138. Springer-Verlag, 1993.
4. H. Cohen, A. Miyaji, and T. Ono. Efficient Elliptic Curve Exponentiation Using Mixed Coordinates. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT'98: International Conference on the Theory and Application of Cryptology*, volume 1514 of *Lecture Note in Computer Science*, pages 51–65. Springer-Verlag, 1998.
5. C. Diem. The GHS-attack in odd characteristic. *Ramanujan Math. Soc.*, 18:1–32, 2003.
6. W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 24:644–654, 1976.
7. T. El Gamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transaction on Information Theory*, IT-31:469–472, 1985.
8. D. Hankerson, J. López Hernandez, and A. Menezes. Software Implementation of Elliptic Curve Cryptography over Binary Fields. In *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 1965 of *Lecture Notes in Computer Science*, 2001.
9. K. Harrison, D. Page, and N. P. Smart. Software implementation of finite fields of characteristic three. *LMS Journal of Computations and Mathematics*, 5:181–193, 2002.
10. T. Itoh and S. Tsujii. A fast algorithm for computing multiplicative inverses in GF($2^m$) using normal bases. *Information and Computation*, 78(3):171–177, 1988.
11. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.
12. M. Brown and D. Hankerson and J. López and A. Menezes. Software Implementation of the NIST Elliptic Curves Over Prime Fields. *Topics in Cryptology - CT - RSA*, 2020, 2001.
13. V. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology, proceeding's of CRYPTO'85*, volume 218 of *Lecture Note in Computer Science*, pages 417–426. Springer-Verlag, 1986.
14. A. Miyaji, T. Ono, and H. Cohen. Efficient Elliptic Curve Exponentiation. In *Proceedings of the first International Conference on Information and Communication Security*, pages 282–290. Springer-Verlag, 1997.
15. R.C. Mullin, I.M. Onyszchuk, S.A. Vanstone, and R.M. Wilson. Optimal Normal Bases in GF($p^n$). *Discrete Applied Mathematics*, 22(2):149–161, 1989.
16. Dan Page and Nigel P. Smart. Hardware implementation of finite fields of characteristic three. In *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2002*, volume 2523, pages 529–539. Springer-Verlag, 2003.
17. S.C. Pohlig and M.E. Hellman. An improved algorithm for computing logaritms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.
18. J.M. Pollard. Monte carlo methods for index computation (mod p). *Mathematics of Computation*, 32:918–924, 1978.

19. L. Adleman R.L. Rivest, A. Shamir. A method for obtaining digital signature and public key cryptosystem. *Comm. ACM*, 21:120–126, 1978.
20. J.H. Silverman. *The Arithmetic of Elliptic Curves*, volume GTM 106. Springer-Verlag, 1986.
21. N.P. Smart. Elliptic curves over small fields of odd characteristic. *J. Cryptology*, 12(2):141–151, August 1999.
22. N.P. Smart and E.J. Westwood. Point Multiplication on Ordinary Elliptic Curves over Fields of Characteristic Three. *Applicable Algebra in Engineering, Communication and Computing*, 13(6), 2003.

## A Tripling Formulas with Sparse Coefficients

We provide below modified formulas for tripling in affine and Jacobian coordinates when the elliptic curve has sparse coefficients $a, b$.

**Table 9.** Tripling in affine coordinates - sparse coefficients

$$\begin{array}{l} x_3 = D^2 - B', \ (S) \\ y_3 = D^3 - B, \ (C) \end{array} \text{with} \left\{ \begin{array}{lll} X = x^3, & & (C) \\ Y = y^3, & & (C) \\ A^\dagger = \frac{1}{a(X+b)}, & & (M+I) \end{array} \right. \text{and} \left\{ \begin{array}{ll} D = YA & (M) \\ B'^\dagger = a^2XA, & (2M) \\ B^\dagger = aD, & (M) \end{array} \right.$$

**Table 10.** Tripling in Jacobian coordinates - sparse coefficients

$$\begin{array}{ll} x_3^\dagger = D_2 - a^3AF & (2M) \\ y_3^\dagger = D_1^3 - aYG^2 & (2M+S+C) \\ z_3 = G & \end{array}$$

$$\text{with} \left\{ \begin{array}{ll} A = (xz)^3 & (M+C) \\ B^\dagger = bz^9 & (M+2C) \\ D_1 = y^3 & (C) \end{array} \right. \text{and} \left\{ \begin{array}{ll} D_2 = D^2 & (S) \\ F = A+B & \\ G^\dagger = aF & (M) \end{array} \right.$$

In each case we put the mark † when constant multiplication are required.

# SCA Resistant Parallel Explicit Formula for Addition and Doubling of Divisors in the Jacobian of Hyperelliptic Curves of Genus 2

Tanja Lange[1,⋆] and Pradeep Kumar Mishra[2]

[1] Department of Mathematics, Technical University of Denmark,
Matematiktorvet – 303, DK-2800 Kgs. Lyngby, Denmark
`t.lange@mat.dtu.dk`

[2] Centre for Information Security and Cryptography (CISaC),
University of Calgary, Calgary, AB, Canada T2N 1N4
`pradeep@math.ucalgary.ca`

**Abstract.** Hyperelliptic curve cryptosystems (HECC) can be implemented on a variety of computing devices, starting from smart cards to high end workstations. Side-channel attacks are one of the most potential threats against low genus HECC. Thus efficient algorithms resistant against side channel attacks are the need of the hour. In the current work we provide implementation ready formulae for addition and doubling on curves of genus 2 which are shielded against simple side-channel analysis by having a uniform performance. This is achieved by applying the concept of side-channel atomicity – introducing cheap dummy operations to make all traces look identical.

So far a detailed study of countermeasures against side-channel attacks exists only for differential attacks. There one assumes that the performance is made predictable by other means. But apart from the double-and-alway-add approach only generalizations of the Montgomery form were suggested and only for odd characteristic. They are less efficient and do not combine well with some of the countermeasures against differential attacks. Hence, our contribution closes the gap to achieve secured implementations of HECC on devices exposed to side-channel attacks.

To increase the performance even further we show how our formulae can be implemented in parallel on two multipliers using a low number of registers. It is also possible to combine our method with precomputations.

**Keywords:** Side-channel attacks, SPA countermeasures, hyperelliptic curve cryptosystems, scalar multiplication, side-channel atomicity, parallel algorithms.

⋆ The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

© Springer-Verlag Berlin Heidelberg 2005

## 1 Introduction

Hyperelliptic curve cryptosystems (HECC) are gradually gaining popularity due to their versatility. For general curves of low genus there is no subexponential algorithm known to solve the hyperelliptic curve discrete logarithm problem (HCDLP). So, like ECC, HECC provides a high level of security with much smaller keys and smaller operands compared to RSA. This makes curve based systems very interesting for resource constrained devices like smart cards and mobile devices. Since these are generally used in hostile outdoor environments, implementations of HECC must be particularly robust.

Although for suitably chosen system parameters the DLP on HEC is hard to compute, a straightforward implementation is vulnerable to side-channel attacks (SCA). These attacks recreate the secret key of a user by sampling side-channel information like timing, power consumption and electromagnetic radiation traces of the computations.

The usual scenario in elliptic and hyperelliptic curve discrete logarithm based cryptosystems is that the scalar multiplication is composed from a sequence of additions and doublings and that these group operations differ from the side-channel information. If the secret is established using a single trace, e. g. by observing the sequence of group additions and doublings which are linked directly to the bits of the key, one speaks of *simple side-channel analysis (SSCA).* To avoid SSCA a possible but ineffective solution is to use a fixed pattern of group additions and doublings which is filled with dummy operations if needed.

