Daniel J. Bernstein
Tanja Lange (Eds.)

LNCS 6055

# Progress in Cryptology – AFRICACRYPT 2010

Third International Conference on Cryptology in Africa
Stellenbosch, South Africa, May 2010
Proceedings

Springer

Lecture Notes in Computer Science 6055

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

Daniel J. Bernstein  Tanja Lange (Eds.)

# Progress in Cryptology – AFRICACRYPT 2010

Third International Conference on Cryptology in Africa
Stellenbosch, South Africa, May 3-6, 2010
Proceedings

Springer

Volume Editors

Daniel J. Bernstein
University of Illinois at Chicago
Department of Computer Science
Chicago, IL 60607–7045, USA
E-mail: djb@cr.yp.to

Tanja Lange
Eindhoven University of Technology
Department of Mathematics and Computer Science
Den Dolech 2, 5612 AZ Eindhoven, The Netherlands
E-mail: tanja@hyperelliptic.org

Library of Congress Control Number: 2010924884

CR Subject Classification (1998): E.3, C.2, K.6.5, D.4.6, J.1, H.4

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-642-12677-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-12677-2 Springer Berlin Heidelberg New York



Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

# Preface

Africacrypt 2010, the Third International Conference on Cryptology in Africa, took place May 3–6, 2010 in Stellenbosch, South Africa. The General Chairs, Riaal Domingues from the South African Communications and Security Agency and Christine Swart from the University of Cape Town, were always a pleasure to work with and did an outstanding job with the local arrangements. We are deeply thankful that they agreed to host Africacrypt 2010 with only four months notice after unanticipated events forced a change of location.

The Africacrypt 2010 submission deadline was split into two. Authors submitting papers were required to register titles and abstracts by the first deadline, January 5. A total of 121 submissions had been received by this deadline, although some were withdrawn before review. Authors were allowed to continue working on their papers until the second deadline, January 10.

Submissions were evaluated in three phases over a period of nearly two months. The selection phase started on January 5: Program Committee members began evaluating abstracts and volunteering to handle various papers. We assigned a team of people to each paper. The review phase started on January 11: Program Committee members were given access to the full papers and began in-depth reviews of 82 submissions. Most of the reviews were completed by February 7, the beginning of the discussion phase. Program Committee members were given access to other reviews and built consensus in their evaluations of the submissions. In the end the discussions included 285 full reports and 203 additional comments. The submissions, reviews, and subsequent discussions were handled smoothly by iChair.

On February 21 we sent out 2 notifications of conditional acceptance and 24 notifications of unconditional acceptance. The next day we sent out comments from the reviewers. One paper eventually met its acceptance conditions; the final program contained 25 contributed papers and 3 invited talks. The authors prepared final versions of the 25 contributed papers by February 28.

It is our pleasure to thank the other 53 Program Committee members for lending their expertise to Africacrypt 2010 and for putting tremendous effort into detailed reviews and discussions. We would also like to thank Thomas Baignères and Matthieu Finiasz for writing the iChair software; Springer for agreeing to an accelerated schedule for printing the proceedings; 70 external referees who reviewed individual papers upon request from the Program Committee; and, most importantly, all authors for submitting interesting new research papers to Africacrypt 2010.

May 2010

Daniel J. Bernstein and Tanja Lange
Program Chairs, Africacrypt 2010

# Organization

## General Chairs

| | |
|---|---|
| Riaal Domingues | South African Communications and Security Agency |
| Christine Swart | University of Cape Town, South Africa |

## Program Chairs

| | |
|---|---|
| Daniel J. Bernstein | University of Illinois at Chicago, USA |
| Tanja Lange | Technische Universiteit Eindhoven, The Netherlands |

## Program Committee

| | |
|---|---|
| Michel Abdalla | Ecole Normale Superieure, France |
| Roberto Avanzi | Ruhr-University Bochum, Germany |
| Hatem M. Bahig | Ain Shams University, Egypt |
| Paulo S.L.M. Barreto | University of São Paulo, Brazil |
| Lejla Batina | Radboud University Nijmegen, The Netherlands and Katholieke Universiteit Leuven, Belgium |
| Daniel J. Bernstein | University of Illinois at Chicago, USA |
| Ashraf M. Bhery | Ain Shams University, Egypt |
| Peter Birkner | Université de Versailles Saint-Quentin-en-Yvelines, France |
| Colin Boyd | Queensland University of Technology, Australia |
| Xavier Boyen | University of Liege, Belgium |
| Johannes Buchmann | TU Darmstadt, Germany |
| Christophe De Cannière | Katholieke Universiteit Leuven, Belgium |
| Chen-Mou Cheng | National Taiwan University, Taiwan |
| Carlos Cid | Royal Holloway, University of London, UK |
| Alexander W. Dent | Royal Holloway, University of London, UK |
| Yvo Desmedt | University College London, UK and RCIS, AIST, Japan |
| Christophe Doche | Macquarie University, Australia |
| Orr Dunkelman | Weizmann Institute, Israel |
| Matthieu Finiasz | ENSTA, France |
| Shay Gueron | University of Haifa and Intel Corporation, Israel |
| Tim Güneysu | Ruhr-University Bochum, Germany |
| Helena Handschuh | Katholieke Universiteit Leuven, Belgium and Intrinsic-ID, USA |

| | |
|---|---|
| Antoine Joux | DGA and University of Versailles Saint-Quentin-en-Yvelines, France |
| Marc Joye | Technicolor, France |
| Tanja Lange | Technische Universiteit Eindhoven, The Netherlands |
| Keith Martin | Royal Holloway, University of London, UK |
| Mitsuru Matsui | Mitsubishi Electric, Japan |
| David McGrew | Cisco, USA |
| Alfred Menezes | University of Waterloo, Canada |
| Michele Mosca | University of Waterloo, Canada |
| Michael Naehrig | Microsoft Research, USA |
| Abderrahmane Nitaj | Université de Caen, France |
| Elisabeth Oswald | University of Bristol, UK |
| Christof Paar | Ruhr-University Bochum, Germany |
| Daniel Page | University of Bristol, UK |
| Josef Pieprzyk | Macquarie University, Australia |
| Bart Preneel | Katholieke Universiteit Leuven, Belgium |
| Christian Rechberger | Katholieke Universiteit Leuven, Belgium |
| Magdy Saeb | Arab Academy for Science, Technology & Maritime Transport, Egypt |
| Palash Sarkar | Indian Statistical Institute, India |
| Berry Schoenmakers | Technische Universiteit Eindhoven, The Netherlands |
| Michael Scott | Dublin City University, Ireland |
| Nicolas Sendrier | INRIA, France |
| Francesco Sica | University of Calgary, Canada |
| Martijn Stam | EPFL, Switzerland |
| François-Xavier Standaert | Universite catholique de Louvain, Belgium |
| Damien Stehlé | CNRS, France; University of Sydney and Macquarie University, Australia |
| Christine Swart | University of Cape Town, South Africa |
| Mike Szydlo | Akamai, USA |
| Brent Waters | University of Texas at Austin, USA |
| Michael J. Wiener | Cryptographic Clarity, Canada |
| Bo-Yin Yang | Academia Sinica, Taiwan |
| Amr M. Youssef | Concordia University, Canada |
| Paul Zimmermann | INRIA Nancy - Grand Est, France |

## Referees

| | | |
|---|---|---|
| Martin Albrecht | Andre Bogdanov | Rafik Chaabouni |
| Daniel Augot | Ignacio Cascudo | Pascale Charpin |
| Timo Bartkewitz | Julien Cathalo | Melissa Chase |
| Gaetan Bisson | Emanuele Cesena | Sanjit Chatterjee |
| Carlo Blundo | Kishan C. Gupta | Sherman Chow |

Iwen Coisel
Baudoin Collard
Ronald Cramer
Özgür Dagdelen
Erik Dahmen
Benne de Weger
Sebastian Faust
Georg Fuchsbauer
David Galindo
Juan A. Garay
Flavio Garcia
Matthew Green
Jens Groth
Darrel Hankerson
Mathias Herrmann
Dennis Hofheinz
Vangelis Karatsiolis
Eike Kiltz
Alptekin Küpçü
Yann Laigle-Chapuy
Lucie Langer
Gregor Leander
Kerstin Lemke-Rust
Benoît Libert
Mark Manulis
Gregory Neven
Claudio Orlandi
David Oswald
Pascal Paillier
Christiane Peters
Christophe Petit
Francesco Regazzoni
Alfredo Rial
Maike Ritzenhofen
Bill Rosgen
Markus Rückert
Ulrich Ruhrmair
Rei Safavi-Naini
Dominique Schröder
Peter Schwabe
Jun Shao
Marcos A. Simplício Jr
Dave Singelee
Douglas Stebila
Douglas Stinson
Enrico Thomae
Michael Tunstall
Pim Tuyls
Berkant Ustaoglu
Marion Videau
Christian Wachsmann
Bogdan Warinschi
Christopher Wolf
Qiushi Yang
Hong-Sheng Zhou

# Table of Contents

## Elliptic Curves



## Side-Channel Attacks and Fault Attacks



## Public-Key Encryption

## Keys and PUFs



## Ciphers and Hash Functions

# A New RSA-Based Signature Scheme

Sven Schäge and Jörg Schwenk

Horst Görtz Institute for IT-Security, University of Bochum, Germany
{sven.schaege,joerg.schwenk}@rub.de

**Abstract.** In this work we present a new and efficient hash-and-sign signature scheme in the standard model that is based on the RSA assumption. Technically it adapts the new proof techniques that are used to prove the recent RSA scheme by Hohenberger and Waters. In contrast to the Hohenberger-Waters scheme our scheme allows to sign blocks of messages and to issue signatures on committed values, two key properties required for building privacy preserving systems.

**Keywords:** digital signature schemes, standard model, RSA, message blocks.

## 1 Introduction

It is well known that the basic RSA signature scheme [23] is not secure with respect to the most stringent security definition for digital signature schemes – existential unforgeability against adaptive chosen message attacks. To overcome this deficiency the basic RSA scheme has been extended in various ways. Several signature schemes have been proven secure in the random oracle model [1] like full domain hash RSA [1] or probabilistic signature scheme (PSS) [2]. It was a long open-standing problem to build compact signature schemes which are solely secure under the RSA assumption and until 2009, the only efficient RSA-based signature schemes that were secure in the standard model were based on tree structures [10,13]. In 2009, Hohenberger and Waters step-wisely solved this problem by first presenting a new stateful RSA signature scheme (at Eurocrypt '09) and then describing the first stateless RSA-based hash-and-sign signature scheme in the standard model (Crypto '09).

Signature Schemes for Signing Message Blocks. In this paper we investigate whether the recent results by Hohenberger and Waters can be used to construct a signature scheme that supports signing message blocks. A signature scheme for message blocks, produces signatures over several independent messages. The user may then prove (with additional NIZK protocols) that it knows the input messages and the corresponding signature without revealing these values. The main advantage is that the user may also disclose a relevant subset of the messages to the verifier without compromising the secrecy of the remaining attributes. Typically these schemes are used to generate certificates on private user attributes like name, age, address, or marital status of the user. For example imagine a user whose personal data has been certified by a global authority.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 1–15, 2010.

If the user wants to order a new book in an online book store he is surely not supposed to provide information on his marital status. On the other hand, to ensure a successful delivery of the book his name and address are of course required information. When using signature schemes for message blocks the user can so protect his private information by only disclosing absolutely neccessary information to his communication partner. In 2002, Camenisch and Lysyanskaya presented a signature scheme that is secure under the Strong RSA assumption (SRSA) [8] (Appendix A) . The scheme has three key properties: 1) it can easily be extended to sign several message blocks, 2) there exists efficient protocols for issuing signatures on committed values and 3) there exist efficient protocols for proving knowledge of a signature without actually revealing it. These properties have made the Camenisch-Lysyanskaya (CL) scheme a useful building block in several privacy preserving systems [4,5,6,7]

Contribution. In this work we present a new signature scheme based on the RSA assumption that fulfills 1) and 2): it supports the signing of several (independent) message blocks and allows to issue signatures on committed values. This makes our signature scheme useful in applications where privacy is critical. Designing additional protocols that accomplish 3) may be the issue of future work.

Technical Contribution. We start with revisiting the Camenisch-Lysyanskaya scheme. In the security proof of the CL scheme the simulator at first guesses one of three types of forgery. According to its guess it prepares the input values that are given to the attacker against the CL scheme. If the simulator's guess is correct it can break the Strong RSA assumption. Our key observation is the following: two of these types of forgery can actually reduce security to the mere RSA assumption. For the design of our new RSA based signature scheme we take this as a starting point. To deal with the third case we essentially use the new proof techniques that were presented by Hohenberger and Waters to prove their RSA based scheme secure. In particular, we integrate that for a string $X$, all prefixes of $X$ are processed as well.

Related Work. Besides being closely related to the recent Hohenberger-Waters signature scheme our work adds to the line of existing signature schemes proven secure under RSA-like assumptions. The first provably secure but impractical signature scheme was published by Goldwasser, Micali and Rivest in 1988 [18]. In 1995 and 1996, Dwork and Naor [13] and Cramer and Damgård [10] presented RSA based tree signature schemes. Naturally for tree-based signatures, the signature size is logarithmic in the maximal number of signatures that can securely be issued. In 1999, Gennaro, Halevi, and Rabin [17] and independently Cramer and Shoup [11] presented the first hash-and-sign signature schemes that are secure in the standard model under a (stronger) variant of the RSA assumption – the Strong RSA assumption. Based on this work, several SRSA-based signature schemes followed in the next years [8, 14, 19, 22, 25, 26]. In 2009, Hohenberger and Waters presented a new RSA signature scheme that is provably secure under the RSA assumption if the signer maintains and regularly updates a counter-value [20]. Later

in this year, Hohenberger and Waters then presented the first stateless RSA-based hash-and-sign signature scheme in the standard model [21].

## 2 Preliminaries

### 2.1 Notation

For $a, b \in \mathbb{Z},\ a \leq b$ we write $[a;b]$ to denote the set $\{a, a+1, \ldots, b-1, b\}$. For a string $x$, we write $|x|_2$ to denote its bit length. For a set $S$, we use $|S|$ to denote its size. By $s \in_R S$ we indicate that $s$ is drawn from $S$ uniformly at random. For $X \in \{0,1\}^k$, let $X = x_1 \ldots x_k$ be the binary presentation of $X$ with $x_i \in \{0,1\}$ for $i \in [1;k]$. We denote by $X^{(i)} = 0^{k-i}x_1 \ldots x_i \in \{0,1\}^k$ the (zero-padded) prefix of $X$ that consists of $k-i$ zeroes and the first $i$ bits of $X$. We use $\mathcal{QR}_n$ to denote the set of quadratic residues modulo $n \in \mathbb{N}$, i.e. $\mathcal{QR}_n = \{x | \exists y \in \mathbb{Z}_n^* : y^2 = x \bmod n\}$. For an algorithm $\mathcal{A}$ we write $\mathcal{A}(x_1, x_2, \ldots)$ to denote that $\mathcal{A}$ has input parameters $x_1, x_2, \ldots$ while $y \leftarrow \mathcal{A}(x_1, x_2, \ldots)$ means that $\mathcal{A}$ outputs $y$ when given the input values $x_1, x_2, \ldots$ . We use $\kappa \in \mathbb{N}$ to indicate the security parameter while $1^\kappa$ describes the string that consist of $\kappa$ ones. Let $l = l(\kappa)$ and $l_X = l_X(\kappa)$ be polynomials with $l_X \leq l$. We will also make use of a prime mapping function $\texttt{nextprime} : \{0,1\}^{l_X} \rightarrow \mathcal{P}_l$ that maps $l_X$-bit strings to the set of primes $\mathcal{P}_l \subseteq [1; 2^l]$.[1] The function `nextprime` maps to the smallest prime that is equal or greater than the input value (when interpreted as a natural number).

### 2.2 Signature Scheme

A digital signature scheme $\mathcal{S} = (\boldsymbol{Gen}, \boldsymbol{Sign}, \boldsymbol{Verify})$ consists of three polynomial-time algorithms.

- $\boldsymbol{Gen}(1^\kappa)$ generates a public key pair $(SK, PK)$.
- $\boldsymbol{Sign}(SK, m)$ outputs the signature $\sigma$.
- $\boldsymbol{Verify}(PK, m, \sigma)$ verifies if $\sigma$ is a signature on $m$ signed by the holder of the secret key corresponding to $PK$; on success it outputs 1 otherwise 0.

### 2.3 Strong Existential Unforgeability

According to the standard definition of security by Goldwasser, Micali and Rivest [18] a signature scheme $\mathcal{S} = (\boldsymbol{Gen}, \boldsymbol{Sign}, \boldsymbol{Verify})$ is existentially unforgeable under an adaptive chosen message attack if no forger that has $q$-time access to a signing oracle $\mathcal{O}(SK, \cdot)$ can produce a new valid signature on a new message. In this work we consider an even stronger version of this security definition called strong existential unforgeability. Now, the message output by the adversary may not necessarily differ from the previous oracle queries.

[1] If we choose $l_X$ such that $2^{l_X} \approx 2^l/l$ we can guarantee by the prime number theorem that $|\mathcal{P}_l| \approx 2^{l_X}$.