If the implementation is shielded against SSCA an attacker could still gain information from the side channel data by using measurements for different inputs and fixed secret scalar. Often the performance depends on the intermediate results and an attacker could use his own device to simulate the computations and look for correlations among the observed traces. One uses the term *differential side-channel analysis (DSCA)* if the attack uses information from different inputs and statistics among them. As DSCA makes use of the knowledge of the internal state the approach is to randomize the internal representation such that the attacker cannot simulate the state and thus cannot group the observed side-channel traces.

For elliptic curves, finding efficient countermeasures against both types of attacks has been an active research area during the past years starting from [8], such that now countermeasures can be applied with only little performance loss. For hyperelliptic curves, Avanzi [2] has generalized the countermeasures against DSCA. His approach assumes the implementation to be secured against SSCA by other means like the double-and-always-add-method. However, there is still the lack of an efficient SSCA secured implementation and with the present paper we aim at filling this gap. [1] We give at hand the complete formulas for an efficient SSCA shielded implementation such that now it is possible to have a fully secured device running HECC.

[1] We like to point out that this problem is not solved by applying Cantor's algorithm as also there the executions differ depending on the input.

To this end we consider the proposal made by Chevallier-Mames, Ciet and Joye in [7] which has been proved to be the most cost effective one in case of ECC. The computational overhead involved in this countermeasure is almost negligible introducing only a few further field additions and no multiplications. The method divides the point addition and doubling algorithms into small parts, called side-channel atomic blocks, which are indistinguishable from the side-channel as they contain always the same sequence of operations. This is based on the assumption that real and dummy operations cannot be distinguished. When the scalar multiplication is computed with these addition and doubling algorithms, it becomes a computation of a series of atomic blocks and the side-channel information becomes uniform. This difference is visualized in the following two pictures: Figure 1 shows the view of the attacker who only sees a uniform sequence of operations. In Figure 2 one sees that the first three steps belonged to an addition while the following ones show the beginning of a doubling.

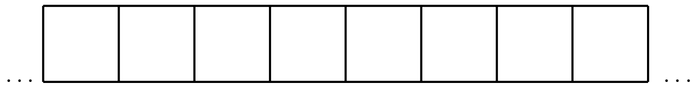


**Fig. 1.** Attacker's view on algorithm

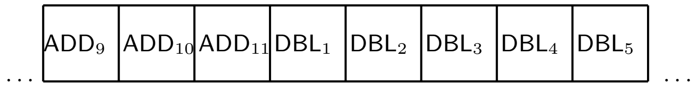


**Fig. 2.** Actual operations inside blocks

In the current work we propose for the first time *hyperelliptic curve* divisor addition and doubling formulae divided into side-channel atomic blocks. The existing formulae [17] for arithmetic on hyperelliptic curves of genus 2 form the basis for this work but to combine them with side-channel atomicity some more considerations are needed as e. g. one needs to decide upon the content of the atomic blocks and arrange the formulas in a suitable manner. In even characteristic most implementations of finite field arithmetic support inversions and thus group operations in affine coordinates are faster than in inversion-free systems. Still, inversions are the most costly field operations and they do not fit in the small atomic block setting. Additionally, over binary fields, squarings are far less expensive than multiplications. Our strategy was to take the whole group operation as one atomic block which forced a rewriting of addition and doubling. This is the same solution as done for elliptic curves in in even characteristic by [7] – but our blocks are much more involved and larger.

As a second topic we show that our algorithms allow for an efficient implementation in parallel if the system admits two field element multipliers, each capable of carrying out a multiplication. We state the formula for odd characteristic in a manner such that the blocks with odd index can be performed by one processor and those with even index by another. For even characteristic we have prescribed the order in which the operations can be computed in parallel.

For implementations it is important to note that the curves are given in the standard form. This allows the DSCA countermeasures of divisor and curve randomization to be applied directly. Furthermore, windowing methods can be used to increase the speed. Obviously, our method leaks the Hamming weight of the scalar. This information can be made uniform by applying the windowing method proposed by Möller [21] (see also [13] for the application to ECC).

Compared to the recently proposed generalizations of Montgomery arithmetic to genus two curves [9, 16], our formulae obtain a much higher speed with the additional option of parallel execution and the advantage of windowing methods. Namely, Duquesne's formulae need 69 multiplications per bit while in odd characteristic our approach needs 55 multiplications per bit of the scalar on average assuming a NAF expansion of the scalar. Additionally, our algorithm allows extra speed-up from windowing methods and precomputations and works for every curve while [9] needs a divisor of order 2. The formulae in [16] are even slower but more general. Very recently Gaudry [10] suggested much more efficient Montgomery-like formulae for the case that the curve has full 2-torsion.

Our algorithms are the fastest for general HECC achieving SPA resistance and allowing for DPA resistance.

In the following we briefly outline the background and then study separately the cases of odd and even characteristic both times investigating parallel implementations. Finally we discuss memory requirements.

## 2 Background

In this section we will briefly describe hyperelliptic curve cryptosystems and side-channel atomicity, the most efficient countermeasure against simple power attacks.

### 2.1 Hyperelliptic Curve Cryptosystems

This section mainly serves to fix the notation. For the background of hyperelliptic curve cryptography we refer to the literature e. g. [1, 4, 5, 14, 15, 19].

We concentrate on curves of genus 2 over a finite field $\mathbb{F}_q$ given by an equation of the form $C : y^2+h(x)y = f(x)$, where $h(x), f(x) \in \mathbb{F}_q[x], \deg(f) = 5, \deg(h) \leq 2$ and $f$ is monic. Furthermore, we require that there is no point on $C$ over the algebraic closure which is a common solution to both partial derivatives. The group one uses is the divisor class group of $C$ and the group elements can be represented by 2 polynomials $[u(x), v(x)]$ satisfying $u, v \in K[x], \deg v < \deg u \leq 2$, $u$ monic and $u \mid v^2 + hv - f$.

Cantor's algorithm [6, 15] describes the arithmetic using these representatives and polynomial arithmetic. The first attempt to compute divisor addition by explicit formulae was made by Spallek [23]. Harley [12] largely improved the running time and Gaudry and Harley used such formulae in their record on point counting [11]. Later many researchers gave detailed studies for various genera and different finite fields. An overview of most proposals can be found e. g. in [20]. For general curves of genus 2, the explicit formulae proposed by Lange are considered the most efficient ones. The latest version with an extensive comparison of coordinate systems is available in [17]. We refer to that paper for further details and notation. To measure the efficiency we count the number of field operations needed; we use $[i], [m], [s], [a]$, and $[n]$ to denote an inversion, a multiplication, a squaring, an addition or a field negation respectively. In general we refer to the coefficient of $x^i$ in a polynomial $k(x)$ as $k_i$, i. e. $k(x) = \sum k_i x^i$.

### 2.2 Side-Channel Atomicity

The formulae in the following sections are side-channel indistinguishable under certain assumptions on the indistinguishability of the finite field operations which we state here for further reference.