**Definition 1.** *We say that the signature scheme $\mathcal{S}$ is $(q,t,\epsilon)$-secure, if for all $t$-time adversaries $\mathcal{A}$ that send at most $q$ queries $m_1,\ldots,m_q$ to the signing oracle $\mathcal{O}(SK,\cdot)$ and receive $q$ answers $\sigma_1,\ldots,\sigma_q$ it holds that*

$$Pr\left[\begin{array}{c}(SK,PK)\leftarrow \boldsymbol{Gen}(1^\kappa),\ (m^*,\sigma^*)\leftarrow \mathcal{A}^{\mathcal{O}(SK,\cdot)}(PK),\\ \boldsymbol{Verify}(PK,m^*,\sigma^*)=1,\ (m^*,\sigma^*)\notin\{(m_1,\sigma_1),\ldots,(m_q,\sigma_q)\}\end{array}\right]\leq\epsilon,$$

*where the probability is taken over the random coins of **Gen** and $\mathcal{A}$.*

Our new signature scheme solely relies on the original RSA assumption [23].

**Definition 2 (RSA assumption (RSA) ).** *Given an RSA modulus $n=pq$, where $p,q$ are sufficiently large primes, a prime $\alpha<\phi(n)$ with $\gcd(\alpha,\phi(n))=1$, and an element $u\in\mathbb{Z}_n^*$, we say that the $(t_{RSA},\epsilon_{RSA})$-RSA assumption holds if for all $t_{RSA}$-time adversaries $\mathcal{A}$*

$$Pr\left[(x)\leftarrow\mathcal{A}(n,u,\alpha),\ x\in\mathbb{Z}_n^*,\ x^\alpha=u \bmod n\right]\leq\epsilon_{RSA},$$

*where the probability is over the random choices of $u,n,\alpha$ and the random coins of $\mathcal{A}$.*

### 2.4 Prime Mapping Function

Let $n=pq$ be an RSA modulus and $l=l(\kappa)$ be such that $2^l<\phi(n)$. We will now construct a function $t:\{0,1\}^{l_X}\rightarrow\mathcal{P}_l$ for mapping bit strings to $\mathcal{P}_l$ that is very similar to the prime mapping function used by Hohenberger and Waters [21] (Appendix B). An essential feature of this function is that the probability of the first polynomial (in the security parameter) inputs to produce collisions is negligible. Basically the construction consists of a keyed pseudo-random permutation $f_k:\{0,1\}^{l_X}\rightarrow\{0,1\}^{l_X}$ with key space $\mathcal{K}$ and key $k\in_R\mathcal{K}$ such that $\log(|\mathcal{K}|)$ is a polynomial in $\kappa$. We also need another random value $s\in_R\{0,1\}^{l_X}$ that is XORed with the output of $f_k$ prior to applying `nextprime`. The final definition of $t$ is $t(X)=\texttt{nextprime}(s\oplus f_k(X))$. We note that `nextprime` is invertible for any prime $y\in\mathcal{P}_l$ although there are possibly several input values that map to $y$. For convenience we additionally define a new function $T:\{0,1\}^{l_X}\rightarrow\mathbb{N}$ as $T(X)=\prod_{i=1}^{l_X}t(X^{(i)})$. We surely have that $T(X_i)\neq T(X_j)\Rightarrow X_i\neq X_j$. Observe that if $t$ is collision-free for the first $q$ inputs, $T$ is also injective such that $T(X_i)\neq T(X_j)\Leftrightarrow X_i\neq X_j$.

*Remark 1.* As in the Hohenberger-Waters proof, the value $s$ is important for embedding the RSA challenge in the security reduction, as it allows the simulator to program $t$ for a single input/output relation. Suppose the simulator wishes to have input $X$ be mapped to the output prime $y$. It can accomplish this by simply setting $s$ such that $\texttt{nextprime}(s\oplus f_k(X))=y$.

## 3 A New RSA-Based Signature Scheme

We now define our new RSA signature scheme $\mathcal{S}$. Let $l=l(\kappa)$, $l_o=l_o(\kappa)$, $l_m=l_m(\kappa)$, and $l_p=l_p(\kappa)$ be polynomials and $l_r=l_m+l_o$, $l_n=2l_p$. For

simplicity we assume $(l_r + 1), l_m, l \leq l_p$. Messages are interpreted as values in $\{0,1\}^{l_m}$. We can sign longer messages by first applying a collision-resistant hash function $h : \{0,1\}^* \rightarrow \{0,1\}^{l_m}$. In the following we often interpret binary strings as values in $\mathbb{Z}$.

- ***Gen***$(1^\kappa)$: outputs an RSA modulus $n$ with $|n|_2 = l_n$ and its factorization $SK = (p, q)$. For simplicity we restrict ourselves to the case $p = 2p' + 1$, $q = 2q' + 1$ where $p, p', q, q'$ are primes and the modulus is balanced, i.e. $l_p = |p|_2 = |q|_2$. Next ***Gen***$(1^\kappa)$ chooses three generators $e, f, g$ of $\mathcal{QR}_n$ . Finally, it draws $k \in_R \mathcal{K}$ and $s \in_R \{0,1\}^{l_X}$ and publishes $PK = (n, e, f, g, k, s)$.
- ***Sign***$(SK, m)$: chooses $r \in_R \{0,1\}^{l_r}$ and $X \in_R \{0,1\}^{l_X}$. Next, it computes

$$z = (ef^m g^r)^{1/T(X)} \bmod n.$$

  The final signature is $\sigma = (z, X, r)$
- ***Verify***$(PK, m, \sigma)$: checks if it holds for $(z, X, r)$ that

$$z^{T(X)} \stackrel{?}{=} ef^m g^r \bmod n.$$

## 4 Strong Existential Unforgeability of $\mathcal{S}$

**Theorem 1.** *Assume the $(t_{RSA}, \epsilon_{RSA})$-RSA assumption holds. Then, $\mathcal{S}$ is $(q, t, \epsilon)$-secure against adaptive chosen message attacks provided that*

$$q = q_{RSA}, \quad t \approx t_{RSA},$$
$$\epsilon \leq 9ql_X\epsilon_{RSA}/2 + 9 \cdot 2^{-(l_p-2)} + 3 \cdot 2^{-l_o} + 3q^2/2^{l_X} + 3(ql_X)^2(l+2)/2^l.$$

*Proof.* Assume that $\mathcal{A}$ is a forger that $(q, t, \epsilon)$-breaks the strong existential unforgeability of $\mathcal{S}$. Then, we can construct a simulator $\mathcal{B}$ that breaks the RSA assumption in time $t_{\text{RSA}}$ with probability $\epsilon_{\text{RSA}}$.

We assume that the adversary makes $q$ queries $m_1, \ldots, m_q$ and receives $q$ responses $(z_1, X_1, r_1), \ldots, (z_q, X_q, r_q)$. We differentiate between three types of forgeries $(m^*, (z^*, X^*, r^*))$.

- In a **Type I Forgery**, there exists a non-empty set $J \subseteq [1; q]$ such that for all $j \in J$ it holds that $T(X^*) = T(X_j)$. Also, there exists at least one $j' \in J$ such that $r^* \neq r_{j'}$.
- For a **Type II Forgery**, there exists a non-empty set $J \subseteq [1; q]$ such that for all $j \in J$ it holds that $T(X^*) = T(X_j)$ but there exists no $j' \in J$ with $r^* \neq r_{j'}$.

Observe that if $t$ is collision-free for the $X_i$ with $i \in [1; q]$ and their prefixes it follows that $|J| = 1$ and $X^* = X_{j'}$ for Type I and Type II forgeries.

- Finally, for a **Type III Forgery** there exists no $j \in [1; q]$ such that $T(X^*) = T(X_j)$.

We let $\mathcal{B}$ guess upfront (with probability at least $\frac{1}{3}$) which type of forgery $\mathcal{A}$ is going to output. Theorem 1 then follows by a standard hybrid argument. Suppose $\mathcal{B}$ is given the RSA challenge instance $(u, \alpha, n)$. Let $\Pr[S_i]$ denote the success probability of an attacker to successfully forge signatures in $\text{Game}_i$.

**Type I Forger.** Suppose $\mathcal{B}$ correctly guesses that $\mathcal{A}$ outputs Type I Forgeries.

**$\text{Game}_0$.** This is the original attack game. By assumption, $\mathcal{A}$ $(q, t, \epsilon)$-breaks $\mathcal{S}$ when interacting with the signing oracle $\mathcal{O}(SK, \cdot)$. We have that,

$$\Pr[S_0] = \epsilon. \tag{1}$$

**$\text{Game}_1$.** In this game $\mathcal{B}$ computes the $X_i$ upfront. First it draws $v \in_R [1; q]$ and $q$ values $X_1, \ldots, X_q \in_R \{0,1\}^{l_X}$. Next it draws $s \in \{0,1\}^{l_X}$ uniformly at random from the set of values with $t(X_v) = \texttt{nextprime}(f_k(X_v) \oplus s) = \alpha$. Since $\mathcal{A}$ does not know $\alpha$ the values are distributed exactly as in the previous game.

$$\Pr[S_1] \geq \Pr[S_0]. \tag{2}$$

**$\text{Game}_2$.** In this game, $\mathcal{B}$ aborts if the $X_i$ are not all pairwisely different. By a union bound we get

$$\Pr[S_2] \geq \Pr[S_1] - q^2/2^{l_X}. \tag{3}$$

**$\text{Game}_3$.** Now, $\mathcal{B}$ aborts if there is a collision in the prime mapping function, meaning that there exists $X_i^{(j)}, X_{i'}^{(j')}$ with $i, i' \in [1; q]$, $j, j' \in [1; l_X]$, and $X_i^{(j)} \neq X_{i'}^{(j')}$ such that $t(X_i^{(j)}) = t(X_{i'}^{(j')})$. Altogether we must consider $ql_X$ different inputs (the $l_X$ prefixes of the $q$ $X_i$ values). By the prime number theorem the number of primes in the interval $[1; 2^l]$ is larger than $2^l/(l+2)$ for $l > 6$ [24]. By a union bound we get that

$$\Pr[S_3] \geq \Pr[S_2] - (ql_X)^2(l+2)/2^l. \tag{4}$$

**$\text{Game}_4$.** This game differs from the previous one in the way the elements $e, f, g \in \mathcal{QR}_n$ are chosen. First, $\mathcal{B}$ draws three random elements $w_e, w_f, w_g$ from $\mathbb{Z}_{(n-1)/4}$ and $c \in_R \{0,1\}^{l_r}$. Next, $\mathcal{B}$ computes $e$, $f$, and $g$ as

$$\begin{aligned} e &= u^{2\left(w_e \prod_{i=1}^{q} T(X_i) + c \prod_{i=1, i\neq v}^{q} T(X_i)\right)} \bmod n \\ f &= u^{2 w_f \prod_{i=1}^{q} T(X_i)} \bmod n \\ g &= u^{2\left(w_g \prod_{i=1}^{q} T(X_i) - \prod_{i=1, i\neq v}^{q} T(X_i)\right)} \bmod n. \end{aligned}$$

Let us now show that the so generated values are indistinguishable from the real attack game. Without loss of generality let $p' > q'$. It holds that $(n-1)/4 = p'q' + (p'+q')/2 > p'q'$ which implies that

$$\Pr[x \in_R \mathbb{Z}_{(n-1)/4},\ x \notin \mathbb{Z}_{p'q'}] = \frac{(p'+q')}{(2p'q'+p'+q')} < \frac{1}{q'+1} < 2^{-(l_p-2)}.$$

So, the distribution of values uniformly drawn from $\mathbb{Z}_{(n-1)/4}$ is statistically close to those drawn from $\mathbb{Z}_{p'q'}$. Therefore $e, f, g$ are indistinguishable (with the same

probability) from random group elements in $\mathcal{QR}_n$. By a union bound we get that,

$$\Pr[S_4] \geq \Pr[S_3] - 3 \cdot 2^{-(l_p-2)}. \tag{5}$$

**Game$_5$.** Now $\mathcal{B}$ simulates the signing oracle. For each query $m_j$ with $j \in \{1, \ldots, q\} \backslash \{v\}$, $\mathcal{B}$ computes a random $r_j \in \{0,1\}^{l_r}$. Then $\mathcal{B}$ outputs the signature $\sigma_j = (z_j, X_j, r_j)$ where $z_j$ is computed as

$$\begin{aligned} z_j &= (ef^{m_j} g^{r_j})^{\frac{1}{T(X_j)}} \\ &= u^{2(w_e + w_f m_j + w_g r_j) \prod_{i=1, i\neq j}^{q} T(X_i) + 2(c - r_j) \prod_{i=1, i\neq j, i\neq v}^{q} T(X_i)} \bmod n. \end{aligned}$$

In the case $j = v$ the signature is set to $\sigma_v = (z_v, X_v, c)$ and $z_v$ is computed as

$$z_v = (ef^{m_v} g^{c})^{\frac{1}{T(X_v)}} = u^{2(w_e + w_f m_v + w_g c) \prod_{i=1, i\neq v}^{q} T(X_i)} \bmod n.$$

These values are exactly distributed as in the previous game. So,

$$\Pr[S_5] \geq \Pr[S_4]. \tag{6}$$

**Game$_6$.** Now assume the adversary outputs the forgery $(m^*, (z^*, X^*, r^*))$. By assumption there exists a non-empty set $J \subseteq [1; q]$ such that for all $j \in J$ it holds that $T(X^*) = T(X_j)$. In the previous games we have ensured that the $X_i$ are distinct and collision-free, thus we have that $|J| = 1$. Hence it must be the case for (the only element) $j' \in J$ that $r^* \neq r_{j'}$. In this game, if $v \neq j'$ then $\mathcal{B}$ aborts. Therefore,

$$\Pr[S_6] \geq \Pr[S_5]/q. \tag{7}$$

**Game$_7$.** For the forgery it now holds that

$$\begin{aligned} (z^*)^{T(X^*)} &= (z^*)^{T(X_v)} \\ &= u^{2(w_e + w_f m^* + w_g r^*) \prod_{i=1}^{q} T(X_i) + 2(c - r^*) \prod_{i=1, i\neq v}^{q} T(X_i)} \bmod n. \end{aligned}$$

Since $t(X_v) = \alpha$ we equally have

$$\begin{aligned} &\left( z^* u^{-2(w_e + w_f m^* + w_g r^*) \prod_{i=1, i\neq v}^{q} T(X_i)} \right)^{\alpha \prod_{j=1}^{l_X - 1} t(X_v^{(j)})} \\ &= u^{2(c - r^*) \prod_{i=1, i\neq v}^{q} T(X_i)} \bmod n. \end{aligned} \tag{8}$$

In the last step we must consider the probability for the event that $\alpha | 2(c - r^*) \prod_{i=1, i\neq v}^{q} T(X_i)$. In this case the simulator aborts. Since the $X_i$ are all distinct and collision-free we get that $\alpha$ does not divide $\prod_{i=1, i\neq v}^{q} T(X_i)$. Since we only consider odd primes ($\gcd(\phi(n), \alpha) = 1$) we must finally only analyze whether $\alpha | (c - r^*)$. Observe that $c$ is hidden from $\mathcal{A}$'s view. However, $\mathcal{A}$ might just by chance compute $r^*$ such that $\alpha | (c - r^*)$. If that happens $\mathcal{B}$ aborts. Since 3 is the smallest prime $\alpha$ can take on, $\mathcal{A}$ will fail with probability at least $2/3$. Therefore,

$$\Pr[S_7] \geq 2\Pr[S_6]/3. \tag{9}$$

In this case, $\mathcal{B}$ can compute values $\beta_0, \beta_1 \in \mathbb{Z}$ with

$$\gcd\left(\alpha, 2(c-r^*)\prod_{i=1,i\neq v}^{q} T(X_i)\right) = \beta_0\alpha + \beta_1 2(c-r^*)\prod_{i=1,i\neq v}^{q} T(X_i) = 1.$$

It surely holds that

$$\begin{aligned} u &= u^{\beta_0\alpha+\beta_1 2(c-r^*)\prod_{i=1,i\neq v}^{q} T(X_i)} \\ &= u^{\beta_0\alpha}\cdot\left(z^* u^{-2(w_e+w_f m^*+w_g r^*)\prod_{i=1,i\neq v}^{q} T(X_i)}\right)^{\beta_1\alpha\prod_{j=1}^{l_X-1} t(X_v^{(j)})} \bmod n. \end{aligned}$$

Using Equation (8), $\mathcal{B}$ can compute a solution to the RSA challenge as

$$u^{1/\alpha} = u^{\beta_0}\cdot\left(z^* u^{-2(w_e+w_f m^*+w_g r^*)\prod_{i=1,i\neq v}^{q} T(X_i)}\right)^{\beta_1\prod_{j=1}^{l_X-1} t(X_v^{(j)})} \bmod n.$$

Finally,

$$\Pr[S_7] = \epsilon_{\mathrm{RSA}}. \tag{10}$$

Putting together Equations (1)-(10) we get that

$$\epsilon \leq 3q\epsilon_{\mathrm{RSA}}/2 + 3\cdot 2^{-(l_p-2)} + q^2/2^{l_X} + (ql_X)^2(l+2)/2^l.$$

**Type II Forger.** Now suppose $\mathcal{B}$ correctly expects $\mathcal{A}$ to be a Type II Forger. We only present the differences to the previous proof.

**Game$_4$.** Now $c$ is chosen uniformly from $[0; 2^{l_r}+2^{l_m}-2]$. Then, $\mathcal{B}$ draws three random elements $w_e, w_f, w_g$ from $\mathbb{Z}_{(n-1)/4}$ and computes $e$, $f$, and $g$ as

$$\begin{aligned} e &= u^{2\left(w_e\prod_{i=1}^{q} T(X_i)+c\prod_{i=1,i\neq v}^{q} T(X_i)\right)} \bmod n \\ f &= u^{2\left(w_f\prod_{i=1}^{q} T(X_i)-\prod_{i=1,i\neq v}^{q} T(X_i)\right)} \bmod n \\ g &= u^{2\left(w_g\prod_{i=1}^{q} T(X_i)-\prod_{i=1,i\neq v}^{q} T(X_i)\right)} \bmod n. \end{aligned}$$

Observe that only $f$ is constructed differently from the proof of Type I forgeries. By the same arguments as above $e, f, g$ are indistinguishable from random group elements in $\mathcal{QR}_n$. We again get that,

$$\Pr[S_4] \geq \Pr[S_3] - 3\cdot 2^{-(l_p-2)}. \tag{11}$$

**Game$_5$.** $\mathcal{B}$ simulates the signing oracle. For each query $m_j$ with $j \in \{1,\ldots,q\}\setminus\{v\}$, $\mathcal{B}$ computes a random $r_j \in \{0,1\}^{l_r}$. Then $\mathcal{B}$ outputs the signature $\sigma_j = (z_j, X_j, r_j)$ where $z_j$ is computed as

$$\begin{aligned} z_j &= (ef^{m_j}g^{r_j})^{\frac{1}{T(X_j)}} \\ &= u^{2(w_e+w_f m_j+w_g r_j)\prod_{i=1,i\neq j}^{q} T(X_i)+2(c-m_j-r_j)\prod_{i=1,i\neq j,i\neq v}^{q} T(X_i)} \bmod n. \end{aligned}$$

In the case $j = v$, $\mathcal{B}$ at first checks whether there exists a string $r_v \in \{0,1\}^{l_r}$ such that $c = m_v + r_v$. If not, $\mathcal{B}$ aborts. We can easily upper bound the probability that $\mathcal{B}$ will abort as

$$\begin{aligned}&\Pr\left[c \in_R [0; 2^{l_r} + 2^{l_m} - 2],\ m_v \leftarrow \mathcal{A},\ c - m_v \notin [0; 2^{l_r} - 1]\right]\\ \leq\ &\Pr\left[c \in_R [0; 2^{l_r} + 2^{l_m} - 2],\ c \notin [2^{l_m} - 1; 2^{l_r} - 1]\right] \leq 2^{-l_o}.\end{aligned}$$

Otherwise, the signature is set to $\sigma_v = (z_v, X_v, r_v)$ and $z_v$ is computed as

$$z_v = (ef^{m_v} g^{r_v})^{\frac{1}{T(X_v)}} = u^{2(w_e + w_f m_v + w_g r_v) \prod_{i=1, i\neq v}^{q} T(X_i)} \bmod n.$$

These values are exactly distributed as in the previous game.

Therefore,

$$\Pr[S_5] \geq \Pr[S_4] - 2^{-l_o}. \tag{12}$$

**Game$_6$.** Now assume the adversary outputs the forgery $(m^*, (z^*, X^*, r^*))$. By assumption there exists a non-empty set $J \subseteq [1; q]$ such that for all $j \in J$ it holds that $T(X^*) = T(X_j)$ but there exists no $j' \in J$ such that $r^* \neq r_{j'}$. Since the $X_i$ are pairwisely different and $t$ is collision-free we have that $|J| = 1$ what implies $X^* = X_{j'}$ and $r^* = r_{j'}$ for some $j' \in J$. If $v \neq j'$ then $\mathcal{B}$ aborts. So,

$$\Pr[S_6] \geq \Pr[S_5]/q. \tag{13}$$

**Game$_7$.** For the forgery it now holds that

$$\begin{aligned}(z^*)^{T(X^*)} &= (z^*)^{T(X_v)} = \left(ef^{m^*} g^{r^*}\right)\\ &= u^{2(w_e + w_f m^* + w_g r^*) \prod_{i=1}^{q} T(X_i) + 2(c - m^* - r^*) \prod_{i=1, i\neq v}^{q} T(X_i)} \bmod n\end{aligned}$$

or

$$\begin{aligned}&\left(z^* u^{-2(w_e + w_f m^* + w_g r^*) \prod_{i=1, i\neq v}^{q} T(X_i)}\right)^{\alpha \prod_{j=1}^{l_X - 1} t(X_v^{(j)})}\\ &= u^{2(c - m^* - r^*) \prod_{i=1, i\neq v}^{q} T(X_i)} \bmod n.\end{aligned}$$

In the last step we must consider the probability for the event that $\alpha | 2(c - m^* - r^*) \prod_{i=1, i\neq v}^{q} T(X_i)$. Observe that we must have $m^* \neq m_t$ because otherwise we have $\sigma^* = \sigma_t$ (and $\mathcal{A}$ did not output a valid forgery). So we surely have that $c - m^* - r^* \neq 0$. With the same arguments as above we must only check whether $\alpha | (c - m^* - r^*)$ in which case $\mathcal{B}$ will abort. Since $c$ is hidden from $\mathcal{A}$'s view the probability for this to happen is at most $1/3$. Thus,

$$\Pr[S_7] \geq 2\Pr[S_6]/3. \tag{14}$$

Otherwise $\mathcal{B}$ can compute values $\beta_0, \beta_1 \in \mathbb{Z}$ with

$$\begin{aligned}&\gcd\left(\alpha, 2(c - m^* - r^*) \prod_{i=1, i\neq v}^{q} T(X_i)\right)\\ &= \beta_0 \alpha + \beta_1 2(c - m^* - r^*) \prod_{i=1, i\neq v}^{q} T(X_i) = 1.\end{aligned}$$

It surely holds that

$$\begin{aligned} u &= u^{\beta_0\alpha+\beta_1 2(c-m^*-r^*)\prod_{i=1,i\neq v}^{q} T(X_i)} \\ &= u^{\beta_0\alpha} \cdot \left(z^* u^{-2(w_e+w_f m^*+w_g r^*)\prod_{i=1,i\neq v}^{q} T(X_i)}\right)^{\beta_1\alpha\prod_{j=1}^{l_X-1} t(X_v^{(j)})} \bmod n. \end{aligned}$$

Using Equation (15) $\mathcal{B}$ can compute a solution to the RSA challenge as

$$u^{1/\alpha} = u^{\beta_0} \cdot \left(z^* u^{-2(w_e+w_f m^*+w_g r^*)\prod_{i=1,i\neq v}^{q} T(X_i)}\right)^{\beta_1\prod_{j=1}^{l_X-1} t(X_v^{(j)})} \bmod n.$$

Finally,

$$\Pr[S_7] = \epsilon_{\mathrm{RSA}}. \tag{15}$$

Putting together Equations (1)-(15) we get that

$$\epsilon \leq 3q\epsilon_{\mathrm{RSA}}/2 + 3 \cdot 2^{-(l_p-2)} + 2^{-l_o} + q^2/2^{l_X} + (ql_X)^2(l+2)/2^l.$$

**Type III Forger.** Now suppose $\mathcal{B}$ correctly guesses that $\mathcal{A}$ is a Type III Forger. For convenience, let $\mathcal{X}$ be the set of all prefixes of the $X_i$, i.e. $\mathcal{X} = \{X_i^{(j)} | i \in [1;q],\ j \in [1;l_X]\}$. Clearly, $|\mathcal{X}| \leq ql_X$. Let $\mathcal{S}_{\mathcal{X}}$ denote the set of all binary strings $s = s_1 \dots s_k$ with $k \in [1;l_X]$ such that $s_1 \dots s_{k-1} \in \mathcal{X}$ but $s \notin \mathcal{X}$. If $\mathcal{X} = \emptyset$, i.e. $\mathcal{A}$ made no query, we define $\mathcal{S}_{\mathcal{X}} = \{0,1\}$. Surely, $|\mathcal{S}_{\mathcal{X}}| \leq ql_X$.

**Game$_1$.** In this game $\mathcal{B}$ computes the $X_i \in_R \{0,1\}^{l_X}$ for $i \in [1;q]$ upfront. Then $\mathcal{B}$ chooses $\bar{X} \in_R \mathcal{S}_{\mathcal{X}}$. Next, $\mathcal{B}$ draws $s \in \{0,1\}^{l_X}$ uniformly at random from the set of values with $t(\bar{X}) = \mathtt{nextprime}(f_k(\bar{X}) \oplus s) = \alpha$. Since $\mathcal{A}$ does not know $\alpha$ the values are distributed exactly as in the previous game.

$$\Pr[S_1] \geq \Pr[S_0]\ . \tag{16}$$

**Game$_2$.** Now the elements $e, f, g \in \mathcal{QR}_n$ are constructed. $\mathcal{B}$ draws three random elements $w_e, w_f, w_g$ from $\mathbb{Z}_{(n-1)/4}$ and $\mathcal{B}$ computes $e$, $f$, and $g$ as

$$\begin{aligned} e &= u^{2w_e\prod_{i=1}^{q} T(X_i)} \bmod n \\ f &= u^{2w_f\prod_{i=1}^{q} T(X_i)} \bmod n \\ g &= u^{2w_g\prod_{i=1}^{q} T(X_i)} \bmod n. \end{aligned}$$

With the same arguments as above we have

$$\Pr[S_2] \geq \Pr[S_1] - 3 \cdot 2^{-(l_p-2)}\ . \tag{17}$$

**Game$_3$.** Now $\mathcal{B}$ simulates the signing oracle. For each queries $m_j$ with $j \in \{1,\dots,q\}$, $\mathcal{B}$ computes $r_j \in_R \{0,1\}^{l_r}$ and outputs the signature $\sigma_j = (z_j, X_j, r_j)$ where $z_j$ is computed as

$$z_j = (ef^{m_j}g^{r_j})^{\frac{1}{T(X_j)}} = u^{2(w_e+w_f m_j+w_g r_j)\prod_{i=1,i\neq j}^{q} T(X_i)} \bmod n.$$

We have that

$$\Pr[S_3] = \Pr[S_2]. \tag{18}$$

**Game$_4$.** Let $(m^*, (z^*, X^*, r^*))$ be $\mathcal{A}$'s forgery. By assumption there exists no $j \in [1; q]$ such that $T(X^*) = T(X_j)$. This means that there is at least one new prime factor in $T(X^*)$. So there must be a prefix $X^{*(k)}$ with $k \in [1; l_X]$ that lies in $\mathcal{S}_X$. If $\bar{X} \neq X^{*(k)}$ $\mathcal{B}$ aborts. We have

$$\Pr[S_4] \geq \Pr[S_3]/ql_X. \tag{19}$$

**Game$_5$.** From the verification equation we know that

$$\begin{aligned}(z^*)^{T(X^*)} &= (z^*)^{\alpha \prod_{j=1, j\neq k}^{l_X} t(X^{*(j)})} = \left(ef^{m^*}g^{r^*}\right) \\ &= u^{2(w_e + w_f m^* + w_g r^*) \prod_{i=1}^{q} T(X_i)} \bmod n.\end{aligned}$$

Since $w_e, w_f, w_g$ are hidden from the adversary's view it holds with probability at least $2/3$ that $\gcd(\alpha, w_e + w_f m^* + w_g r^*) = 1$.

$$\Pr[S_5] \geq 2\Pr[S_4]/3\ . \tag{20}$$

Now we can compute two values $\beta_0, \beta_1 \in \mathbb{Z}$ with

$$\begin{aligned}&\gcd\left(\alpha, 2(w_e + w_f m^* + w_g r^*) \prod_{i=1}^{q} T(X_i)\right) \\ &= \beta_0 \alpha + \beta_1 2(w_e + w_f m^* + w_g r^*) \prod_{i=1}^{q} T(X_i) = 1\end{aligned}$$

and find a response to the RSA challenge as

$$u^{1/\alpha} = u^{\beta_0} (z^*)^{\beta_1 \prod_{j=1, j\neq k}^{l_X} t(X^{*(j)})} \bmod n\ .$$

Finally,

$$\Pr[S_5] = \epsilon_{\mathrm{RSA}}\ . \tag{21}$$

Putting together Equations (1)-(21) we get that

$$\epsilon \leq 3ql_X\epsilon_{\mathrm{RSA}}/2 + 3 \cdot 2^{-(l_p - 2)}.$$

## 5 Signing Message Blocks

Our new signature scheme can easily be extended to support signing message blocks. Assume we want to sign $u \in \mathbb{N}$ message $m_1, \ldots, m_u \in \{0,1\}^{l_m}$. The modified scheme looks as follows.

– ***Gen***$(1^\kappa)$: is exactly the same as in our main RSA signature scheme except that it now chooses $u+2$ generators $e, f_1, \ldots, f_u, g$ of $\mathcal{QR}_n$. Next, it computes

$$z = \left( eg^r \prod_{i=1}^{u} f_i^{m_i} \right)^{1/T(X)} \mod n.$$

The final signature is $\sigma = (z, X, r)$

– ***Verify***$(PK, m_1, \ldots, m_u, \sigma)$: to verify a signature $(z, X, r)$ the verifier checks whether

$$z^{T(X)} \stackrel{?}{=} eg^r \prod_{i=1}^{u} f_i^{m_i} \mod n.$$

The security proof for this scheme closely follows the proof for the main RSA signature scheme. In all three types of forgery, the $f_i$ are constructed exactly like $f$ above. The main difference is in $\text{Game}_4$ of the proof of Type II forgeries. We now have to draw $c$ from $[0; u(2^{l_m} - 1) + 2^{l_r} - 1]$. In $\mathbf{Game}_5$ we have to upper bound the probability for the event that given the attacker's query $(m_{v,1}, \ldots, m_{v,u})$ we cannot find $r_v \in \{0,1\}^{l_r}$ such that $r_v = c - \sum_{i=1}^{u} m_i$: $Pr[\text{Game}_5] \geq Pr[\text{Game}_4] - u2^{-l_o}$. In $\mathbf{Game}_7$ (where $X^* = X_t$ and $r^* = r_t$ for some $t \in [1; q]$), we then have to analyze whether $\alpha | (c - \sum_{i=1}^{u} m_i^* - r^*)$. This again holds with probability at least $2/3$.

## 6 Protocol for Signing Committed Messages

In [8], Camenisch-Lysyanskaya also presented an interactive zero-knowledge protocol between a signer and a user $u$ for issuing signatures on committed values. This protocol can easily be adapted to our signature scheme. The common inputs to this protocol are the public values $(n, k, s)$ and the generators $e, f, g \in \mathcal{QR}_n$ of the RSA signature scheme. Additionally $u$ provides a public key to a commitment scheme that consists of an RSA modulus $n_u$, two generators $e_u, f_u$ of $\mathcal{QR}_{n_u}$, and a commitment [12,16] $C_u$ of $m \in \{0,1\}^{l_m}$ under randomness $w_u$: $C_u = f_u^m g_u^{w_u} \mod n_u$. In the first step the user generates a commitment on $m$ using the signer's parameters and a new randomness $w$ such that $C = f^m g^w \mod n$. Additionally $u$ delivers three non-interactive zero knowledge proofs. The first one proves that $C$ and $C_u$ commit to the same value [9]. The second proves knowledge of $m$ and $w$ [15]. The third proof shows that $m \in \{0,1\}^{l_m}$ and $w \in \{0,1\}^{l_r - 1}$ [3,9]. Next the signer generates a signature on $C$ by choosing $r' \in_R \{0,1\}^{l_r - 1}$ and $X \in_R \{0,1\}^{l_X}$ and computing

$$z = \left( C g^{r'} e \right)^{1/T(X)} \mod n.$$

The values $(z, X, r')$ are given to $u$. Finally $u$ can compute a signature on $m$ as $(z, X, r' + w)$.

## 7 Extensions

Our design methodology can be transferred to the existing SRSA signature schemes by Fischlin [14] and Zhu [25, 26]. This is because, similar to the Camenisch-Lysyanskaya scheme, their security proofs also consider three forgers among which two actually reduce to the RSA assumption. The main tasks consists of substituting the prime component of the signature with $X \in_R \{0,1\}^{l_X}$. Moreover, we can also use our technique to build a Cramer-Shoup-like RSA signature scheme. The proof for the Cramer-Shoup signature scheme [11] also considers three different forgers but only one of them actually reduces to the SRSA assumption. Similar to before we can modify the Cramer-Shoup scheme by substituting the prime element in the signature with $T(X)$. In general, our modifications have the consequence that signature verification takes much more time in the RSA schemes because the prime exponents have to be *generated* using $t(\cdot)$ what includes costly primality tests.

## 8 Conclusion

In this work we have presented a new signature scheme that is secure under the RSA assumption. The scheme can easily be extended to sign several blocks of messages. Additionally it supports protocols to issue signatures on committed values. As a drawback the signature length is longer (by one component) than the Hohenberger-Waters scheme's.