1. In a finite field of large characteristic all multiplications are indistinguishable. In particular, this includes the case of squaring. This is usually satisfied provided that the squaring is carried out by the same hardware used for multiplication. Furthermore, it is also possible to use the concept of atomicity on a lower level and design the field operations to be side-channel independent. In even characteristic we assume this property only if both inputs to the multiplication are different, i. e. unless squarings are involved since they are usually much faster and thus easy to identify.
2. Any additions are indistinguishable, similarly all negations have the same side-channel traces.
3. Table-look ups do not reveal the storage address, access to the same element cannot be distinguished from access to a different element.

We would like to point out that we are aware that these assumptions are not automatically satisfied but need to be enabled by a careful implementation of the underlying operations.

## 3 Our Methodology

First of all we mention here that the task at hand is very much implementation dependent and that our assumptions might not hold for all environments. Furthermore, we need to consider separately the cases of odd and even characteristic.

### 3.1 Odd Characteristic

For odd characteristic one can always achieve $h(x) = 0$ and $f_4 = 0$ by a linear transformation of the coordinates, for the latter the characteristic needs to differ from 5. We deal with the new coordinates (see [17]) as they are most efficient and assume the input to be affine, therefore we can use mixed coordinates. An intermediate result is represented by 8 coordinates $[U_0, U_1, V_0, V_1, Z_1, Z_2, z_1, z_2]$ with the correspondence $u_i = U_i/Z_1^2$ and $v_i = V_1/(Z_1^3 Z_2)$ to affine coordinates. The additional entries are used to accelerate the computation and satisfy $z_1 = Z_1^2$, $z_2 = Z_2^2$.

Our design can be combined with the DSCA countermeasures proposed for HECC [2]. The easiest one is divisor randomization: $Z_1$ and $Z_2$ are multiplied with random field elements $r_1$ and $r_2$. The new representation of *the same group element* is given by $[U_0 r_1^2, U_1 r_1^2, V_0 r_1^3 r_2, V_1 r_1^3 r_2, Z_1 r_1, Z_2 r_2, z_1 r_1^2, z_2 r_2^2]$. One update takes $2[s], 10[m]$, thus far less than one full group operation, and should be performed several times during a scalar multiplication. To avoid attacks similar to [22] the final result should be made affine or at least be randomized before output. As this countermeasure is not effective against Goubin type attacks, Avanzi suggests to add a divisor before the scalar multiplication for which the respective multiple is known and which can be removed from the result. For this a small table is sufficient which is updated during the execution. This countermeasure can be combined with our approach.

Curve randomization is usually not suggested in odd characteristic as one either needs to use a non-trivial $h(x)$ leading to worse performance or do not prevent Goubin type attacks as the $V_i$ are multiplied by a constant only while not gaining any performance advantage over the divisor randomization method.

### 3.2 Even Characteristic

Most implementations of characteristic 2 arithmetic allow to compute inversions rather efficiently, hence, for this case we stick to affine coordinates. The inversion free systems trade one inversion per group operation for several multiplication, e. g. the cost for an addition with inversion is $1[i] + 22[m] + 3[s]$ and the cost without inversion is $38[m] + 6[s]$.

For doublings the fastest explicit formulae can be found in [18] but they depend heavily on the curve coefficients. Since we want to prevent DSCA as well, our approach should allow for curve randomization (cf.[2]) as divisor randomization cannot be used with affine input. Therefore, we assume a general curve equation requiring only $h_2 \in \{0, 1\}$ which can always be achieved. Hence, the multiplications with $h_2$ are not counted.

The transformations are induced by the changes of variables $y \mapsto y' + ax' + b$ and $x \mapsto x + c$. Due to the constant terms, Goubin type attacks are avoided. The coefficients $h_2$ and $h_1$ remain fixed which is not a problem as at least one of them is nonzero. To keep enough randomness we need to allow nonzero $f_4$.

## 4 Formulas for Even Characteristic

The complexities of the divisor addition and doubling for curves over fields of even characteristic are $1[i] + 22[m] + 3[s]$ and $1[i] + 23[m] + 5[s]$ respectively besides some additions each. Here we take in to account the extra costs due to multiplications with $f_4$. However, the atomic blocks corresponding to addition and doubling must match perfectly, i.e. they must have the same operations at the corresponding positions and we distinguish between $[i], [m], [s]$, and $[a]$.

The inversions must be placed at the same locations in both addition and doubling. Let us divide each block into two parts each. We name the part before the inversion in addition to be $A_1$ and that following inversion to be $A_2$. Similar parts for doubling are named as $D_1$ and $D_2$ respectively. For the atomic blocks to be indistinguishable from the side-channel, the costs of $A_1$ and $D_1$ must be equal. Also, the costs of $A_2$ and $D_2$ must be equal. Looking at the explicit formula in [17] we find that $cost(A_1) = 9[m] + 1[s]$ and $cost(D_1) = 11[m] + 2[s]$. So a natural way of attaining $cost(A_1) = cost(D_1)$ is to add 2 dummy multiplications and 1 dummy squaring in $A_1$. Furthermore, to allow parallel execution we try to group the multiplications in tuples.

But we observe that the inversion step in doubling needs only $r$ and $s_1$ (see [17] as reference for the variables). Hence, the computation of $s_0$ can be postponed to a later stage. The computation of $s_0$ needs one multiplication which can be carried out after the inversion. By doing so we can save one dummy multiplication in $A_1$.

Now, the costs of the parts $A_2$ and $D_2$ are $13[m] + 2[s]$ and $13[m] + 3[s]$, respectively. Hence, introducing one dummy squaring to $A_2$ we can make their costs equal. In Table 1 we show HCADD and HCDBL algorithms for even characteristic as one atomic block each. We denote dummy additions, squarings and multiplications by the notations $*[a]$, $*[s]$, $*[m]$ respectively. These entries also show which register is used as write location when executing the dummy operation.

Note that both operations HCADD and HCDBL now have the same number and same order of field operations. Hence, they are indistinguishable by SSCA. This way they can be safely used in any binary or windowing algorithm for computing the scalar multiplication.

In the table some expressions involving $h_2T_k$ can be found; as $h_2 \in \{0, 1\}$, this either means 0 or $T_k$. Additions involving the curve coefficients might turn out to be dummy additions if the respective coefficient is 0. If $h_1 = 0$ or $f_4 = 0$, some steps can be skipped completely.

## 4.1 Parallel Implementation

A system having two multipliers and two adders and hardware for other field operations (one each) can implement our algorithms. Design of one such hyperelliptic parallel coprocessor has been recently proposed in [3]. The same architecture except for the scheduler will work nicely for our scheme, too. As field addition is an inexpensive operation in comparison to multiplication, it is possible to use only one adder to minimize the chip area. Our algorithm seems to require some more registers. But as the register size is half the size of the ones used for ECC, the memory requirement is not as high as it seems from the register count. We consider the memory requirement in Section 6.