## References

1. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
2. Bellare, M., Rogaway, P.: The exact security of digital signatures - how to sign with rsa and rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
3. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
4. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Atluri, V., Pfitzmann, B., McDaniel, P.D. (eds.) ACM Conference on Computer and Communications Security, pp. 132–145. ACM, New York (2004)
5. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
6. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Balancing accountability and privacy using e-cash (extended abstract). In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 141–155. Springer, Heidelberg (2006)
7. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)

8. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
9. Camenisch, J., Michels, M.: Separability and efficiency for generic group signature schemes. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 413–430. Springer, Heidelberg (1999)
10. Cramer, R., Damgård, I.B.: New generation of secure and practical rsa-based signatures. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 173–185. Springer, Heidelberg (1996)
11. Cramer, R., Shoup, V.: Signature schemes based on the Strong RSA assumption. ACM Trans. Inf. Syst. Secur. 3(3), 161–185 (2000)
12. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)
13. Dwork, C., Naor, M.: An efficient existentially unforgeable signature scheme and its applications. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 234–246. Springer, Heidelberg (1994)
14. Fischlin, M.: The cramer-shoup strong-rsasignature scheme revisited. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 116–129. Springer, Heidelberg (2002)
15. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
16. Fujisaki, E., Okamoto, T.: A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 32–46. Springer, Heidelberg (1998)
17. Gennaro, R., Halevi, S., Rabin, T.: Secure hash-and-sign signatures without the random oracle. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 123–139. Springer, Heidelberg (1999)
18. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
19. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (2008)
20. Hohenberger, S., Waters, B.: Realizing hash-and-sign signatures under standard assumptions. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 333–350. Springer, Heidelberg (2009)
21. Hohenberger, S., Waters, B.: Short and stateless signatures from the RSA assumption. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 654–670. Springer, Heidelberg (2009)
22. Naccache, D., Pointcheval, D., Stern, J.: Twin signatures: an alternative to the hash-and-sign paradigm. In: ACM Conference on Computer and Communications Security, pp. 20–27 (2001)
23. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2), 120–126 (1978)
24. Rosser, B.: Explicit bounds for some functions of prime numbers. American Journal of Mathematics 63(1), 211–232 (1941)
25. Zhu, H.: New digital signature scheme attaining immunity to adaptive-chosen message attack. Chinese Journal of Electronics 10(4), 484–486 (2001)
26. Zhu, H.: A formal proof of Zhu's signature scheme. Cryptology ePrint Archive, Report 2003/155 (2003), http://eprint.iacr.org/

# Appendix

## A The Camenisch-Lysyanskaya Signature Scheme

- ***Gen***$(1\kappa)$: computes a balanced RSA modulus $n = pq$ with $p = 2p' + 1$ and $q = 2q' + 1$ for primes $p, q, p', q'$. Then it chooses three generators $a, b, d$ of $\mathcal{QR}_n$. It publishes $PK = (n, a, b, d)$ and sets $SK = (p, q)$.
- ***Sign***$(SK, M)$: to sign a message $M \in \{0,1\}^{l_m}$ the signing algorithm at first chooses a random $l_w$ bit prime $t$ with $l_w \geq l_m + 2$. Then it chooses a random $r \in \{0,1\}^{l_m + |n|_2 + l}$ where $l = l(\kappa)$ is security parameter. Finally it computes $v = (ab^M d^r)^{1/w} \bmod n$ and outputs $\sigma = (w, r, v)$.
- ***Verify***$(PK, M, \sigma)$: checks, given a signature $\sigma = (w, r, v)$, whether $v^w = ab^M d^r \bmod n$ and if $w$, $r$ and $v$ have the correct bitsize.

## B The Hohenberger-Waters Signature Scheme

We subsequently present the Hohenberger-Waters signature scheme that is secure against generic chosen message attacks under the RSA assumption [21].

- ***Gen***$(1^\kappa)$: Choose a key $s$ for a pseudo-random function $v : \{0,1\}^* \rightarrow \{0,1\}^l$. Choose an RSA modulus $n$ with $n = pq$. Choose $u \in_R \mathbb{Z}_n^*$ and $r \in_R \{0,1\}^l$. Next, define the function $v' : \{0,1\}^* \rightarrow \{0,1\}^l$ as $v'(M) := \texttt{nextprime}(v_s(M) \oplus r)$. Set $SK = (p, q)$ and publish $PK = (u, n, s, r)$.
- ***Sign***$(SK, M)$: To sign a message $M \in \{0,1\}^*$ compute the signature as $\sigma = u^{1/(\prod_{i=1}^{l} v'(M^{(i)}))} \bmod n$.
- ***Verify***$(PK, M, \sigma)$: If it holds that $\sigma^{\prod_{i=1}^{l} v'(M^{(i)})} = u \bmod n$ output 1, else output 0.

# Fair Blind Signatures without Random Oracles

Georg Fuchsbauer and Damien Vergnaud

École normale supérieure, LIENS - CNRS - INRIA, Paris, France
http://www.di.ens.fr/{~fuchsbau,~vergnaud}

**Abstract.** A fair blind signature is a blind signature with revocable anonymity and unlinkability, i.e. an authority can link an issuing session to the resulting signature and trace a signature to the user who requested it. In this paper we first revisit the security model for fair blind signatures given by Hufschmitt and Traoré in 2007. We then give the first practical fair blind signature scheme with a security proof in the standard model. Our scheme satisfies a stronger variant of the Hufschmitt-Traoré model.

**Keywords:** Blind signatures, revocable anonymity, standard model, Groth-Sahai proof system.

## 1 Introduction

A blind signature scheme is a protocol for obtaining a signature from an issuer (signer) such that the issuer's view of the protocol cannot be linked to the resulting message/signature pair. Blind signatures are employed in privacy-related protocols where the issuer and the message author are different parties (e.g., e-voting or e-cash systems). However, blind signature schemes provide perfect unlinkability and could therefore be misused by dishonest users. Fair blind signatures were introduced by Stadler, Piveteau and Camenisch [SPC95] to prevent abuse of unlinkability. They allow two types of blindness revocation: linking a signature to the user who asked for it and identifying a signature that resulted from a given signing session. A security model for fair blind signatures was introduced by Hufschmitt and Traoré [HT07].

We first revisit this security model and propose a stronger variant. We then present the first efficient fair blind signature scheme with a standard-model security proof (i.e. without resorting to the random-oracle heuristic) in the strengthened model. We make extensive use of the non-interactive proof system due to Groth and Sahai [GS08] and of the *automorphic signatures* recently introduced by Fuchsbauer [Fuc09]; we do not rely on interactive assumptions. We note that this is an extended abstract and refer to the full version [FV10] for detailed proofs and an improved scheme based on recent results in [Fuc09].

### 1.1 Prior Work

The concept of *blind signatures* was introduced by Chaum in [Cha83]. A blind signature scheme is a cryptographic primitive that allows a user to obtain from

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 16–33, 2010.

the *issuer* (signer) a digital signature on a message of the user's choice in such a way that the issuer's view of the protocol cannot be linked to the resulting message/signature pair. Blind signatures have numerous applications including e-cash: they prevent linking withdrawals and payments made by the same customer. However, the impossibility of this linking might lead to frauds (money laundering, blackmailing, ...); some applications therefore require means to identify the resulting signature from the transcript of a signature-issuing protocol or to link a message/signature pair to user who requested it.

*Fair blind signatures* were introduced by Stadler, Piveteau and Camenisch in [SPC95] to provide these means. Several schemes have been proposed since then [SPC95, AO01, HT07] with applications to e-cash [GT03] or e-voting [CGT06]. In [HT07], Hufschmitt and Traoré presented the first formal security model for fair blind signatures and a scheme based on bilinear maps satisfying it in the random oracle model under an interactive assumption. In a recent independent work, Rückert and Schröder [RS10] proposed a *generic* construction of fair *partially* blind signatures [AF96].

### 1.2 Our Contribution

As a first contribution, we strengthen the security model proposed in [HT07]. In our model, opening a transcript of an issuing session not only reveals information to identify the resulting signature, but also the user that requested it.

We give a definition of blindness analogously to [Oka06], but additionally provide tracing oracles to the adversary; in contrast to [HT07], this models *active* adversaries. We propose a traceability notion that implies the original one. Finally, we formalize the non-frameability notions analogously to [BSZ05], where it is the adversary's task to output a framing signature (or transcript) *and a proof*. (In [HT07] the experiment produces the proof, limiting thus the adversary.) We believe that our version of signature non-frameability is more intuitive: no corrupt issuer can output a transcript, an opening framing a user, and a proof. (In [HT07] the adversary must output a message/signature pair such that an honest transcript opens to it.) (See §2.3 for the details.)

In 2008, Groth and Sahai [GS08] proposed a way to produce efficient non-interactive zero-knowledge (NIZK) and non-interactive witness-indistinguishable (NIWI) proofs for (algebraic) statements related to groups equipped with a bilinear map. In particular, they give proofs of satisfiability of *pairing-product equations* (cf. §4.2 and [BFI+10] for efficiency improvements for proof verification). In [Fuc09], Fuchsbauer introduced the notion of *automorphic signatures* whose verification keys lie in the message space, messages and signatures consist of group elements only, and verification is done by evaluating a set of pairing-product equations (cf. §5). Among several applications, he constructed an (automorphic) blind signature in the following way: the user commits to the message, and gives the issuer a randomized message; the issuer produces a "pre-signature" from which the user takes away the randomness to recover a signature. The actual signature is then a Groth-Sahai NIWI proof of knowledge of a signature, which guarantees unlinkability to the issuing.

In this paper, we modify Fuchsbauer's blind signature scheme in order to construct the first practical fair blind signature scheme with a security reduction in the standard model. Our security analysis does not introduce any new computational assumptions and relies only on falsifiable assumptions [Nao03] (cf. §3). First, we extend Fuchsbauer's automorphic signature so it can sign three messages at once. Then, to achieve blindness even against adversaries provided with tracing oracles, we use Groth's technique from [Gro07] to achieve CCA-anonymous group signatures: instead of just committing to the tracing information, we additionally encrypt it (using Kiltz' tag-based encryption scheme [Kil06]) and provide NIZK proofs of consistency with the commitments. In order to achieve the strengthened notion of non-frameability, we construct simulation-sound NIZK proofs of knowledge of a Diffie-Hellman solution which consist of group elements only and are verified by checking a set of pairing-product equations (i.e. they are Groth-Sahai compatible).

Since messages and signatures consist of group elements only and their verification predicate is a conjunction of pairing-product equations, our fair blind signatures are Groth-Sahai compatible themselves which makes them perfectly suitable to design efficient fair e-cash systems following the approach proposed in [GT03]. In addition, our scheme is compatible with the "generic" variant[1] of Votopia [OMA+99] proposed by Canard, Gaud and Traoré in [CGT06]. Combined with a suitable mix-net (e.g. [GL07]), it provides a practical electronic voting protocol in the standard model including public verifiability, and compares favorably with other similar systems in terms of computational cost.

## 2 The Model

### 2.1 Syntax

**Definition 1.** *A* fair blind signature scheme *is a 10-tuple*

$$(\mathsf{Setup}, \mathsf{IKGen}, \mathsf{UKGen}, \mathsf{Sign}, \mathsf{User}, \mathsf{Ver}, \mathsf{TrSig}, \mathsf{TrId}, \mathsf{ChkSig}, \mathsf{ChkId})$$

*of (interactive) (probabilistic) polynomial-time Turing machines ((P)PTs):*

Setup *is a PPT that takes as input an integer $\lambda$ and outputs the* parameters *pp and the* revocation key *rk. We call $\lambda$ the* security parameter.

IKGen *is a PPT that takes as input the parameters pp and outputs a pair $(ipk, isk)$, the* issuer's public and secret key.

UKGen *is a PPT that takes as input the parameters pp and outputs a pair $(upk, usk)$, the* user's public and secret key.

Sign *and* User *are interactive PPTs such that* User *takes as inputs pp, the issuer's public key ipk, the user's secret key usk and a bit string $m$;* Sign *takes as input pp, the issuer's secret key isk and user public key upk.* Sign *and* User *engage in the* signature-issuing protocol *and when they stop,* Sign *outputs* `completed` *or* `not-completed` *while* User *outputs $\perp$ or a bit string $\sigma$.*

[1] This variant was used during the French referendum on the European Constitution in May 2005.

Ver *is a deterministic PT (DPT) that on input the parameters* pp*, an issuer public key* ipk *and a pair of bit strings* $(m, \sigma)$ *outputs either* 0 *or* 1*. If it outputs* 1 *then* $\sigma$ *is a* valid signature *on the* message $m$*.*

TrSig *is a DPT that on input* pp*, an issuer public key* ipk*, a transcript* trans *of a signature-issuing protocol and a revocation key* rk *outputs three bit strings* $(\mathrm{upk}, \mathrm{id}_\sigma, \pi)$*.*

TrId *is a DPT that on input* pp*, an issuer public key* ipk*, a message/signature pair* $(m, \sigma)$ *for* ipk *and a revocation key* rk *outputs two bit strings* $(\mathrm{upk}, \pi)$*.*

ChkSig *is a DPT that on input* pp*, an issuer public key* ipk*, a transcript of a signature issuing protocol, a pair message/signature* $(m, \sigma)$ *for* ipk *and three bit strings* $(\mathrm{upk}, \mathrm{id}_\sigma, \pi)$*, outputs either* 0 *or* 1*.*

ChkId *is a DPT that on input* pp*, an issuer public key* ipk*, a message/signature pair* $(m, \sigma)$ *for* ipk *and two bit strings* $(\mathrm{upk}, \pi)$*, outputs either* 0 *or* 1*.*

*For all* $\lambda \in \mathbb{N}$*, all pairs* $(pp, \mathrm{rk})$ *output by* $\mathsf{Setup}(\lambda)$ *all pairs* $(ipk, isk)$ *output by* $\mathsf{IKGen}(pp)$*, and all pairs* $(\mathrm{upk}, usk)$ *output by* $\mathsf{UKGen}(pp)$*:*

1. *if* Sign *and* User *follow the signature-issuing protocol with input* $(pp, isk, \mathrm{upk})$ *and* $(pp, usk, ipk, m)$ *respectively, then* Sign *outputs* `completed` *and* User *outputs a bit string* $\sigma$ *that satisfies* $\mathsf{Ver}(ipk, (m, \sigma)) = 1$*;*
2. *on input* ipk*, the transcript* trans *of the protocol and* rk*,* TrSig *outputs three bit strings* $(\mathrm{upk}, \mathrm{id}_\sigma, \pi)$ *s.t.* $\mathsf{ChkSig}(pp, ipk, trans, (m, \sigma), (\mathrm{upk}, \mathrm{id}_\sigma, \pi)) = 1$*;*
3. *on input* ipk*, the pair* $(m, \sigma)$ *and* rk*,* TrId *outputs two bit strings* $(\mathrm{upk}, \pi)$ *such that* $\mathsf{ChkId}(pp, ipk, (m, \sigma), (\mathrm{upk}, \pi)) = 1$*.*

### 2.2 Security Definitions

To define the security notions for fair blind signatures, we use a notation similar to the one in [BSZ05] used in [HT07]:

*HU* denotes the set of honest users and *CU* is the set of corrupted users.

AddU is an *add-user* oracle. The oracle runs $(upk, usk) \leftarrow \mathsf{UKGen}(pp)$, adds *upk* to *HU* and returns it to the adversary.

CrptU is a *corrupt-user* oracle. The adversary calls it with a pair $(upk, usk)$ and *upk* is added to the set *CU*.

USK is a *user-secret-key* oracle enabling the adversary to obtain the private key *usk* for some $upk \in HU$. The oracle transfers *upk* to *CU* and returns *usk*.

User is an *honest-user* oracle. The adversary impersonating a corrupt issuer calls it with $(upk, m)$. If $upk \in HU$, the experiment simulates the honest user holding *upk* running the signature issuing protocol with the adversary for message $m$. If the issuing protocol completed successfully, the adversary is given the resulting signature. The experiment keeps a list *Set* with entries of the form $(upk, m, trans, \sigma)$, to record an execution of User, where *trans* is the transcript of the issuing session and $\sigma$ is the resulting signature. (Note that only valid $\sigma$'s (i.e. the protocol was successful) are written to *Set*.

Sign is a *signing* oracle. The adversary impersonating a corrupt user can use it to run the issuing protocol with the honest issuer. The experiment keeps a list *Trans* in which the transcripts $trans_i$ resulting from Sign calls are stored.

$\mathsf{Challenge}_b$ is a *challenge* oracle, which (w.l.o.g.) can only be called once. The adversary provides two user public keys $upk_0$ and $upk_1$ and two messages $m_0$ and $m_1$. The oracle first simulates $\mathsf{User}$ on inputs $(pp, ipk, usk_b, m_b)$ and then, in a second protocol run, simulates $\mathsf{User}$ on inputs $(pp, ipk, usk_{1-b}, m_{1-b})$. Finally, the oracle returns $(\sigma_0, \sigma_1)$, the resulting signatures on $m_0$ and $m_1$.

$\mathsf{TrSig}$ (resp. $\mathsf{TrId}$) is a *signature (resp. identity) tracing* oracle. When queried on the transcripts (or messages) emanating from a $\mathsf{Challenge}$ call, they return $\perp$.

Figure 1 formalizes the experiments for the following security notions:

**Blindness.** Not even the issuer with access to tracing oracles can link a message/signature pair to the signature-issuing session it stems from.

**Identity Traceability.** No coalition of users can produce a set of signatures containing signatures which cannot be linked to an identity.

**Signature Traceability.** No coalition of users is be able to produce a message/signature pair which is not traced by any issuing transcript or two pairs which are traced by the same transcript.

**Identity Non-Frameability.** No coalition of issuer, users and tracing authority should be able to provide a signature and a proof that the signature opens to an honest user who did not ask for the signature.

**Signature Non-Frameability.** No coalition of issuer, users and tracing authority should be able to provide a transcript that either wrongfully opens to an honest signature or an honest user.

We say that a fair blind signature achieves *blindness* if for all p.p.t. adversaries $\mathcal{A}$, the following is negligible: $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{blind-1}} = 1] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{blind-0}} = 1] - \frac{1}{2}$. The remaining security notions are achieved if for all p.p.t. $\mathcal{A}$, the probability that the corresponding experiment returns 1 is negligible.

### 2.3 A Note on the Hufschmitt-Traoré Security Notions

**Blindness.** In [HT07], the challenge oracle (called "Choose") is defined as follows: the adversary provides two user public keys $upk_0$ and $upk_1$ and a message, and obtains a signature under $upk_b$. This gives a weak security guarantee, as the adversary—who should impersonate the issuer—cannot actively participate in the issuing of the challenge signature. We define our oracle in the spirit of [Oka06]: the adversary chooses two users (and messages) which interact with him in random order; he gets to see both resulting signatures and has to determine the order of issuing.

**Traceability Notions.** Intuitively, identity traceability means that no coalition of users and the authority can create a message/signature pair that is not traceable to a user, which is what was formalized in [HT07].

We propose the following experiment leading to a stronger notion: the adversary gets the authority's key and impersonates corrupt users, who, via the $\mathsf{Sign}$ oracle can request signatures from the honest issuer. The latter is simulated by the experiment and keeps a set *Trans* of transcripts of oracle calls. Eventually,

$\mathbf{Exp}^{\text{blind-b}}_{\mathcal{A}}(\lambda)$

$(pp, rk) \leftarrow \mathsf{Setup}(1^\lambda)$; $(ipk, isk) \leftarrow \mathsf{IKGen}(pp)$
$b' \leftarrow \mathcal{A}(pp, ipk, isk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{Challenge}_b, \mathsf{User}, \mathsf{TrSig}, \mathsf{TrId})$
return $b'$

$\mathbf{Exp}^{\text{IdTrac}}_{\mathcal{A}}(\lambda)$

$(pp, rk) \leftarrow \mathsf{Setup}(1^\lambda)$; $(ipk, isk) \leftarrow \mathsf{IKGen}(pp)$; $Trans \leftarrow \emptyset$
$(m_1, \sigma_1, \ldots, m_n, \sigma_n) \leftarrow \mathcal{A}(pp, ipk, rk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{Sign})$
for $i = 1 \ldots |Trans|$ do $(upk_i, id_i, \pi_i) \leftarrow \mathsf{TrSig}(pp, rk, ipk, trans_i)$
for $i = 1 \ldots n$ do $(upk'_i, \pi'_i) \leftarrow \mathsf{TrId}(pp, rk, ipk, m_i, \sigma_i)$
if $\exists\, i : upk'_i = \perp$ or $\mathsf{ChkId}(pp, ipk, (m_i, \sigma_i), upk'_i, \pi'_i) = 0$ then return 1
if some $upk$ appears more often in $(upk'_1, \ldots, upk'_n)$ than in
$(upk_1, \ldots, upk_{|Trans|})$ then return 1; else return 0

$\mathbf{Exp}^{\text{IdNF}}_{\mathcal{A}}(\lambda)$

$(pp, rk) \leftarrow \mathsf{Setup}(1^\lambda)$; $(ipk, isk) \leftarrow \mathsf{IKGen}(pp)$
$Set \leftarrow \emptyset$; $HU \leftarrow \emptyset$; $CU \leftarrow \emptyset$
$(upk, m, \sigma, \pi) \leftarrow \mathcal{A}(pp, ipk, isk, rk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{User})$
if $\mathsf{Ver}(pp, ipk, m, \sigma) = 0$ or $\mathsf{ChkId}(pp, ipk, m, \sigma, upk, \pi) = 0$ then return 0
if $(upk, m, \cdot, \sigma) \notin Set$ and $upk \in HU$ then return 1; else return 0

$\mathbf{Exp}^{\text{SigTrac}}_{\mathcal{A}}(\lambda)$

$(pp, rk) \leftarrow \mathsf{Setup}(1^\lambda)$; $(ipk, isk) \leftarrow \mathsf{IKGen}(pp)$; $Trans \leftarrow \emptyset$
$(m_1, \sigma_1, m_2, \sigma_2) \leftarrow \mathcal{A}(pp, ipk, rk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{Sign})$
let $Trans = (trans_i)_{i=1}^n$; for $i = 1 \ldots n$ do $(upk_i, id_i, \pi_i) \leftarrow \mathsf{TrSig}(pp, rk, ipk, trans_i)$
if $\mathsf{Ver}(pp, ipk, m_1, \sigma_1) = 1$ and
$\forall\, i : \mathsf{ChkSig}(pp, ipk, trans_i, m_1, \sigma_1, upk_i, id_i, \pi_i) = 0$ then return 1
if $(m_1, \sigma_1) \neq (m_2, \sigma_2)$ and $\mathsf{Ver}(pp, ipk, m_1, \sigma_1) = 1$ and $\mathsf{Ver}(pp, ipk, m_2, \sigma_2) = 1$
and $\exists\, i : \mathsf{ChkSig}(pp, ipk, trans_i, m_1, \sigma_1, upk_i, id_i, \pi_i) =$
$= \mathsf{ChkSig}(pp, ipk, trans_i, m_2, \sigma_2, upk_i, id_i, \pi_i)) = 1$
then return 1; else return 0

$\mathbf{Exp}^{\text{SigNF}}_{\mathcal{A}}(\lambda)$

$(pp, rk) \leftarrow \mathsf{Setup}(1^\lambda)$; $(ipk, isk) \leftarrow \mathsf{IKGen}(pp)$
$Set \leftarrow \emptyset$; $HU \leftarrow \emptyset$; $CU \leftarrow \emptyset$
$(trans^*, m^*, \sigma^*, upk^*, id^*_\sigma, \pi^*) \leftarrow \mathcal{A}(pp, ipk, isk, rk : \mathsf{AddU}, \mathsf{CrptU}, \mathsf{USK}, \mathsf{User})$
let $Set = (upk_i, m_i, trans_i, \sigma_i)_{i=1}^n$
if $\exists\, i : trans^* \neq trans_i$ and $\mathsf{ChkSig}(pp, ipk, trans^*, m_i, \sigma_i, upk^*, id^*_\sigma, \pi^*) = 1$
then return 1
if $(\forall\, i : upk^* = upk_i \Rightarrow trans^* \neq trans_i)$
and $\mathsf{ChkSig}(\ldots, trans^*, m^*, \sigma^*, upk^*, id^*_\sigma, \pi^*) = 1$
then return 1; else return 0

**Fig. 1.** Security experiments for fair blind signatures

the adversary outputs a set of message/signature pairs. The experiment opens all transcripts to get a list of users to which signatures were issued. Another list of users is constructed by opening the returned signatures. The adversary wins if there exists a user who appears more often in the second list than in the first, or

if $\perp$ is in the second list, or if any of the proofs output by the opening algorithm do not verify. Note that the notion of [HT07] is implied by ours.

**Non-Frameability Notions.** Non-frameability means that not even a coalition of everyone else can "frame" an honest user. For example, no adversary can output a signature which opens to a user who did not participate in its issuing. In [HT07], the adversary outputs a message/signature pair, which is then opened by the experiment to determine if it "framed" a user. Analogously to [BSZ05] (who defined non-frameability for group signatures), we define a strictly stronger notion requiring the adversary to output an incriminating signature, an honest user, *and a valid proof* that the signature opens to that user. Note that only this formalization makes the $\pi$ output by the tracing algorithms a proof, as it guarantees that no adversary can produce a proof that verifies for a false opening.

Identity Non-Frameability. In [HT07], the adversary wins if it produces a pair $(m, \sigma)$ such that, when opened to $upk$, we have $(m, \sigma, upk) \notin Set$. This seems to guarantee *strong* unforgeability where an adversary modifying a signature returned by the experiment wins the game. This is however not the case in the scheme proposed in [HT07]: the final signature is a proof of knowledge of some values computed by the issuer made non-interactive by the Fiat-Shamir heuristic; hence from a given signature issuing session the user may derive several valid signatures on a message $m$. For that reason, the model in [HT07] considers two signatures different only if the underlying secrets are different. We adopt the same convention in this paper in that we consider two signatures equivalent if they have the same (public) *identifier*.

Signature Non-Frameability. Non-frameability of signature tracing intuitively means: even if everyone else colludes against an honest user, they cannot produce a transcript that opens to an honest signature. In the definition proposed in [HT07], the adversary plays the issuer in that he gets his secret key. However, he has no possibility to communicate with honest users since the *challenger* plays the issuer in the signature-issuing sessions with honest users and the adversary only gets the transcripts. His goal is to produce a *new* message/signature pair (one that does not emanate from a $\mathsf{User}$-oracle call) such that an honest transcript opens to it.

We give the following security notion which we think is more intuitive. No corrupt issuer can produce a transcript of an issuing session and one of the following: either a public key of an honest user and a proof that this user participated in the transcript whereas she did not; or a signature identifier of an honest signature coming from a different session and a proof that the transcript opens to it. Similarly to signatures we consider two transcripts equivalent if the contain the same user randomness and the same issuer randomness.

**Unforgeability.** Consider an adversary that breaks the classical security notion for blind signatures, one-more unforgeability, i.e. after $q - 1$ $\mathsf{Sign}$-oracle queries, he outputs $q$ signatures on different messages. We show that the adversary must have broken signature traceability: indeed since there are more signatures than

transcripts, either there is a signature which no transcripts points to, or there is a transcript that points to two signatures.

## 3 Assumptions

A (symmetric) *bilinear group* is a tuple $(p, \mathbb{G}, \mathbb{G}_T, e, G)$ where $(\mathbb{G}, \cdot)$ and $(\mathbb{G}_T, \cdot)$ are two cyclic groups of prime order $p$, $G$ is a generator of $\mathbb{G}$, and $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a non-degenerate bilinear map, i.e. $\forall U, V \in \mathbb{G}\ \forall a, b \in \mathbb{Z}\colon\ e(U^a, V^b) = e(U, V)^{ab}$, and $e(G, G)$ is a generator of $\mathbb{G}_T$.

The *Decision Linear (DLIN) Assumption* [BBS04], in $(p, \mathbb{G}, \mathbb{G}_T, e, G)$ states that given $(G^\alpha, G^\beta, G^{r\alpha}, G^{s\beta}, G^t)$ for random $\alpha, \beta, r, s \in \mathbb{Z}_p$, it is hard to decide whether $t = r + s$ or $t$ is random.

The following two assumptions were introduced by [FPV09] and [Fuc09], respectively. Under the *knowledge of exponent assumption* [Dam92], the first is equivalent to SDH [BB04] and the second is equivalent to computing discrete logarithms.

**Assumption 1 ($q$-DHSDH).** *Given* $(G, H, K, X = G^x) \in \mathbb{G}^4$ *and* $q - 1$ *tuples*

$$\left(A_i = (KG^{v_i})^{\frac{1}{x+d_i}},\ C_i = G^{d_i},\ D_i = H^{d_i},\ V_i = G^{v_i},\ W_i = H^{v_i}\right)_{i=1}^{q-1},$$

*for* $d_i, v_i \leftarrow \mathbb{Z}_p$*, it is hard to output a new tuple* $(A, C, D, V, W) \in \mathbb{G}^5$ *satisfying*

$$e(A, XC) = e(KV, G) \qquad e(C, H) = e(G, D) \qquad e(V, H) = e(G, W) \tag{1}$$

The next assumption states that, given $(G, H, T) \in \mathbb{G}^3$, it is hard to produce a non-trivial $(G^m, H^m, G^r, H^r)$ such that $G^m = T^r$.

**Assumption 2 (HDL).** *Given a random triple* $(G, H, T) \in \mathbb{G}^3$*, it is hard to output* $(M, N, R, S) \neq (1, 1, 1, 1)$ *such that*

$$e(R, T) = e(M, G) \qquad e(M, H) = e(G, N) \qquad e(R, H) = e(G, S) \tag{2}$$

## 4 Tools

We recall some tools from the literature which we use to construct our scheme.

### 4.1 A Signature Scheme to Sign Group Elements

We present the signature scheme from [Fuc09], which is secure against chosen-message attacks under Assumptions 1 and 2. Its message space is the set of *Diffie-Hellman pairs* $\mathcal{DH} := \{(A, B) \in \mathbb{G}^2 \mid \exists \alpha : A = G^\alpha, B = H^\alpha\}$ w.r.t. two fixed generators $G, H \in \mathbb{G}$. Note that $(A, B) \in \mathcal{DH}$ iff $e(A, H) = e(G, B)$.

**Scheme 1 ($\mathsf{Sig}_1$).**

$\mathsf{Setup}_1$ Given $(p, \mathbb{G}, \mathbb{G}_T, e, G)$, choose additional generators $H, K, T \in \mathbb{G}$.

$\mathsf{KeyGen}_1$ Choose $sk = x \leftarrow \mathbb{Z}_p$ and set $vk = G^x$.

$\mathsf{Sign}_1$ To sign $(M,N) \in \mathcal{DH}$ with secret key $x$, choose $d, r \leftarrow \mathbb{Z}_p$ and output

$$\left(A := (KT^r M)^{\frac{1}{x+d}}, C := G^d, D := H^d, R := G^r, S := H^r\right) ,$$

$\mathsf{Verify}_1$ $(A,C,D,R,S)$ is valid on $(M,N) \in \mathcal{DH}$ under public key $vk = X$ iff

$$e(A, XC) = e(KM, G)\, e(T, R) \qquad \begin{array}{l} e(C,H) = e(G,D) \\ e(R,H) = e(G,S) \end{array} \tag{3}$$

### 4.2 Groth-Sahai Proofs

We sketch the results of Groth and Sahai [GS08] on proofs of satisfiability of sets of equations over a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, G)$. Due to the complexity of their methodology, we present what is needed for our results and refer to the full version of [GS08] for any additional details.

We define a key for *linear commitments*. Choose $\alpha, \beta, r_1, r_2 \leftarrow \mathbb{Z}_p$ and define $U = G^\alpha$, $V = G^\beta$, $W_1 := U^{r_1}$, $W_2 := V^{r_2}$, and $W_3$ which is either

- soundness setting: $W_3 := G^{r_1+r_2}$ (which makes $\vec{\mathbf{u}}$ a binding key); or
- witness-indistinguishable setting: $W_3 := G^{r_1+r_2-1}$ (making $\vec{\mathbf{u}}$ a hiding key)

Under key $ck = (U, V, W_1, W_2, W_3)$, a commitment to a group element $X \in \mathbb{G}$ using randomness $(s_1, s_2, s_3) \leftarrow \mathbb{Z}_p^3$ is defined as

$$\mathsf{Com}\big(ck, X; (s_1, s_2, s_3)\big) := \left(U^{s_1} W_1^{s_3},\ V^{s_2} W_2^{s_3},\ XG^{s_1+s_2} W_3^{s_3}\right) .$$

In the soundness setting, given the *extraction key* $ek := (\alpha, \beta)$, the committed value can be extracted from a commitment $\mathbf{c} = (c_1, c_2, c_3)$. On the other hand, in the witness-indistinguishable (WI) setting, $\mathbf{c}$ is equally distributed for every $X$. The two settings are indistinguishable under the DLIN assumption.

A *pairing-product equation* is an equation for variables $\mathcal{Y}_1, \ldots, \mathcal{Y}_n \in \mathbb{G}$ of the form

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{Y}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(\mathcal{Y}_i, \mathcal{Y}_j)^{\gamma_{i,j}} \;=\; t_T \ ,$$

with $\mathcal{A}_i \in \mathbb{G}$, $\gamma_{i,j} \in \mathbb{Z}_p$ and $t_T \in \mathbb{G}_T$ for $1 \le i, j \le n$.

To prove satisfiability of a set of equations of this form, one first makes commitments to a satisfying witness (i.e. an assignment to the variables of each equation) and then adds a "proof" per equation. Groth and Sahai describe how to construct these: they are in $\mathbb{G}^{3\times 3}$ (or $\mathbb{G}^3$ when all $\gamma_{i,j} = 0$). In the soundness setting, if the proof is valid then $\mathsf{Extr}$ extracts the witness satisfying the equations. In the WI setting, commitments and proofs of different witnesses which both satisfy the same pairing-product equation are equally distributed.

### 4.3 Commit and Encrypt

In order to build CCA-anonymous group signatures, Groth [Gro07] uses the following technique: a group signature consists of linear commitments to a certified

signature and Groth-Sahai proofs that the committed values constitute a valid signature. CPA-anonymity follows from WI of GS proofs: once the commitment key has been replaced by a perfectly hiding one, a group signature reveals no information about the signer. However, in order to simulate opening queries in the WI setting, some commitments are doubled with a *tag-based encryption* under Kiltz' scheme [Kil06] and a Groth-Sahai NIZK proof that the committed and the encrypted value are the same. To produce a group signature, the user first chooses a key pair for a one-time signature scheme, uses the verification key as the tag for the encryption and the secret key to sign the group signature.

By $\mathbf{Sig}_{\mathrm{ot}} = (\mathsf{KeyGen}_{\mathrm{ot}}, \mathsf{Sign}_{\mathrm{ot}}, \mathsf{Ver}_{\mathrm{ot}})$ we will denote the signature scheme discussed in §5.2 which satisfies the required security notion. By $\mathsf{CEP}$ (commit-encrypt-prove) we denote the following:

$$\mathsf{CEP}(ck, pk, tag, msg;\ (\rho, r)) := \\ \big(\mathsf{Com}(ck, msg;\ \rho),\ \mathsf{Enc}(pk, tag, msg;\ r),\ \mathsf{NizkEq}(ck, pk,\ tag;\ msg, \rho, r)\big)$$

where $\mathsf{Enc}$ denotes Kiltz' encryption and $\mathsf{NizkEq}$ denotes a Groth-Sahai NIZK proof that the commitment and the encryption contain the same plaintext (cf. [Gro07]). We say that an output $\psi = (\mathbf{c}, C, \zeta)$ of $\mathsf{CEP}$ is *valid* if the ciphertext and the zero-knowledge proof are valid.

## 5 New Tools

### 5.1 A Scheme to Sign Three Diffie-Hellman Pairs

We extend the scheme from §4.1, so it signs three messages at once; we prove existential unforgeability (EUF) against adversaries making a particular chosen message attack (CMA): the first message is given (as usual) as a Diffie-Hellman pair, whereas the second and third message are queried as their logarithms; that is, instead of querying $(G^v, H^v)$, the adversary has to give $v$ explicitly. As we will see, this combines smoothly with our application.

**Scheme 2 ($\mathbf{Sig}_3$).**

$\mathsf{Setup}_3(\mathcal{G})$ Given $\mathcal{G} = (p, \mathbb{G}, \mathbb{G}_T, e, G)$, choose additional generators $H, K, T \in \mathbb{G}$.

$\mathsf{KeyGen}_3(\mathcal{G})$ Choose $sk = (x, \ell, u) \leftarrow \mathbb{Z}_p^3$ and set $vk = (G^x, G^\ell, G^u)$.