Field addition is a very cheap operation, thus we propose that it should be carried out by a single adder sequentially. In a multiplication round we use two

**Table 1.** HCDBL and HCADD Algorithms as One Atomic Block, even char

| | **Algorithm HCADD** | **Algorithm HCDBL** |
|---|---|---|
| *In:* | $D_1 = [u_{10}, u_{11}, v_{10}, v_{11}]$ $D_2 = [u_{20}, u_{21}, v_{20}, v_{21}]$ | $D = [u_0, u_1, v_0, v_1]$ |
| *Out:* | $D_1 + D_2 = [u'_0, u'_1, v'_0, v'_1]$ | $2D = [u'_0, u'_1, v'_0, v'_1]$ |
| *Init:* | $T_1 \leftarrow u_{10}, T_2 \leftarrow u_{11}$ $T_3 \leftarrow v_{10}, T_4 \leftarrow v_{11},$ $T_5 \leftarrow u_{20}, T_6 \leftarrow u_{21},$ $T_7 \leftarrow v_{20}, T_8 \leftarrow v_{21}$ | $T_1 \leftarrow u_0, T_2 \leftarrow u_1$ $T_3 \leftarrow v_0, T_4 \leftarrow v_1$ |
| 1 | $T_9 = T_2 + T_6\ (inv_1)$ | $T_5 = h_1 + h_2T_2\ (\tilde{v}_1)$ |
| 2 | $T_{10} = T_1 + T_5\ (z_2)$ | $T_7 = h_0 + h_2T_1\ (\tilde{v}_0)$ |
| 3 | $T_{11} = T_9 * T_9$ | $T_6 = T_2 * T_2\ (w_1)$ |
| 4 | $*[s],\ T_{12}$ | $T_8 = T_5 * T_5\ (w_2)$ |
| 5 | $T_{13} = T_2 * T_9$ | $T_9 = T_2 * T_5\ (w_3)$ |
| 6 | $T_{11} = T_1 * T_{11}$ | $T_8 = T_1 * T_8$ |
| 7 | $T_{12} = T_{10} + T_{13}\ (inv_0)$ | $T_9 = T_7 + T_9\ (inv_0)$ |
| 8 | $T_{13} = T_{10} * T_{12}$ | $T_7 = T_7 * T_9$ |
| 9 | $*[m],\ T_{13}$ | $T_{12} = T_2 * f_4$ |
| 10 | $T_{10} = T_{11} + T_{13}\ (r)$ | $T_7 = T_8 + T_7\ (r)$ |
| 11 | $T_3 = T_3 + T_7\ (w_0)$ | $T_8 = f_3 + T_6\ (w_3)$ |
| 12 | $T_4 = T_4 + T_8\ (w_1)$ | $T_{10} = T_8 + h_2T_4\ (k'_1)$ |
| 13 | $T_{11} = T_9 + T_{12}$ | $T_8 = h_2T_4 + T_8$ |
| 14 | $*[a],\ T_{13}$ | $T_8 = T_8 + T_{12}$ |
| 15 | $T_{13} = T_3 + T_4$ | $T_6 = f_2 + h_2T_3$ |
| 16 | $T_{14} = 1 + T_2$ | $T_{11} = h_1 + T_4$ |
| 17 | $T_3 = T_3 * T_{12}\ (w_2)$ | $T_8 = T_2 * T_8$ |
| 18 | $T_4 = T_4 * T_9\ (w_3)$ | $T_{11} = T_4 * T_{11}$ |
| 19 | $*[a],\ T_{12}$ | $T_8 = T_8 + T_{11}$ |
| 20 | $*[a],\ T_{12}$ | $T_8 = T_6 + T_8\ (k'_0)$ |
| 21 | $T_{12} = T_{11} * T_{13}$ | $T_{11} = T_8 * T_9\ (w_0)$ |
| 22 | $T_{14} = T_4 * T_{14}$ | $T_{12} = T_{10} * T_5\ (w_1)$ |
| 23 | $T_{12} = T_3 + T_{12}$ | $T_5 = T_5 + T_9$ |
| 24 | $T_{12} = T_{12} + T_{14}\ (s'_1)$ | $T_8 = T_8 + T_{10}$ |
| 25 | $*[a],\ T_{13}$ | $T_6 = 1 + T_2$ |
| 26 | $T_4 = T_1 * T_4$ | $T_5 = T_5 * T_8$ |
| 27 | $T_{13} = T_{10} * T_{12}$ | $T_6 = T_{12} * T_6$ |
| 28 | $T_4 = T_3 + T_4\ (s'_0)$ | $T_5 = T_5 + T_{11}$ |
| 29 | $*[a],\ T_3$ | $T_5 = T_5 + T_6\ (s'_1)$ |
| 30 | $*[m],\ T_3$ | $T_8 = T_7 * T_5$ |
| 31 | $T_{12} = T_{12} * T_{12}$ | $T_5 = T_5 * T_5$ |
| 32 | $T_3 = 1/T_{13}\ (w_1)$ | $T_8 = 1/T_8\ (w_1)$ |
| 33 | $T_{13} = T_3 * T_{10}\ (w_2)$ | $T_9 = T_7 * T_8\ (w_2)$ |
| 34 | $T_{14} = T_3 * T_{12}\ (w_3)$ | $T_5 = T_5 * T_8\ (w_3)$ |
| 35 | $T_{10} = T_{10} * T_{13}\ (w_4)$ | $T_7 = T_7 * T_9\ (w_4)$ |
| 36 | $T_4 = T_4 * T_{13}\ (s''_0)$ | $T_6 = T_1 * T_{12}$ |
| 37 | $T_{13} = T_4 + T_6\ (l'_2)$ | $T_6 = T_{11} + T_6\ (s'_0)$ |
| 38 | $*[s],\ T_{11}$ | $T_8 = T_7 * T_7\ (w_5)$ |
| 39 | $T_3 = T_4 * T_6$ | $T_6 = T_6 * T_9\ (s''_0)$ |
| 40 | $T_{12} = T_{10} * T_{10}\ (w_5)$ | $T_{11} = T_6 * T_6$ |
| 41 | $T_3 = T_3 + T_5\ (l'_1)$ | $T_9 = T_2 + T_6\ (l'_2)$ |
| 42 | $T_{11} = T_4 * T_5\ (l'_0)$ | $T_{10} = T_2 * T_6$ |
| 43 | $T_2 = T_2 + T_4$ | $T_{10} = T_{10} + T_1\ (l'_1)$ |
| 44 | $T_4 = T_4 + T_9$ | $T_2 = T_6 + T_2$ |
| 45 | $T_4 = T_4 + h_2T_{10}$ | $T_2 = h_2T_2 + h_1$ |
| 46 | $T_2 = T_9 + h_2T_{10}$ | $*[a],\ T_{12}$ |
| 47 | $T_4 = T_2 * T_4$ | $T_6 = T_1 * T_6\ (l'_0)$ |
| 48 | $T_{10} = T_{10} * h_1$ | $T_{12} = T_8 * f_4$ |
| 49 | $T_9 = T_9 + f_4$ | $T_{11} = T_{11} + T_{12}$ |
| 50 | $T_9 = T_9 * T_{12}$ | $T_2 = T_2 * T_7$ |
| 51 | $T_4 = T_1 + T_4$ | $T_1 = T_2 + T_{11}\ (u'_0)$ |
| 52 | $T_4 = T_4 + T_3$ | $T_2 = h_2T_7 + T_8\ (u'_1)$ |
| 53 | $T_2 = T_2 + T_{12}\ (u'_1)$ | $T_7 = T_6 + T_9\ (w_1)$ |
| 54 | $T_4 = T_4 + T_{10}$ | $*[a],\ T_8$ |
| 55 | $T_1 = T_4 + T_9\ (u'_0)$ | $*[a],\ T_8$ |
| 56 | $T_{10} = T_2 + T_{13}\ (w_1)$ | $*[a],\ T_8$ |
| 57 | $T_9 = T_2 * T_{10}$ | $T_8 = T_2 * T_7$ |
| 58 | $T_9 = T_1 + T_9$ | $T_8 = T_8 + T_1$ |
| 59 | $T_9 = T_9 + T_3\ (w_2)$ | $T_8 = T_8 + T_{10}\ (w_2)$ |
| 60 | $T_9 = T_{14} * T_9$ | $T_8 = T_8 * T_5$ |
| 61 | $T_9 = T_8 + T_9$ | $T_8 = T_8 + T_4$ |
| 62 | $T_9 = h_1 + T_9$ | $T_8 = T_8 + h_1$ |
| 63 | $T_4 = h_2T_2 + T_9\ (v'_1)$ | $T_4 = T_8 + h_2T_2\ \ (v'_1)$ |
| 64 | $T_9 = T_1 * T_{10}$ | $T_9 = T_1 * T_7$ |
| 65 | $T_9 = T_9 + T_{11}\ (w_2)$ | $T_7 = T_6 + T_9\ (w_2)$ |
| 66 | $T_9 = T_{14} * T_9$ | $T_5 = T_7 * T_5$ |
| 67 | $T_9 = T_7 + T_9$ | $T_5 = T_5 + T_3$ |
| 68 | $T_9 = h_0 + T_9$ | $T_7 = T_5 + h_0$ |
| 69 | $T_3 = h_2T_1 + T_9\ (v'_0)$ | $T_3 = T_7 + h_2T_1\ (v'_1)$ |
| | | |
| | $u'_0 \leftarrow T_1, u'_1 \leftarrow T_2,$ $v'_0 \leftarrow T_3, v'_1 \leftarrow T_4$ | $u'_0 \leftarrow T_1, u'_1 \leftarrow T_2,$ $v'_0 \leftarrow T_3, v'_1 \leftarrow T_4$ |

multipliers $M_1$ and $M_2$. The operations given in Table 1 are already grouped in a way to allow two multipliers. Starting from the top every two adjacent multiplications can be executed in parallel. This ordering takes into account that not both processors try to write to the same location at the same time and that not one processor reads a register which is changed in the same step by the other processor.

## 5 Formulas for Odd Characteristic

For explicit formulae for curves over fields of odd characteristic, mixed addition in new coordinates involves $35[m]+6[s]$ and doubling involves $34[m]+7[s]$. In the

**Table 2.** HCDBL Algorithm in Atomic Blocks, odd char

**Algorithm HCDBL**

*Input:* $D = (U_0, U_1, V_0, V_1, Z_1, Z_2, z_1, z_2)$
*Out:* $2D = (U_0', U_1', V_0', V_1', Z_1', Z_2', z_1', z_2')$
*Init:* $T_1 = U_0, T_2 = U_1, T_3 = V_0, T_4 = V_1,$
$T_5 = Z_1, T_6 = Z_2, T_7 = z_1, T_8 = z_2$

| | | | |
|---|---|---|---|
| $\Gamma_1$ | $T_9 = T_4 * T_4$<br>*<br>*<br>* | $\Gamma_2$ | $T_{10} = T_2 * T_4$<br>*<br>$T_{10} = -T_{10}$<br>* |
| $\Gamma_3$ | $T_{11} = T_3 * T_7$<br>$T_{11} = T_{11} + T_{10}$<br>$T_4 = -T_4$<br>* | $\Gamma_4$ | $T_{12} = T_9 * T_1$<br>*<br>*<br>* |
| $\Gamma_5$ | $T_{10} = T_3 * T_{11}$<br>$T_{10} = T_{10} + T_{12}$<br>$T_9 = -T_9$<br>* | $\Gamma_6$ | $T_{13} = T_1 * T_7$<br>*<br>$T_{13} = -T_{13}$<br>* |
| $\Gamma_7$ | $T_{12} = T_2 * T_2$<br>$T_{14} = T_{12} + T_{13}$<br>$T_{13} = -T_{13}$<br>$T_{14} = T_{14} + T_{14}$ | $\Gamma_8$ | $T_{15} = T_7 * T_7$<br>*<br>*<br>* |
| $\Gamma_9$ | $T_{16} = T_{15} * f3$<br>$T_{16} = T_{16} + T_{12}$<br>*<br>$T_{12} = T_{13} + T_{13}$ | $\Gamma_{10}$ | $T_6 = T_6 * T_{10}$<br>*<br>*<br>* |
| $\Gamma_{11}$ | $T_{15} = T_{15} * T_7$<br>$T_{14} = T_{14} + T_{16}$<br>$T_{16} = -T_{16}$<br>$T_{12} = T_{12} + T_{12}$ | $\Gamma_{12}$ | $T_6 = T_6 * T_7$<br>*<br>*<br>* |
| $\Gamma_{13}$ | $T_{15} = T_{15} * f2$<br>$T_{12} = T_{12} + T_{16}$<br>*<br>* | $\Gamma_{14}$ | $T_{14} = T_8 * T_{14}$<br>*<br>$T_{14} = -T_{14}$<br>* |
| $\Gamma_{15}$ | $T_{12} = T_{12} * T_2$<br>$T_{12} = T_{12} + T_{15}$<br>$T_{15} = -T_6$<br>$T_2 = T_2 + 1$ | $\Gamma_{16}$ | $T_{16} = T_{14} * T_4$<br>*<br>*<br>* |
| $\Gamma_{17}$ | $T_8 = T_8 * T_{12}$<br>$T_8 = T_8 + T_9$<br>$T_6 = -T_{14}$<br>* | $\Gamma_{18}$ | $T_{13} = T_{13} * T_{16}$<br>*<br>*<br>* |
| $\Gamma_{19}$ | $T_9 = T_8 * T_{11}$<br>$T_8 = T_8 + T_6$<br>$T_{14} = -T_{15}$<br>$T_{12} = T_4 + T_{11}$ | $\Gamma_{20}$ | $T_{16} = T_{16} * T_2$<br>*<br>$T_2 = -T_2$<br>$T_2 = T_2 + 1$ |