$\mathsf{Sign}_3((x, \ell, u), (M, N, Y, Z, V, W))$ A signature on $((M, N), (Y, Z), (V, W)) \in \mathcal{DH}^3$ under public key $G^x$, is defined as (for random $d, r \leftarrow \mathbb{Z}_p$)

$$\big(A := (KT^r M Y^\ell V^u)^{\frac{1}{x+d}}, C := G^d, D := H^d, R := G^r, S := H^r\big)$$

$\mathsf{Verify}_3$ $(A, C, D, R, S)$ is valid on messages $(M, N), (Y, Z), (V, W) \in \mathcal{DH}$ under a public key $(X, L, U)$ iff

$$e(A, XC) = e(KM, G)\, e(T, R)\, e(L, Y)\, e(U, V) \qquad \begin{matrix} e(C, H) = e(G, D) \\ e(R, H) = e(G, S) \end{matrix} \qquad (4)$$

**Theorem 1.** $\mathbf{Sig}_3$ *is existentially unforgeable against adversaries making chosen message attacks of the form* $((M_1, N_1), m_2, m_3)$.

*Proof.* Let $(M_i, N_i, y_i, v_i)$ be the queries, $(A_i, C_i, D_i, R_i = G^{r_i}, S_i)$ be the responses. Let $(M, N, Y, Z, V, W)$ and $(A, C, D, R = G^r, S)$ be a successful forgery. We distinguish 4 types of forgers (where $Y_i := G^{y_i}, V_i := G^{v_i}$):

Type I
$$\forall\, i : T^{r_i} M_i Y_i^{\ell} V_i^{u} \neq T^{r} M Y^{\ell} V^{u} \tag{5}$$

Type II
$$\exists\, i : T^{r_i} M_i Y_i^{\ell} V_i^{u} = T^{r} M Y^{\ell} V^{u} \;\wedge\; M_i Y_i^{\ell} V_i^{u} \neq M Y^{\ell} V^{u} \tag{6}$$

Type III
$$\exists\, i : M_i Y_i^{\ell} V_i^{u} = M Y^{\ell} V^{u} \;\wedge\; M_i V_i^{u} \neq M V^{u} \tag{7}$$

Type IV
$$\exists\, i : M_i Y_i^{\ell} V_i^{u} = M Y^{\ell} V^{u} \;\wedge\; M_i V_i^{u} = M V^{u} \tag{8}$$

**Type I** is reduced to DHSDH. Let $\left(G, H, K, (A_i, C_i, D_i, E_i, F_i)_{i=1}^{q-1}\right)$ be an instance. Choose and $t, \ell, u \leftarrow \mathbb{Z}_p$ and set $T = G^t$, $L = G^{\ell}$ and $U = G^u$. A signature on $(M_i, N_i, Y_i, Z_i, y_i, V_i, W_i, v_i)$ is (after a consistency check) answered as $(A_i, C_i, D_i, (E_i M_i^{-1} Y_i^{-\ell} V_i^{-u})^{1/t}, (F_i N_i^{-1} Z_i^{-\ell} W_i^{-u})^{1/t})$. After a successful forgery, return $(A, C, D, R^t M Y^{\ell} V^{u}, S^t N Z^{\ell} W^{u})$, which is a valid DHSDH solution by (5).

**Type II** is reduced to HDL. Let $(G, H, T)$ be an HDL instance. Generate the rest of the parameters and a public key and answer the queries by signing. After a successful forgery return the following, which is non-trivial by (6):

$$(M Y^{\ell} V^{u} M_i^{-1} Y_i^{-\ell} V_i^{-u}, N Z^{\ell} W^{u} N_i^{-1} Z_i^{-\ell} W_i^{-u}, R_i R^{-1}, S_i S^{-1}) \ .$$

**Type III** is reduced to HDL. Let $(G, H, L)$ be an instance. Choose $K, T \leftarrow \mathbb{G}$ and $x, u \leftarrow \mathbb{Z}_p$ and return the parameters and public key $(X = G^x, L, U = G^u)$. Thanks to the $y_i$ in the signing queries, we can simulate them: return $((K T^{r_i} M_i L^{y_i} V_i^{u})^{\frac{1}{x+d_i}}, G^{d_i}, H^{d_i}, G^{r_i}, H^{r_i})$. We have $M V^{u} M_i^{-1} V_i^{-u} = Y_i^{\ell} Y^{-\ell} = L^{y_i - y}$ from (7), so from a successful forgery, we can return

$$(M V^{u} M_i^{-1} V_i^{-u}, N W^{u} N_i^{-1} W_i^{-u}, Y_i Y^{-1}, Z_i Z^{-1}) \ ,$$

which is non-trivial by (7).

**Type IV** is also reduced to HDL. Let $(G, H, U)$ be an HDL instance. Choose $K, T \leftarrow \mathbb{G}$ and $x, \ell \leftarrow \mathbb{Z}_p$ and return the parameters and public key $(X = G^x, L = G^{\ell}, U)$. Thanks to the $v_i$ in the signing queries, we can simulate them: return $((K T^{r_i} M_i Y_i^{\ell} U^{v_i})^{\frac{1}{x+d_i}}, G^{d_i}, H^{d_i}, G^{r_i}, H^{r_i})$. From a successful forgery of Type IV we have $M M_i^{-1} = U^{v_i - v}$ from (7), we can thus return $(M M_i^{-1}, N N_i^{-1}, V_i V^{-1}, W_i W^{-1})$, which is non-trivial, $(M, N, Y, Z, V, W)$ being a valid forgery and $(Y, Z) = (Y_i, Z_i)$ by (8). ◻

### 5.2 A Simulation-Sound Non-interactive Zero-Knowledge Proof of Knowledge of a CDH Solution

Let $(G, F, V)$ be elements of $\mathbb{G}$. We construct a simulation-sound non-interactive zero-knowledge (SSNIZK) proof of knowledge (PoK) of $W$ s.t. $e(V, F) = e(G, W)$. We follow the overall approach by Groth [Gro06]. The common reference string (CRS) contains a CRS for Groth-Sahai (GS) proofs and a public key for a EUF-CMA signature scheme **Sig**. A proof is done as follows: choose a key pair for a one-time signature scheme $\mathbf{Sig}_{\text{ot}}$, and make a witness-indistinguishable GS proof of the following: either to know $W$, a CDH solution for $(G, F, V)$ or to know a signature on the chosen one-time key which is valid under the public key from the CRS;[2] finally sign the proof using the one-time key. A SSNIZKPoK is verified by checking the GS proofs and the one-time signature. Knowing the signing key corresponding to the key in the CRS, one can simulate proofs by using as a witness a signature on the one-time key.

We require that a proof consist of group elements only and is verified by checking a set of pairing-product equations. This can be achieved by using Scheme 1 and a one-time scheme to sign group elements using the commitment scheme in [Gro09] based on the DLIN assumption.[3]

## 6 A Fair Blind Signature Scheme

The basis of our protocol is the blind automorphic signature scheme from [Fuc09]: the user randomizes the message to be signed, the issuer produces a pre-signature from which the user obtains a signature by removing the randomness; the final signature is a Groth-Sahai (GS) proof of knowledge of the resulting signature.

In our scheme, in addition to the message, the issuer signs the user's public key, and an *identifier* of the signature, which the issuer and the user define jointly. Note that the issuer may neither learn the user's public key nor the identifier. To guarantee provable tracings, the user signs what she sends in the issuing protocol and the final signature. To prevent malicious issuers from producing a transcript that opens to an honest signature, the proof contains a SSNIZK proof of knowledge of the randomness introduced by the user. To achieve blindness against adversaries with tracing oracles, the elements that serve as proofs of correct tracing are additionally encrypted and the transcript (and final signature) is signed with a one-time key (cf. §4.3).

[2] [Gro06] shows how to express a disjunction of equation sets by a new set of equations.

[3] The strong one-time signature scheme from [Gro06] works as follows: the verification key is an (equivocable) Pedersen commitment to 0; to sign a message, the commitment is opened to the message using the trapdoor; putting a second trapdoor in the commitment scheme, we can simulate one signing query and use a forger to break the binding property of the commitment. In [Gro09], Groth proposes a scheme to commit to group elements which is computationally binding under DLIN. Using his scheme instead of Pedersen commitments, we can construct an efficient one-time signature on group elements s.t. signatures consist of group elements (see the full version [FV10] for the details).

To open a signature (i.e. to trace a user), the authority extracts tracing information from the commitments as well as signatures that act as proofs.

### 6.1 A Blind Signature Scheme

**Setup.** Choose a group $\mathcal{G} := (p, \mathbb{G}, \mathbb{G}_T, e, G)$ and parameters $(H, K, T)$ for $\mathbf{Sig}_3$. Pick $F, H' \leftarrow \mathbb{G}$, a commitment and extraction key $(ck, ek)$ for GS proofs, a key pair for tag-based encryption $(epk, esk)$ and $sscrs$, a common reference string for $\mathsf{SSNIZKPoK}$. Output $pp := (\mathcal{G}, G, H, K, T, F, H', ck, epk, sscrs)$ and $rk := ek$.

**Key Generation.** Both $\mathsf{IKGen}$ and $\mathsf{UKGen}$ are defined as $\mathsf{KeyGen}$, i.e. the key generation algorithm for $\mathbf{Sig}_1$.

**Signature Issuing.** The common inputs are $(pp, ipk = G^x)$, the issuer's additional input is $isk = x$, the user's inputs are $(upk = G^y, usk = y,\ (M, N) \in \mathcal{DH})$.

**User** Choose $\eta, v' \leftarrow \mathbb{Z}_p$ and set $P = G^\eta, Q = F^\eta, V' = G^{v'}, W' = F^{v'}$.
Produce $\xi \leftarrow \mathsf{SSNIZKPoK}(sscrs, (P, V'), (Q, W'))$. [4]
Choose $(vk'_{\mathrm{ot}}, sk'_{\mathrm{ot}}) \leftarrow \mathsf{KeyGen}_{\mathrm{ot}}(\mathcal{G})$ and set $\Sigma' \leftarrow \mathsf{Sign}(usk, vk'_{\mathrm{ot}})$. [5]
Send the following

1. $Y = G^y, Z = H^y$, $vk'_{\mathrm{ot}}$, $\Sigma'$;
2. $\mathbf{c}_M = \mathsf{Com}(ck, M)$; $\mathbf{c}_N := \mathsf{Com}(ck, N)$,
   $\psi_P, \psi_V, \vec{\psi}_\xi$, with $\psi_\odot := \mathsf{CEP}(ck, epk, vk'_{\mathrm{ot}}, \odot)$,
   a proof $\phi_M$ that $(M, N) \in \mathcal{DH}$ and a proof $\phi_\xi$ of validity of $\xi$;
3. $J := (KML^yU^{v'})^{\frac{1}{\eta}}$;
4. a zero-knowledge proof $\zeta$ of knowledge of $\eta, y$ and $v'$ such that
   - $Y = G^y$,
   - $\mathbf{c}_V$ commits to $G^{v'}$, and
   - $\mathbf{c}_M$ commits to $J^\eta L^{-y} U^{-v'} K^{-1}$;
5. $sig' \leftarrow \mathsf{Sign}_{\mathrm{ot}}(sk'_{\mathrm{ot}}, (Y, Z, \Sigma', \mathbf{c}_M, \mathbf{c}_N, \psi_P, \psi_V, \vec{\psi}_\xi, \phi_M, \phi_\xi, J, \zeta, vk'_{\mathrm{ot}}))$.

**Issuer** If $\Sigma', \psi_P, \psi_V, \vec{\psi}_\xi, \phi_M, \phi_\xi, sig'$ and the proof of knowledge are valid, choose $d, r, v'' \leftarrow \mathbb{Z}_p$ and send:

$$A' := (JT^rU^{v''})^{\frac{1}{x+d}} \quad C := G^d \quad D := F^d \quad R' := G^r \quad S' := H^r \quad v''$$

The user does the following:

1. set $A := (A')^\eta, R := (R')^\eta, S := (S')^\eta, V := G^{v'+\eta v''}, W := H^{v'+\eta v''}$ and check if $(A, C, D, R, S)$ is valid on $\big((M, N), (Y, Z), (V, W)\big)$ under $ipk$;
2. choose $(vk_{\mathrm{ot}}, sk_{\mathrm{ot}}) \leftarrow \mathsf{KeyGen}_{\mathrm{ot}}$ and define $\Sigma \leftarrow \mathsf{Sign}(y, vk_{\mathrm{ot}})$;

[4] A simulation-sound non-interactive proof of knowledge of $Q$ and $W'$ such that $e(V', F) = e(G, W')$ and $e(P, F) = e(G, Q)$. (cf. §5.2).

[5] The message space for **Sig** is the set of DH pairs w.r.t. $(G, H')$. Since all logarithms of $vk_{\mathrm{ot}}$ are known when picking a key, the user can complete the second components of the DH pairs.

3. make commitments $\mathbf{c}_A, \mathbf{c}_C, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S$ to $A, C, D, R, S$ under $ck$;
4. run $\mathsf{CEP}(ck, epk, vk_{\text{ot}}, \cdot)$ on $Y, Z, \Sigma$; let $\psi_Y, \psi_Z, \vec{\psi}_\Sigma$ denote the outputs;
5. make a proof $\phi_Y$ that $(Y, Z) \in \mathcal{DH}$ and proofs $\phi_S$ and $\phi_\Sigma$ of validity of the signatures $(A, C, D, R, S)$ and $\Sigma$;
6. set $sig \leftarrow \mathsf{Sign}_{\text{ot}}\big(sk_{\text{ot}}, (V, W, M, N, \mathbf{c}_A, \mathbf{c}_C, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \psi_Y, \psi_Z, \vec{\psi}_\Sigma, \phi_Y, \phi_S, \phi_\Sigma, vk_{\text{ot}})\big)$.

The signature on $(M, N)$ is

$$(V, W, \mathbf{c}_A, \mathbf{c}_C, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \psi_Y, \psi_Z, \vec{\psi}_\Sigma, \phi_Y, \phi_S, \phi_\Sigma, vk_{\text{ot}}, sig) \ .$$

**Verification.** A signature is verified by verifying $sig$ under $vk_{\text{ot}}$, checking the proofs $\phi_Y, \phi_S$ and $\phi_\Sigma$, and verifying the encryptions and NIZK proofs in $\psi_Y$, $\psi_Z$ and $\vec{\psi}_\Sigma$.

*Remark 1.* As mentioned by [Fuc09], there are two possible instantiations of the zero-knowledge proof of knowledge in Step 4 of $\mathsf{User}$: either using bit-by-bit techniques (which makes the protocol round-optimal); or optimizing the amount of data sent by adding 3 rounds using interactive concurrent Schnorr proofs.

**Theorem 2.** *The above scheme is an unforgeable blind signature (in the classical sense) under the DLIN, the DHSDH and the HDL assumption.*

The proof of unforgeability is by reduction to unforgeability of Scheme 2, analogously to the proof in [Fuc09]. Note that by additionally extracting $y$ and $v'$ from the proof of knowledge, the simulator can make the special signing queries. The proof of blindness is analogous, too.

**Opening of a Transcript ("Signature Tracing").** Given a transcript

$$(Y, Z, \Sigma', \mathbf{c}_M, \mathbf{c}_N, \psi_P, \psi_V, \vec{\psi}_\xi, \phi_M, \phi_\xi, J, \zeta, vk'_{\text{ot}}, sig') \ , \quad v''$$

verify $\Sigma'$, $sig'$, the proofs $\phi_M$ and $\phi_\xi$ and the ciphertexts and proofs in $\psi_P, \psi_V$ and $\vec{\psi}_\xi$. If everything is valid, use $rk = ek$ to open the commitments in $\psi_P, \psi_V$ and $\vec{\psi}_\xi$ to $P, V'$ and $\xi$ respectively and set $V := V' P^{v''} = G^{v' + \eta v''}$.

Return $id_\sigma := V$, $upk = Y$ and $\pi := (V', P, v'', \xi, \Sigma')$. The proof $\pi$ is verified by checking $V = V' P^{v''}$, verifying $\xi$ on $V'$ and $P$, and verifying $\Sigma'$ under $Y$.

**Opening of a Signature ("Identity Tracing").** Given a *valid* signature

$$(V, W, \mathbf{c}_A, \mathbf{c}_C, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \psi_Y, \psi_Z, \vec{\psi}_\Sigma, \phi_Y, \phi_S, \phi_\Sigma, vk_{\text{ot}}, sig) \ ,$$

open the commitments in $\psi_Y, \psi_Z$ and $\vec{\psi}_\Sigma$ using $ek$ and return $upk = Y$ and $\pi = \Sigma$. A proof $\pi$ is verified by checking if $\Sigma$ is valid on $(V, W)$ under $Y$.

## 7 Security Proofs

**Theorem 3.** *The above scheme is a secure fair blind signature scheme (in the model defined in §2) under the DLIN, the DHSDH and the HDL assumptions.*

Due to space limitation, we sketch the security proofs of all security notions.

**Blindness (under DLIN).** In the WI setting of GS proofs, commitments and proofs do not reveal anything—and neither do the ciphertexts. Furthermore, for every $M$ and $V$, there exist $\eta$ and $v'$ that explain $J$. In more detail: we proceed by games, Game 0 being the original game. In Game 1, we use the decryption key for the tag-based encryptions to answer tracing queries. Soundness of the NIZK proofs in the $\psi$'s guarantee that the committed and the encrypted values are the same; the games are thus indistinguishable.