**Algorithm HCDBL(Contd.)**

| | | | |
|---|---|---|---|
| $\Gamma_{21}$ | $T_{12} = T_{12} * T_8$<br>$T_{12} = T_{12} + T_{16}$<br>$T_4 = -T_4$<br>* | $\Gamma_{22}$ | $T_6 = T_{14} * T_5$<br>$T_{13} = T_{13} + T_9$<br>$T_9 = -T_9$<br>$T_6 = T_6 + T_6$ |
| $\Gamma_{23}$ | $T_{11} = T_{13} * T_{13}$<br>$T_{12} = T_{12} + T_9$<br>*<br>* | $\Gamma_{24}$ | $T_{14} = T_{14} * T_{14}$<br>*<br>$T_2 = -T_2$<br>* |
| $\Gamma_{25}$ | $T_5 = T_7 * T_{12}$<br>*<br>*<br>* | $\Gamma_{26}$ | $T_9 = T_{12} * T_{13}$<br>*<br>*<br>* |
| $\Gamma_{27}$ | $T_7 = T_5 * T_5$<br>*<br>*<br>* | $\Gamma_{28}$ | $T_{14} = T_{14} * T_2$<br>$T_8 = T_1 + T_2$<br>*<br>$T_{14} = T_{14} + T_{14}$ |
| $\Gamma_{29}$ | $T_1 = T_1 * T_9$<br>*<br>$T_1 = -T_1$<br>* | $\Gamma_{30}$ | $T_{12} = T_{12} * T_5$<br>$T_{15} = T_{12} + T_9$<br>*<br>$T_{14} = T_{14} + T_{14}$ |
| $\Gamma_{31}$ | $T_2 = T_2 * T_{12}$<br>*<br>$T_2 = -T_2$<br>* | $\Gamma_{32}$ | $T_{15} = T_8 * T_{15}$<br>$T_{15} = T_1 + T_{15}$<br>$T_1 = -T_1$<br>* |
| $\Gamma_{33}$ | $T_{13} = T_5 * T_{13}$<br>$T_{15} = T_{15} + T_2$<br>$T_{13} = -T_{13}$<br>$T_2 = T_2 + T_{13}$ | $\Gamma_{34}$ | $T_{10} = T_5 * T_{10}$<br>$T_{10} = T_{10} + T_{10}$<br>*<br>* |
| $\Gamma_{35}$ | $T_4 = T_4 * T_{10}$<br>$T_{14} = T_4 + T_{14}$<br>$T_4 = -T_4$<br>$T_{14} = T_{14} + T_{14}$ | $\Gamma_{36}$ | $T_3 = T_3 * T_{10}$<br>$T_{13} = T_{13} + T_{13}$<br>$T_{15} = -T_{15}$<br>$T_3 = T_1 + T_3$ |
| $\Gamma_{37}$ | $T_3 = T_3 * T_7$<br>$T_1 = T_{14} + T_{11}$<br>$T_3 = -T_3$<br>$T_4 = T_1 + T_4$ | $\Gamma_{38}$ | $T_8 = T_6 * T_6$<br>$T_{16} = T_{13} + T_8$<br>$T_2 = -T_2$<br>$T_{12} = T_2 + T_{16}$ |
| $\Gamma_{39}$ | $T_{11} = T_{12} * T_1$<br>$T_3 = T_3 + T_{11}$<br>*<br>$T_4 = T_4 + T_{15}$ | $\Gamma_{40}$ | $T_{13} = T_{12} * T_{16}$<br>*<br>$T_2 = -T_{16}$<br>* |
| $\Gamma_{41}$ | $T_4 = T_4 * T_7$<br>*<br>$T_{13} = -T_{13}$<br>$T_4 = T_4 + T_{13}$ | $(\Gamma_{42})$ | *<br>*<br>*<br>* |
| $U_0' \leftarrow T_1, U_1' \leftarrow T_2, V_0' \leftarrow T_3, V_1' \leftarrow T_4$<br>$Z_1' \leftarrow T_5, Z_2' \leftarrow T_6, z_1' \leftarrow T_7, z_2' \leftarrow T_8.$ | | | |

**Table 3.** HCADD Algorithm in Atomic Blocks, odd char

**Algorithm HCADD**

*Input:* $D_1 = (U_{10}, U_{11}, V_{10}, V_{11}, 1, 1, 1, 1)$
$D_2 = (U_{20}, U_{21}, V_{20}, V_{21}, Z_{21}, Z_{22}, z_{21}, z_{22})$
*Out:* $D_1 + D_2 = (U'_0, U'_1, V'_0, V'_1, Z'_1, Z'_2, z'_1, z'_2)$
*Init:* $T_1 = U_{10}, T_2 = U_{11}, T_3 = V_{10}, T_4 = V_{11},$
$T_5 = U_{20}, T_6 = U_{21}, T_7 = V_{20}, T_8 = V_{21},$
$T_9 = Z_{21}, T_{10} = Z_{22}, T_{11} = z_{21}, T_{12} = z_{22}$

| | | | |
|---|---|---|---|
| $\varGamma_1$ | $T_{10} = T_9 * T_{10}$ | $\varGamma_2$ | $T_{13} = T_2 * T_{11}$ |
| | $*$ | | $*$ |
| | $*$ | | $T_6 = -T_6$ |
| | $*$ | | $T_{13} = T_6 + T_{13}$ |
| $\varGamma_3$ | $T_{12} = T_{10} * T_{11}$ | $\varGamma_4$ | $T_{14} = T_1 * T_{11}$ |
| | $*$ | | $*$ |
| | $T_6 = -T_6$ | | $T_{14} = -T_{14}$ |
| | $*$ | | $T_{14} = T_5 + T_{14}$ |
| $\varGamma_5$ | $T_{15} = T_2 * T_{13}$ | $\varGamma_6$ | $T_{16} = T_{13} * T_{13}$ |
| | $T_{15} = T_{14} + T_{15}$ | | $*$ |
| | $*$ | | $*$ |
| | $*$ | | $*$ |
| $\varGamma_7$ | $T_{14} = T_{14} * T_{15}$ | $\varGamma_8$ | $T_{16} = T_1 * T_{16}$ |
| | $*$ | | $*$ |
| | $*$ | | $*$ |
| | $*$ | | $*$ |
| $\varGamma_9$ | $T_{17} = T_3 * T_{12}$ | $\varGamma_{10}$ | $T_{18} = T_4 * T_{12}$ |
| | $T_{14} = T_{14} + T_{16}$ | | $*$ |
| | $T_7 = -T_7$ | | $T_8 = -T_8$ |
| | $T_{17} = T_7 + T_{17}$ | | $T_{18} = T_8 + T_{18}$ |
| $\varGamma_{11}$ | $T_{12} = T_{15} * T_{17}$ | $\varGamma_{12}$ | $T_{16} = T_{13} * T_{18}$ |
| | $T_{15} = T_{13} + T_{15}$ | | $T_{18} = T_{17} + T_{18}$ |
| | $T_{12} = -T_{12}$ | | $T_{16} = -T_{16}$ |
| | $*$ | | $*$ |
| $\varGamma_{13}$ | $T_{15} = T_{15} * T_{18}$ | $\varGamma_{14}$ | $T_{17} = T_{10} * T_{14}$ |
| | $T_{15} = T_{12} + T_{15}$ | | $*$ |
| | $T_{12} = -T_{12}$ | | $T_7 = -T_7$ |
| | $T_{18} = 1 + T_2$ | | $*$ |
| $\varGamma_{15}$ | $T_{18} = T_{16} * T_{18}$ | $\varGamma_{16}$ | $T_{10} = T_9 * T_{17}$ |
| | $T_{15} = T_{15} + T_{18}$ | | $*$ |
| | $*$ | | $T_8 = -T_8$ |
| | $*$ | | $*$ |
| $\varGamma_{17}$ | $T_{18} = T_1 * T_{16}$ | $\varGamma_{18}$ | $T_{17} = T_{17} * T_{17}$ |
| | $T_{18} = T_{12} + T_{18}$ | | $*$ |
| | $*$ | | $*$ |
| | $*$ | | $*$ |
| $\varGamma_{19}$ | $T_{12} = T_{13} * T_{15}$ | $\varGamma_{20}$ | $T_9 = T_9 * T_{15}$ |
| | $*$ | | $*$ |
| | $T_{12} = -T_{12}$ | | $*$ |
| | $*$ | | $*$ |