In Game 2, we replace the commitment key $ck$ by a WI key (indistinguishable under DLIN). In Game 3, we simulate the NIZK proofs in the $\psi$'s and in Game 4, we replace the ciphertexts in the $\psi$'s by encryptions of 0. Games 3 and 4 are indistinguishable by selective-tag weak CCA security of Kiltz' cryptosystem (which follows from DLIN): by unforgeability of the one-time signature, the adversary cannot query a different transcript (or signature) with the same tag as the target transcript (signature), we can thus answer all tracing queries.

In Game 5, we simulate the zero-knowledge proofs in Step 4. In this game, the adversary's view is the following: $J = (KML^yU^{v'})^{\frac{1}{\eta}}$ and $M^*, V^*$ which are either $M$ and $G^{v'+\eta v''}$ or not. Let small letters denote the logarithms of the respective capital letters. Then for every $m^* = \log M^*, v^* = \log V^*$ there exist $\eta, v'$ such that $v^* = v' + \eta v''$ and $j = \frac{1}{\eta}(k + m^* + yl + v'u)$, i.e. that make $M^*, V^*$ consistent with $J$. In Game 5, which is indistinguishable from the original game, the adversary has thus no information on whether a given transcript corresponds to a given signature.

**Identity Traceability (under DHSDH+HDL).** An adversary wins if he can produce a set of valid pairs $(m_i, \sigma_i)$ s.t. either (I) for one of them the tracing returns $\perp$ or the proof does not verify, or (II) a user appears more often in the openings of the signatures than in the openings of the transcripts. By soundness of Groth-Sahai, we can always extract a user public key and a valid signature. If an adversary wins by (II), then we can use him to forge a $\mathbf{Sig}_3$ signature:

Given parameters and a public key for $\mathbf{Sig}_3$, we set up the rest of the parameters for the blind signature. Whenever the adversary queries his $\mathsf{Sign}$ oracle, we do the following: use $ek$ to extract $(M, N)$ from $(\mathbf{c}_M, \mathbf{c}_N)$, extract $\eta, y$ and $v'$ from the zero-knowledge proof of knowledge $\zeta$. Choose $v'' \leftarrow \mathbb{Z}_p$ and query $(M, N, y, v' + \eta v'')$ to signing oracle, receive $(A, C, D, R, S)$ and return $(A^{\frac{1}{\eta}}, C, D, R^{\frac{1}{\eta}}, S^{\frac{1}{\eta}}, v'')$. If the adversary wins by outputting a set of different (i.e. with distinct identifiers $(V, W)$) blind signatures with one user appearing more often than in the transcripts then among the $\mathbf{Sig}_3$ signatures extracted from the blind signatures there must be a forgery.

**Identity Non-Frameability (under DLIN+DHSDH+HDL).** Using a successful adversary, we can either forge a signature by the user on $vk'_{\mathrm{ot}}$ or a one-time signature (which is secure under DLIN). More precisely, we call an adversary of Type I if it reuses a one-time key from the signatures it received from the $\mathsf{User}$ oracle. Since the signature that $\mathcal{A}$ returns must not be contained in $Set$, it is different from the one containing the reused one-time key. The contained one-time signature can thus be returned as a forgery.

An adversary of Type II uses a new one-time key for the returned signature. We use $\mathcal{A}$ to forge a **Sig** signature. The simulator is given parameters $(H', K, T)$ and a public key $Y$ for **Sig**, sets it as one of the honest users' *upk* and queries its signing oracle to simulate the user. Having set $H = G^h$, the simulator can produce $Z = H^y = Y^h$ in the $\mathsf{User}$ oracle queries. Since the $vk'_{\text{ot}}$ contained $\mathcal{A}$'s output was never queried, we get a valid forgery.

**Signature Traceability (under DHSDH+HDL).** If the adversary wins by outputting a message/signature pair with an identifier $(V, W)$ s.t. no transcript opens to it, we can extract a $\mathbf{Sig}_3$ signature on $(M, N, Y, Z, V, W)$ without having ever queried a signature on any $(\cdot, \cdot, \cdot, \cdot, V, W)$. The simulation is done analogously to the proof of identity traceability. If the adversary outputs two different signatures they must have different identifiers; one of the $\mathsf{ChkSig}$ calls in the experiment returns thus 0. Note that with overwhelming probability two identifiers from different issuing sessions are different (since $v''$ is chosen randomly by the experiment *after* the adversary chose $v'$ and $\eta$).

**Signature Non-Frameability (under DLIN+DHSDH+HDL).** There are two ways for an issuer to "wrongfully" open a transcript: either he opens it to a user (not necessarily honest) and an identifier of a signature which was produced by an honest user in another session; or it opens to an honest user who has not participated in the issuing session.

Framing an honest signature. Suppose the adversary impersonating the issuer manages to produce a new opening of a transcript that leads to an honestly generated signature. We reduce this framing attack to break CDH, whose hardness is implied by DLIN. Let $(G, F, V')$ be a CDH challenge, i.e. we seek to produce $W' := F^{(\log_G V')}$. Set up the parameters of the scheme setting $H = G^h$ and knowing the trapdoor for $\mathsf{SSNIZKPoK}$. In one of the adversary's $\mathsf{User}$ oracle calls, choose $\eta \leftarrow \mathbb{Z}_p$ and use $V'$ from the CDH challenge. Simulate the proof of knowledge of $W'$. Let $v''$ be the value returned by the adversary, and let $(V := V'P^\eta, W := V^h)$ be the identifier of the resulting signature.

Suppose the adversary produces a proof $(\bar{V}', \bar{P}, \bar{v}'', \bar{\pi}, \bar{\Sigma})$ with $(\bar{V}', \bar{P}) \neq (V', P)$ for the honest identifier $(V, W)$. By simulation soundness of $\mathsf{SSNIZKPoK}$, we can extract $\bar{W}' = F^{(\log_G \bar{V}')}$ and $\bar{Q} = F^{(\log_G \bar{P})}$. From $V'G^{\eta v''} = V = \bar{V}'\bar{P}^{\bar{v}''}$ we get $V' = \bar{V}'\bar{P}^{\bar{v}''}G^{-\eta v''}$; thus $W' = \bar{W}'\bar{Q}^{\bar{v}''}F^{-\eta v''}$ is a CDH solution. If the adversary recycles $(V', P)$ then it must find a new $v''$ which leads to a $V$ of an honest signature, and thus has to solve a discrete logarithm.

Framing an honest user. Suppose the adversary outputs an opening of a transcript and a proof revealing an honest user that has never participated in that transcript. Analogously to the proof for signature traceability, we can use the adversary to either forge a signature under a user public key or to forge a one-time signature.

## 8 Conclusion

We presented the first practical fair blind signature scheme with a security proof in the standard model. The scheme satisfies a new security model strengthening the one proposed by Hufschmitt and Traoré in 2007. The new scheme is efficient (both keys and signatures consist of a constant number of group elements) and does not rely on any new assumptions. As byproducts, we proposed an extension of Fuchsbauer's automorphic signatures, a one-time signature on group elements, and a simulation-sound non-interactive zero-knowledge proof of knowledge of a Diffie-Hellman solution, all three compatible with the Groth-Sahai methodology.

## Acknowledgments

This work was supported by the French ANR 07-TCOM-013-04 PACE Project, the European Commission through the IST Program under Contract ICT-2007-216646 ECRYPT II, and EADS.

## References

[AF96] Abe, M., Fujisaki, E.: How to date blind signatures. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 244–251. Springer, Heidelberg (1996)

[AO01] Abe, M., Ohkubo, M.: Provably secure fair blind signatures with tight revocation. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 583–602. Springer, Heidelberg (2001)

[BB04] Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

[BBS04] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

[BFI+10] Blazy, O., Fuchsbauer, G., Izabachène, M., Jambert, A., Sibert, H., Vergnaud, D.: Batch Groth-Sahai. Cryptology ePrint Archive, Report 2010/040 (2010), http://eprint.iacr.org/

[BSZ05] Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)

[CGT06] Canard, S., Gaud, M., Traoré, J.: Defeating malicious servers in a blind signatures based voting system. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 148–153. Springer, Heidelberg (2006)

[Cha83] Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO 1982, pp. 199–203. Plenum Press, New York (1983)

[Dam92] Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)

[FPV09] Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Transferable constant-size fair e-cash. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 226–247. Springer, Heidelberg (2009)

[Fuc09] Fuchsbauer, G.: Automorphic signatures in bilinear groups. Cryptology ePrint Archive, Report 2009/320 (2009), http://eprint.iacr.org/

[FV10] Fuchsbauer, G., Vergnaud, D.: Fair blind signatures without random oracles. Cryptology ePrint Archive (2010), Report 2010/101, http://eprint.iacr.org/

[GL07] Groth, J., Lu, S.: A non-interactive shuffle with pairing based verifiability. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 51–67. Springer, Heidelberg (2007)

[Gro06] Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)

[Gro07] Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)

[Gro09] Groth, J.: Homomorphic trapdoor commitments to group elements. Cryptology ePrint Archive, Report 2009/007 (2009), http://eprint.iacr.org/

[GS08] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)

[GT03] Gaud, M., Traoré, J.: On the anonymity of fair offline e-cash systems. In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 34–50. Springer, Heidelberg (2003)

[HT07] Hufschmitt, E., Traoré, J.: Fair blind signatures revisited. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 268–292. Springer, Heidelberg (2007)

[Kil06] Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)

[Nao03] Naor, M.: On cryptographic assumptions and challenges (invited talk). In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)

[Oka06] Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)

[OMA+99] Ohkubo, M., Miura, F., Abe, M., Fujioka, A., Okamoto, T.: An improvement on a practical secret voting scheme. In: Zheng, Y., Mambo, M. (eds.) ISW 1999. LNCS, vol. 1729, pp. 225–234. Springer, Heidelberg (1999)

[RS10] Rückert, M., Schröder, D.: Fair partially blind signatures. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 34–51. Springer, Heidelberg (2010)

[SPC95] Stadler, M., Piveteau, J.-M., Camenisch, J.: Fair blind signatures. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 209–219. Springer, Heidelberg (1995)

# Fair Partially Blind Signatures

Markus Rückert* and Dominique Schröder**

TU Darmstadt, Germany
markus.rueckert@cased.de, schroeder@me.com

**Abstract.** It is well-known that blind signature schemes provide full anonymity for the receiving user. For many real-world applications, however, this leaves too much room for fraud. There are two generalizations of blind signature schemes that compensate this weakness: fair blind signatures and partially blind signatures. Fair blind signature schemes allow a trusted third party to revoke blindness in case of a dispute. In partially blind signature schemes, the signer retains a certain control over the signed message because signer and user have to agree on a specific part of the signed message.

In this work, we unify the previous well-studied models into a generalization, called fair partially blind signatures. We propose an instantiation that is secure in the standard model without any setup assumptions. With this construction, we also give a positive answer to the open question of whether fair blind signature schemes in the standard model exist.

**Keywords:** Blind signatures, generic construction, security model.

## 1 Introduction

Blind signatures, proposed by Chaum in 1982 [7], are interactive signature schemes between a signer and a user with the property that the message is hidden from the signer (blindness). Simultaneously, the user cannot produce more signatures than interactions with the signer took place (unforgeability). As one of the first applications of blind signatures, Chaum proposed untraceable digital payment (e-cash). In this context, Chaum pointed out that such a high degree of privacy enables an adversary to doubly spend an electronic coin if no countermeasures are taken. Further fraud scenarios are given in [25]. Another potential loophole in ordinary blind signature schemes, first mentioned by Abe and Okamoto [2], is that the signer entirely loses control over the signed message. Consider, e.g., vouchers with a predetermined expiration date. The signer is willing to blindly sign the voucher but wants to ensure that the receiving user cannot control the expiry date. For both weaknesses, two different countermeasures have been proposed, namely, fair blind and partially blind signatures.

* This work was supported by CASED (www.cased.de).
** Dominique Schröder was supported by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG).

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 34–51, 2010.

Fair blind signatures, suggested by Stadler, Piveteau, and Camenisch [24], involve a trusted third party, which is able to revoke blindness in case of a dispute between signer and user. The revocation mechanism works in two directions: it either uniquely identifies an obtained signature from the signer's view of a specific session (signature tracing), or connects a given signature with a specific session of the protocol (session tracing). The trusted party is *offline*, i.e., there is no initial setup phase and the signature issue protocol does not involve this party. It is only appealed to in case of irregularities, such as fraud or other crimes.

A second approach by Abe and Fujisaki [1], named partially blind signatures, compensates for the potential loophole that the signer entirely loses control over the signed message. Reconsider the example of vouchers with a predetermined expiration date. Signer and user both agree on some piece of information, such as the expiry date. Here, verification only works if user and signer agree on the same date. As soon as the user tries to change this auxiliary information, verification will fail.

The goal of our work is to establish a unified model of fair partially blind signatures that encompasses all previous concepts and their security models and to find provably secure instantiations. We motivate the need for the generalization of both concepts with the following simple example that can be transferred to other, more complex, application scenarios: consider the scenario where a bank issues electronic coins of different value. With ordinary blind signatures, the bank has to apply different keys for different values and cannot be sure that no malicious user doubly spends the coin. The use of partially blind signatures schemes allows the signer to work with a single key, while including the value of the coin as auxiliary information. Still, criminal investigations will be hindered by the customer's irrevocable anonymity. On the other hand, using fair blind signatures allows for revocable anonymity but then it is again necessary for the signer to use multiple keys.

Thus, with fair partially blind signature schemes, we are able to get the best of both worlds. A bank that employs fair partially blind signatures only needs a single key pair, while simultaneously being able to remove blindness if necessary. Note that the revocation will probably not be done by the bank itself but by a law enforcement agency or a trusted notary. We believe that combining both concepts is most suitable for real-world applications, where the individual needs of customers (blindness), service providers (partial control), and those of the authorities (fairness) have to be satisfied.

Related Work. Since Stadler, Piveteau, and Camenisch described the idea of *fair blind signatures* in 1995 [24], many constructions have been proposed, e.g., [14,19,21,20,3,17]. Unfortunately, some of these constructions cannot be considered "blind" in the sense of Juels et al. [18] and thus Abe and Ohkubo [3] developed a formal security model. Unfortunately, all previous results either provide only security arguments, or are provably secure in the random oracle model, which is discouraged by the work of Canetti, Goldreich, and Halevi [6]. We are

not aware of any instantiation in the standard model. *Partially blind signatures*, due to Abe and Fujisaki [1], are also well-studied and several instantiations have been proposed, e.g., [1,2,8,22]. Recently, the first instantiation without random oracles or setup assumptions has been proposed in [22].

In [17], Hufschmitt and Traoré consider dynamic fair blind signatures, a concept that is inspired by dynamic group signatures [5]. They require that each user of the blind signature scheme has to register before being able to obtain signatures. We also discuss the relation to our model.

Contribution. We propose a novel security model, which is a generalization of the well-studied models of Juels, Luby, and Ostrovsky [18] and Pointcheval and Stern [23] for blind signatures, the model of Abe and Ohkubo [3] for fair blind signatures, and that of Abe and Fujisaki [1] for partially blind signatures. With our model for fair partially blind signatures, we provide a unified framework that can be used to instantiate blind, fair blind, and partially blind signature schemes under a strong security model. We present a provably secure instantiation within this model. The construction, which is inspired by the works of Fischlin [11] and Hazay et al. [16], relies on general assumptions and is provably secure in the standard model without any setup assumptions. By eliminating the auxiliary information, our scheme solves the longstanding problem of instantiability of fair blind signature schemes in the standard model [24]. Removing the trusted third party also yields the first partially blind signature scheme based on general assumption which is provably secure in the standard model, again without any setup assumptions. Independently of our work, Fuchsbauer and Vergnaud construct the first efficient instantiation of fair blind signatures in the standard model [15].

Organization. After recalling basic definitions and notations, Section 2 introduces the concept of fair partially blind signatures along with a security model and a discussion of the various security properties. Section 3 is a warm-up for Section 4 to make it more accessible. Then, we provide a provably secure instantiation from general assumption in Section 4.

## 2 Fair Partially Blind Signatures

Notation. We use the following notation for interactive executions between algorithms $\mathcal{X}$ and $\mathcal{Y}$. The joint execution between $\mathcal{X}$ and $\mathcal{Y}$ is denoted by $(a, b) \leftarrow \langle \mathcal{X}(x), \mathcal{Y}(y) \rangle$, where $x$ is the private input of $\mathcal{X}$ and $y$ is the private input of $\mathcal{Y}$. The private output of algorithm $\mathcal{X}$ is $a$ and $b$ is the private output of $\mathcal{Y}$. If an algorithm $\mathcal{Y}$ can invoke an unbounded number of executions of the interactive protocol with $\mathcal{X}$, then we write $\mathcal{Y}^{\langle \mathcal{X}(x), \cdot \rangle^{\infty}}(y)$. Accordingly, $\mathcal{X}^{\langle \cdot, \mathcal{Y}(y_0) \rangle^1, \langle \cdot, \mathcal{Y}(y_1) \rangle^1}(x)$ means that $\mathcal{X}$ can execute arbitrarily ordered executions with $\mathcal{Y}(y_0)$ and $\mathcal{Y}(y_1)$, but only once. An algorithm is efficient, if it runs in probabilistic polynomial-time.

### 2.1 Definition

The concept of fair partially blind signature schemes generalizes the idea of partially blind signatures and fair blind signatures. A partially blind signature scheme is a blind signature scheme such that signer and user agree on some piece of information, denoted with $\mathsf{info}$. The signature should only be valid if *both* parties use this specific $\mathsf{info}$ element during the signature issue protocol.