**Algorithm HCADD(Contd.)**

| | | | |
|---|---|---|---|
| $\varGamma_{21}$ | $T_{16} = T_{11} * T_{18}$ | $\varGamma_{22}$ | $T_{14} = T_{14} * T_{15}$ |
| | $T_{16} = T_{12} + T_{16}$ | | $*$ |
| | $*$ | | $*$ |
| | $*$ | | $*$ |
| $\varGamma_{23}$ | $T_{12} = T_2 * T_{15}$ | $\varGamma_{24}$ | $T_8 = T_8 * T_{14}$ |
| | $*$ | | $*$ |
| | $T_{12} = -T_{12}$ | | $*$ |
| | $T_{12} = T_{12} + T_{18}$ | | $*$ |
| $\varGamma_{25}$ | $T_{16} = T_{12} * T_{16}$ | $\varGamma_{26}$ | $T_{18} = T_{15} * T_{18}$ |
| | $*$ | | $*$ |
| | $*$ | | $*$ |
| | $*$ | | $*$ |
| $\varGamma_{27}$ | $T_{12} = T_{11} * T_{18}$ | $\varGamma_{28}$ | $T_{15} = T_{15} * T_{15}$ |
| | $T_{11} = T_6 + T_{13}$ | | $*$ |
| | $T_{12} = -T_{12}$ | | $*$ |
| | $T_{11} = T_6 + T_{11}$ | | $*$ |
| $\varGamma_{29}$ | $T_{17} = T_{11} * T_{17}$ | $\varGamma_{30}$ | $T_{13} = T_{13} * T_{15}$ |
| | $T_{16} = T_{16} + T_{17}$ | | $T_{13} = T_{12} + T_{13}$ |
| | $*$ | | $*$ |
| | $T_{16} = T_8 + T_{16}$ | | $T_{13} = T_{12} + T_{13}$ |
| $\varGamma_{31}$ | $T_{11} = T_6 * T_{15}$ | $\varGamma_{32}$ | $T_{17} = T_5 * T_{18}$ |
| | $T_6 = T_5 + T_6$ | | $T_{18} = T_{15} + T_{18}$ |
| | $T_{11} = -T_{11}$ | | $T_{12} = -T_{12}$ |
| | $T_{16} = T_8 + T_{16}$ | | $*$ |
| $\varGamma_{33}$ | $T_{18} = T_6 * T_{18}$ | $\varGamma_{34}$ | $T_7 = T_7 * T_{14}$ |
| | $T_{18} = T_{11} + T_{18}$ | | $T_7 = T_7 + T_{17}$ |
| | $T_{11} = -T_{11}$ | | $T_{17} = -T_{17}$ |
| | $T_{11} = T_{11} + T_{12}$ | | $*$ |
| $\varGamma_{35}$ | $T_6 = T_9 * T_9$ | $\varGamma_{36}$ | $T_{12} = T_{10} * T_{10}$ |
| | $T_{18} = T_{17} + T_{18}$ | | $T_{13} = T_{12} + T_{13}$ |
| | $T_{17} = -T_6$ | | $*$ |
| | $*$ | | $T_{11} = T_{11} + T_{13}$ |
| $\varGamma_{37}$ | $T_5 = T_1 * T_{17}$ | $\varGamma_{38}$ | $T_7 = T_7 * T_{17}$ |
| | $T_5 = T_5 + T_{16}$ | | $*$ |
| | $T_5 = -T_5$ | | $T_6 = -T_{13}$ |
| | $T_8 = T_5 + T_8$ | | $*$ |
| $\varGamma_{39}$ | $T_8 = T_8 * T_{17}$ | $\varGamma_{40}$ | $T_{14} = T_{11} * T_{13}$ |
| | $*$ | | $*$ |
| | $T_5 = -T_5$ | | $T_{14} = -T_{14}$ |
| | $T_5 = T_5 + T_{18}$ | | $*$ |
| $\varGamma_{41}$ | $T_{15} = T_5 * T_{11}$ | $(\varGamma_{42})$ | $*$ |
| | $T_7 = T_7 + T_{15}$ | | $*$ |
| | $T_{11} = -T_{17}$ | | $*$ |
| | $T_8 = T_8 + T_{14}$ | | $*$ |

$U'_0 \leftarrow T_5, U'_1 \leftarrow T_6, V'_0 \leftarrow T_7, V'_1 \leftarrow T_8,$
$Z'_1 \leftarrow T_9, Z'_2 \leftarrow T_{10}, z'_1 \leftarrow T_{11}, z'_2 \leftarrow T_{12}.$

following we assume that multiplications and squarings cannot be distinguished from the side-channel information. This is a common assumption for *odd characteristic* also made in [7], but it needs to be verified for each environment. We comment on this after the main explanation in Remark 1.

With these conventions both divisor addition and doubling require $41[m]$ each. From a detailed investigation of the formulas in [17] and the papers referred therein, we observe that there are at most 2 additions and 1 negation needed between 2 multiplications. Accordingly we design our atomic blocks to contain one multiplication, two additions and a negation. We remark that the same solution has been chosen in [7]. We checked that this leads to the optimal design as on the one hand more additions and negations are not useful. On the other hand, even though more dummy operations (additions and negations) are needed compared to ECC, smaller blocks lead to many dummy multiplications which turn out to be worse than the extra additions. Furthermore, these idle operations will help in case $[s] \neq [m]$.

Now that we fixed the content of the blocks our aim is to split the explicit formulae into atomic blocks which will easily conform to a parallel implementation as well. In [20] the authors have proposed a general methodology to parallelize any explicit formula. The proposed methodology requires to create a precedence digraph of the multiplication operations. The vertices represent the multiplication operations. There is one arc from a vertex $v_1$ to $v_2$ if the later can be computed after the first one without any intermediate multiplication operation.

In the situation we are in now, one feels tempted to split the explicit formulae into atomic blocks first and then go for the said methodology treating one atomic block as one operation in order to create a precedence graph of atomic blocks. One important observation of our study is that such a strategy may not bear fruit. That is simply because one can split the HCADD and HCDBL into atomic operations in several ways, hence the precedence graph is not unique. Therefore we use the other strategy, i. e. we start by using the methodology of [20] to find out the multiplication operations which can be implemented in parallel. Put them in different atomic blocks and then distribute the addition and subtraction operation suitably among them. We take care that while two atomic blocks are being processed in parallel no conflict occurs.

In Table 2 we present the HCDBL algorithm for odd characteristic split into atomic blocks and in Table 3 the HCADD (mixed coordinates) is stated. Note, that the dummy step $\Gamma_{42}$ is not needed in a sequential implementation. In a parallel implementation one needs to introduce this additional atomic block as otherwise the second processor would remain idle which would be visible from the side-channel.

*Remark 1.* If squarings and multiplications can be distinguished from their side-channel information one can work around by replacing the squaring $T_j = T_i * T_i$ by the sequence of assignments $T_j = T_i + (-1), T_j = T_j * T_i$ and $T_j = T_j + T_i$. The value $-1$ is stored in some extra variable. As there are many idle addition states one would expect that this countermeasure does not require any overhead. However, the first block in doubling is a squaring such that the inputs cannot be adjusted beforehand. Our suggestion for this case is to use a different atomic block which starts with an addition of the constant $-1$ which can obviously be replaced by a dummy operation if not needed. The resulting formulas follow directly from ours. At first sight it might seem desirable to use the cheaper step "addition of the constant 1" instead. But this requires a subtraction later on and in the later steps of HCADD and HCDBL there are not enough idle negation and addition steps to take care of this.

### 5.1 Parallel Implementation

One can observe that the explicit formula for odd characteristic can be processed in parallel. One set of multiplier and adder can be engaged in computing the atomic blocks with odd subscript (e. g. $\Gamma_1, \Gamma_3, \ldots$ in HCADD), while the other set of multiplier and adder can be assigned the atomic blocks with even indices (e. g. $\Gamma_2, \Gamma_4, \ldots$ in HCADD). The tables are designed in order to avoid conflicts in writing and reading registers.

## 6 Memory Requirements

Memory requirement is an important consideration for an algorithm which is likely to be used in resource constrained devices. In even characteristic, as can be seen in the tables, HCADD uses 14 registers including 4 registers for the base divisor and at most 3 registers to store non-trivial curve parameters $h_1, h_0$ and $f_4$. HCDBL uses 12 and at most 5 registers to store non-trivial curve parameters $h_0, h_1, f_2, f_3, f_4$. Table 1 also shows how to schedule the dummy operations in order to avoid using a further register. Thus for even characteristic 17 registers are required to compute the scalar multiplication. To avoid expensive reads from the long term memory one could add 4 more registers to keep the base divisor and the curve parameters. From these 21 registers only 12 are active, the others are read only.

In odd characteristic, HCADD uses 18 registers including 4 read-only registers ($T_1 - T_4$) for the base divisor in affine coordinates, the curve parameters are not used. HCDBL uses 16 active registers and only two curve coefficients need to be stored, namely $f_2, f_3$. We need 1 register for dummy operations, thus 19 registers are enough to perform a scalar multiplication. To reduce the number of memory accesses, the base divisor and the curve coefficients should be kept in registers throughout the scalar multiplication. Then in total, 23 registers are required out of which 6 are nonactive. Note that we have used more registers to enable the algorithm to be implemented in parallel. To avoid conflicts, we do not use a register for writing if it read earlier (e. g. in the additions after the multiplications) in the same block by the parallel processor, even though it might be free at that time. We do not see a problem with both processors reading at the same time. HCADD and HCDBL designed for sequential implementation would require slightly smaller number of registers.

It is worthwhile to note that the registers are around 80 bits here, capable of holding one field element. In ECC, the registers have double size, showing that in relation we do not need much more space while achieving not only SPA resistance but also parallel executable formulae.

## 7 Conclusion

In this paper we have developed the arithmetic on genus 2 curves in odd and even characteristic such that the side-channel information is uniform. This was done in odd characteristic by introducing atomic blocks from which the whole group operation can be built. The additional dummy operations are only additions and negations.

In even characteristic each group operation constitutes a separate atomic block which results in one additional dummy multiplication and one squaring. This approach is natural as inversions are not too costly in characteristic two and, hence, inversion-free formulae are slower for most devices. If inversions are more expensive than we assume, one can use Lange's new coordinates [17] and build the group operations from small atomic blocks as in the case of odd characteristic. Since in even characteristic squarings are no longer comparable to multiplications, we suggest to build an atomic block consisting of 1 multiplication and $n+2$ additions, where $n$ is the number of additions needed to perform a squaring in the given field representation. As this depends highly on the representation – trinomial vs. pentanomial and polynomial bases vs. normal basis we have not included this study but for each fixed field implementation this is possible to build – but too lengthy for publication. For special choices of the curve, especially for $\deg h = 1$, this approach is more efficient as the doublings need fewer operations [18] and a cheap doubling is useful in windowing methods.

On the following pages we state the tables for odd characteristic.

## References

1. R. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *The Handbook of Elliptic and Hyperelliptic Curve Cryptography.* CRC Press, 2005.
2. R. M. Avanzi. Countermeasures Against Differential Power Analysis for Hyperelliptic Curve Cryptosystems. In *Cryptographic Hardware and Embedded Systems CHES 2003*, volume 2779 of *LNCS*, pages 366–381, 2004.
3. G. Bertoni, L. Breveglieri, T. Wollinger, and C. Paar. *Embedded Cryptographic Hardware: Design and Security*, chapter Hyperelliptic Curve Cryptosystem: What is the Best Parallel Hardware Architecture. Nova Science Publishers, 2004.
4. I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic curves in cryptography*, volume 265 of *London Mathematical Society Lecture Note Series.* Cambridge University Press, Cambridge, 1999.
5. I. F. Blake, G. Seroussi, and N. P. Smart. *Advances in Elliptic Curve Cryptography.* London Mathematical Society Lecture Note Series. Cambridge University Press, Cambridge, 2005.
6. D. G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Math. Comp.*, 48:95–101, 1987.
7. B. Chevallier-Mames, M. Ciet, and M. J oye. Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity. *IEEE Trans. on Computers*, 53:760–768, 2004. Cryptology ePrint Archive, Report 2003/237.
8. J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *Cryptographic Hardware and Embedded Systems CHES 1999*, volume 1717 of *Lect. Notes Comput. Sci.*, pages 392–302. Springer, 1999.

9. S. Duquesne. Montgomery scalar multiplication for genus 2 curves. In *Algorithmic Number Theory Seminar ANTS-VI*, volume 3076 of *Lect. Notes Comput. Sci.*, pages 153–168, 2004.
10. P. Gaudry. Fast genus 2 arithmetic based on Theta functions. Cryptology ePrint Archive, Report 2005/314.
11. P. Gaudry and R. Harley. Counting points on hyperelliptic curves over finite fields. In *Advances in cryptology – Eurocrypt'2000*, volume 1838 of *Lect. Notes Comput. Sci.*, pages 313–332. Springer, 2000.
12. R. Harley. Fast arithmetic on genus 2 curves. available at `http://cristal.inria.fr/~harley/hyper`, 2000.
13. T. Izu, B. Möller, and T. Takagi. Improved Elliptic Curve Multiplication Methods Resistant Against Side Channel Attacks. In *Progress in Cryptology - Indocrypt 2002*, volume 2551 of *Lect. Notes Comput. Sci.*, pages 296–313. Springer, 2002.
14. M. Jacobson, A. Menezes, and A. Stein. Hyperelliptic Curves in Cryptography. In *High Primes and Misdemeanours: lectures in honour of the 60th birthday of Hugh Cowie Williams*, volume 41 of *Fields Institute Communications*, pages 255–282. AMS, 2004.
15. N. Koblitz. Hyperelliptic cryptosystems. *J. Cryptology*, 1:139–150, 1989.
16. T. Lange. Montgomery Addition for Genus Two Curves. In *Algorithmic Number Theory Seminar ANTS-VI*, volume 3076 of *Lect. Notes Comput. Sci.*, pages 309–317, 2004.
17. T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *J. AAECC*, 15:295 – 328, 2005.
18. T. Lange and M. Stevens. Efficient doubling for genus two curves over binary fields. Preproceedings of SAC 2004, p. 189–202, 2004.
19. A. J. Menezes, Y.-H. Wu, and R. Zuccherato. An Elementary Introduction to Hyperelliptic Curves. In N. Koblitz, editor, *Algebraic Aspects of Cryptography*, pages 155–178. Springer, 1998.
20. P. K. Mishra and P. Sakar. Parallelizing Explicit Formula in the Jacobian of Hyperelliptic Curves. In *Proceedings of Asiacrypt 2003*, volume 2894 of *LNCS*, pages 93–110, 2003. full version at Cryptology ePrint Archive, Report 2003/180.
21. B. Möller. Securing elliptic curve point multiplication against side-channel attacks. In *Proc. of ISC 2001*, pages 324–334, 2001.
22. D. Naccache, J. Stern, and N. P. Smart. Projective Coordinates Leak. In *Advances in cryptology – Eurocrypt'2004*, volume 3027 of *Lect. Notes Comput. Sci.*, pages 257–267. Springer, 2004.
23. A. M. Spallek. *Kurven vom Geschlecht 2 und ihre Anwendung in Public-Key-Kryptosystemen.* PhD thesis, University Essen, 1994.

# Author Index