Fair blind signatures have a revocation mechanism, enforceable by a trusted third party, which can revoke blindness upon disputes between signer and user. Revocation works in both directions. Whenever we need to find the signature that corresponds to a certain session, we query the revocation authority with a view of a session and obtain a signature identifier $\mathsf{id}_{\mathsf{Sig}}$. Now, when given a signature, we can verify whether it corresponds to $\mathsf{id}_{\mathsf{Sig}}$ or not. If, instead, we would like to find the session that resulted in a certain signature, we query the revocation authority with this signature to obtain a session identifier $\mathsf{id}_{\mathsf{Ses}}$. Again, we can easily verify whether a given session corresponds to $\mathsf{id}_{\mathsf{Ses}}$ or not.

In both cases, the trusted party outputs an identifier, which *uniquely* associates an execution with a signature. In the following definition, we combine both types of blind signatures.

**Definition 1 (Fair Partially Blind Signature Scheme).** *A fair partially blind signature scheme* FPBS *consists of the following efficient algorithms:*

**Key Generation.** $\mathsf{Kg}(1^n)$ *generates a key pair* $(sk, pk)$.
**Revocation Key Generation.** $\mathsf{RevKg}(1^n)$ *outputs a key pair* $(rsk, rpk)$.
**Signature Issuing.** *The joint execution of the algorithms* $\mathcal{S}(sk, rpk, \mathsf{info})$ *and* $\mathcal{U}(pk, rpk, m, \mathsf{info})$ *with message* $m \in \{0,1\}^n$ *and information element* $\mathsf{info} \in \{0,1\}^n$*, generates the private output* $\sigma$ *of the user and the private view* $\mathsf{view}$ *of the signer,* $(\mathsf{view}, \sigma) \leftarrow \langle \mathcal{S}(sk, rpk, \mathsf{info}), \mathcal{U}(pk, rpk, m, \mathsf{info}) \rangle$.
**Verification.** $\mathsf{Vf}(pk, rpk, m, \mathsf{info}, \sigma)$ *outputs a bit, indicating the validity of* $\sigma$.
**Signature Revocation.** $\mathsf{SigRev}$ *takes as input the signer's view* $\mathsf{view}$ *of a session and the secret revocation key rsk. It outputs an identifier* $\mathsf{id}_{\mathsf{Sig}}$*, which corresponds to the signature that the user obtained in this same session.*
**Session Revocation.** *When queried with a message* $m$*, with an information element* $\mathsf{info}$*, with a (valid) signature* $\sigma$ *and with the secret revocation key rsk,* $\mathsf{SesRev}$ *discloses the session identifier* $\mathsf{id}_{\mathsf{Ses}}$.
**Signature Tracing.** *On input* $(\mathsf{id}_{\mathsf{Sig}}, \sigma)$, $\mathsf{SigVf}$ *outputs a bit indicating whether* $\mathsf{id}_{\mathsf{Sig}}$ *matches* $\sigma$.
**Session Tracing.** *On input* $(\mathsf{id}_{\mathsf{Ses}}, \mathsf{view})$, $\mathsf{SesVf}$ *returns 1 if* $\mathsf{id}_{\mathsf{Ses}}$ *matches to* $\mathsf{view}$ *and 0 otherwise.*

*It is assumed that a fair partially blind signature scheme is* complete*:*

- *For any* $n \in \mathbb{N}$*, any* $(sk, pk) \leftarrow \mathsf{Kg}(1^n)$*, any* $(rsk, rpk) \leftarrow \mathsf{RevKg}(1^n)$*, any* $\mathsf{info} \in \{0,1\}^n$*, any message* $m \in \{0,1\}^n$ *and any* $\sigma$ *output by* $\mathcal{U}(pk, rpk, m, \mathsf{info})$ *after the joint execution with* $\mathcal{S}(sk, rpk, \mathsf{info})$*, we have* $\mathsf{Vf}(pk, rpk, m, \mathsf{info}, \sigma) = 1$.

– *For any* $n \in \mathbb{N}$, *any* $(sk, pk) \leftarrow \mathsf{Kg}(1^n)$, *any* $(rsk, rpk) \leftarrow \mathsf{RevKg}(1^n)$, *any message* $m \in \{0,1\}^n$, *any* $\mathsf{info} \in \{0,1\}^n$, *any* $\sigma$ *output by* $\mathcal{U}(pk, rpk, m, \mathsf{info})$ *in the joint execution with* $\mathcal{S}(sk, rpk, \mathsf{info})$, *and any* $\mathsf{view}$, *as observed by the signer, we have* $\mathsf{SigVf}(\mathsf{SigRev}(rsk, \mathsf{view}), \sigma) = 1$ *and* $\mathsf{SesVf}(\mathsf{SesRev}(rsk, m, \mathsf{info}, \sigma), \mathsf{view}) = 1$.

Note that the algorithms SigRev and SesRev may get additional public parameters as input if needed. As for info, we point out that signer and user agree on it before engaging in the signature issue protocol. This process is application specific and will not be addressed here. Furthermore, note that restricting $m$ and info to $\{0,1\}^n$ is no limitation in practice because one can always extend it to $\{0,1\}^*$ via a collision resistant hash function.

### 2.2 Security of Fair Partially Blind Signatures

Our security model for fair partially blind signature schemes is related to the security model for blind signatures [18,23,13], to the model for partially blind signatures [2], and to that for fair blind signatures [3]. In fact, it unifies the various types of blind signature schemes into a single model.

Security of blind signature schemes requires two properties, namely, unforgeability and blindness [18,23]. Unforgeability demands that a malicious user should not be able to produce more signatures than there were successful interactions with the signer. Here, the user is allowed to choose the messages as well as the information elements adaptively.

In the context of partially blind signatures, we want unforgeability to be stronger than in the classical case since "recombination" attacks should be considered. That is, an adversarial user should not be able to generate a valid signature for a *new* info instead of just for a new message. Depending on the application, one might even want to consider the stronger notion of "strong unforgeability". There, the adversary also wins if it outputs a *new* signature.

**Definition 2 (Unforgeability).** *A fair partially blind signature scheme* FPBS *is called* unforgeable *if for any efficient algorithm* $\mathcal{U}^*$ *the probability that experiment* $\mathsf{Forge}^{\mathsf{FPBS}}_{\mathcal{U}^*}(n)$ *evaluates to* 1 *is negligible (as a function of* $n$*) where*

***Experiment*** $\mathsf{Forge}^{\mathsf{FPBS}}_{\mathcal{U}^*}(n)$
$(rsk, rpk) \leftarrow \mathsf{RevKg}(1^n)$
$(sk, pk) \leftarrow \mathsf{Kg}(1^n)$
*For each* $\mathsf{info}$, *let* $k_{\mathsf{info}}$ *denote the number of successful, complete interactions*
$((m_1, \mathsf{info}, \sigma_1), \ldots, (m_{k_{\mathsf{info}}+1}, \mathsf{info}, \sigma_{k_{\mathsf{info}}+1})) \leftarrow \mathcal{U}^{*\langle \mathcal{S}(sk, rpk, \cdot), \cdot \rangle^\infty}(pk, rsk, rpk)$
*Return* 1 *iff*
$m_i \neq m_j$ *for* $1 \leq i < j \leq k_{\mathsf{info}} + 1$, *and*
$\mathsf{Vf}(pk, rpk, m_i, \mathsf{info}, \sigma_i) = 1$ *for all* $i = 1, 2, \ldots, k_{\mathsf{info}} + 1$.

Partial blindness is a generalization of blindness. It allows the malicious signer $\mathcal{S}^*$ to choose a public key, two messages $m_0, m_1$, and info on its own. The signer then interacts with two honest user instances. Based on a coin flip $b$, the first user obtains a signature for $m_b$ and the second obtains one for $m_{1-b}$. If both

protocol executions were successful, $\mathcal{S}^*$ gets the signatures $\sigma_0, \sigma_1$ for $m_0, m_1$, respectively, in the original order. The task is to guess $b$.

**Definition 3 (Partial Blindness).** FPBS *is* partially blind *if for any efficient algorithm* $\mathcal{S}^*$ *(working in modes* find, issue, *and* guess*) the probability that the following experiment* $\mathsf{FPBlind}^{\mathsf{FPBS}}_{\mathcal{S}^*}(n)$ *outputs* 1 *is negligibly close to* 1/2*, where*

***Experiment*** $\mathsf{FPBlind}^{\mathsf{FPBS}}_{\mathcal{S}^*}(n)$
$(rsk, rpk) \leftarrow \mathsf{RevKg}(1^n)$
$(pk, m_0, m_1, \mathit{info}, \mathit{st}_{\mathit{find}}) \leftarrow \mathcal{S}^*(\mathit{find}, rpk, 1^n)$
$b \leftarrow \{0,1\}$
$\mathit{st}_{\mathit{issue}} \leftarrow \mathcal{S}^{*\langle\cdot,\mathcal{U}(\ldots,m_b)\rangle^1,\langle\cdot,\mathcal{U}(\ldots,m_{1-b})\rangle^1}(\mathit{issue}, \mathit{st}_{\mathit{find}}, rpk)$
*Let* $\sigma_b$ *and* $\sigma_{1-b}$ *be the private outputs of* $\mathcal{U}(pk, rpk, m_b)$ *and* $\mathcal{U}(pk, rpk, m_{1-b})$*. and let* $\mathsf{view}_0$ *and* $\mathsf{view}_1$ *be the corresponding views of* $\mathcal{S}^*$*.*
*Set* $(\sigma_0, \sigma_1) = (\bot, \bot)$ *if* $\sigma_0 = \bot$ *or* $\sigma_1 = \bot$
$b^* \leftarrow \mathcal{S}^*(\mathit{guess}, \sigma_0, \sigma_1, \mathit{st}_{\mathit{issue}})$
*Return* 1 *iff* $b = b^*$*.*

As already mentioned in [3], it is possible to strengthen the notion of partial blindness even further by giving the adversarial signer access to conditional signature and session revocation oracles. There, the signer is allowed to query the oracles on any but the "relevant" signatures $\sigma_0, \sigma_1$ or sessions $\mathsf{view}_0, \mathsf{view}_1$. We call this stronger notion *strong partial blindness*.

Fairness of (partially) blind signatures consists of signature and session traceability. Signature traceability means that, given the revocation key *rsk* every message-signature pair can be related to exactly one execution of the signature issue protocol. This intuition is formalized in the following experiment, where a malicious user tries to output a valid message-signature tuple such that no matching protocol instance can be found. The second possibility to win the experiment is to output two message-signature pairs corresponding to the same protocol instance.

In the context of partially blind signatures, we give a strengthened definition in the following sense. The malicious user even succeeds with a valid message-signature tuple $(m, \mathsf{info}, \sigma)$ such that no session corresponding to the *same* info exists. Thus, the adversary merely needs to find a message-signature tuple, whose view matches that of $\mathsf{info}' \neq \mathsf{info}$. Observe that the adversary has access to *rsk*.

For each info, let $k_{\mathsf{info}}$ be the number of successful, complete interactions and let $V_{\mathsf{info}}$ be the set of corresponding protocol views. Let $V_* = \bigcup V_{\mathsf{info}}$ contain all views and assume that the adversary $\mathcal{U}^*$ asks its signature oracle at most $|V_*| \leq q$ times.

**Definition 4 (Signature Traceability).** FPBS *is* signature traceable *if for any efficient algorithm* $\mathcal{U}^*$ *the probability that experiment* $\mathsf{SigTrace}^{\mathsf{FPBS}}_{\mathcal{U}^*}(n)$ *evaluates to* 1 *is negligible (as a function of* $n$*), where*

***Experiment*** $\mathsf{SigTrace}^{\mathsf{FPBS}}_{\mathcal{U}^*}(n)$
$(rsk, rpk) \leftarrow \mathsf{RevKg}(1^n)$
$(sk, pk) \leftarrow \mathsf{Kg}(1^n)$
$((m_1, \mathsf{info}_1, \sigma_1), (m_2, \mathsf{info}_2, \sigma_2)) \leftarrow \mathcal{U}^{*\langle \mathcal{S}(sk,rpk,\cdot),\cdot\rangle^\infty}(pk, rsk, rpk)$
*Return* 1 *if*
*for one of the message-signature tuples, denoted by* $(m, \mathsf{info}, \sigma)$*, we have:*
$\mathsf{Vf}(pk, rpk, m, \mathsf{info}, \sigma) = 1$ *and* $\mathsf{SigVf}(\mathsf{id}_{\mathsf{Sig}}, \sigma) = 0$
*for all* $\mathsf{id}_{\mathsf{Sig}} \leftarrow \mathsf{SigRev}(\mathsf{view}, rsk)$ *with* $\mathsf{view} \in V_{\mathsf{info}}$*,*
*or if* $(m_1, \mathsf{info}_1) \neq (m_2, \mathsf{info}_2)$ *and*
$\mathsf{Vf}(pk, rpk, m_i, \mathsf{info}_i, \sigma_i) = 1$*,* $i = 1, 2$*, and there exists a* $\mathsf{view} \in V_*$ *s.t.*
$\mathsf{SigVf}(\mathsf{id}_{\mathsf{Sig}}, \sigma_0) = \mathsf{SigVf}(\mathsf{id}_{\mathsf{Sig}}, \sigma_1) = 1$ *where* $\mathsf{id}_{\mathsf{Sig}} \leftarrow \mathsf{SigRev}(rsk, \mathsf{view})$.

Given, on the other hand, a view of a protocol execution, the trusted party can uniquely determine the message-signature tuple that was generated. This property is called session traceability. In the following experiment, the adversarial user tries to output a message-signature tuple such that either no session corresponds to this signature or such that at least two sessions can be associated to it. Again, we give a stronger definition by letting the attacker win if the output signature matches to a session for a different $\mathsf{info}$ and we let it have *rsk*.

**Definition 5 (Session Traceability).** $\mathsf{FPBS}$ *is* session traceable *if for any efficient algorithm* $\mathcal{U}^*$ *the probability that experiment* $\mathsf{SesTrace}^{\mathsf{FPBS}}_{\mathcal{U}^*}(n)$ *evaluates to* 1 *is negligible (as a function of* $n$*), where*

***Experiment*** $\mathsf{SesTrace}^{\mathsf{FPBS}}_{\mathcal{U}^*}(n)$
$(rsk, rpk) \leftarrow \mathsf{RevKg}(1^n)$
$(sk, pk) \leftarrow \mathsf{Kg}(1^n)$
$(m, \mathsf{info}, \sigma) \leftarrow \mathcal{U}^{*\langle \mathcal{S}(sk,rpk,\cdot),\cdot\rangle^\infty}(pk, rsk, rpk)$
*Let* $\mathsf{id}_{\mathsf{Ses}} \leftarrow \mathsf{SesRev}(rsk, m, \mathsf{info}, \sigma)$.
*Return* 1 *iff*
$\mathsf{Vf}(pk, rpk, m, \mathsf{info}, \sigma) = 1$ *and* $\mathsf{SesVf}(\mathsf{id}_{\mathsf{Ses}}, \mathsf{view}) = 0$ *for all* $\mathsf{view} \in V_{\mathsf{info}}$
*or there exist distinct* $\mathsf{view}_1, \mathsf{view}_2 \in V_*$ *such that*
$\mathsf{SesVf}(\mathsf{id}_{\mathsf{Ses}}, \mathsf{view}_1) = \mathsf{SesVf}(\mathsf{id}_{\mathsf{Ses}}, \mathsf{view}_2) = 1$.

As already observed in [17] and others, unforgeability is implied by traceability. Thereby, we obtain the following definition of "security".

**Definition 6.** $\mathsf{FPBS}$ *is* secure *if it is partially blind, signature traceable, and session traceable.*

Relation to the Security Notion of Hufschmitt and Traoré. As already mentioned in the introduction, the security model of [17] for fair blind signatures is inspired by dynamic group signatures [5]. In contrast to blind signatures, group signature consider a fixed group where each member is statically registered. Dynamic group signatures allow in addition members to dynamically join and leave the group. Although this may be common in many practical scenarios, it complicates the definitions. In our work, we want to keep the definitions as simple as possible (and we want to follow the widely accepted notion of blind signatures).

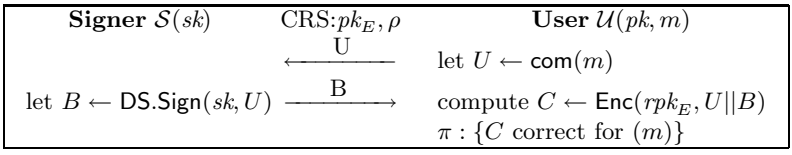



**Fig. 1.** Fischlin's protocol

With our approach, we also demonstrate that registration procedures are not at all necessary to construct provably secure and flexible fair and partially blind signatures. Therefore, we do not consider their alternative security model.

## 3 A Warm-Up — Fischlin's Blind Signature Scheme

Our blind signature scheme relies on an elegant construction due to Fischlin [11] that is provably secure in the *common reference string* (CRS) model. We review a simplified version of Fischlin's scheme ([11] presents a *strongly* unforgeable scheme) and then propose a partial solution to fair partially blind signatures. Fischlin's scheme performs the following steps:

**Setup.** The CRS contains a public key $pk_E$ for a semantically secure public-key encryption scheme, and a string $\rho$ used as a CRS for the non-interactive zero-knowledge proof.

**Key Generation.** The key generator runs the key generation algorithm of a standard signature scheme $(pk, sk) \leftarrow \mathsf{DS.Kg}(1^n)$ and returns both keys.

**Signing.** The interactive protocol in which the user $\mathcal{U}$ derives a signature on a message $m$ is as follows:

- $\mathcal{U}$ computes a commitment $U \leftarrow \mathsf{com}(m)$ and sends it to the signer.
- $\mathcal{S}$ signs the commitment $B \leftarrow \mathsf{DS.Sign}(sk, U)$ and sends $B$.
- $\mathcal{U}$ checks whether the signature is valid and aborts if it is invalid. Otherwise, the user computes $C \leftarrow \mathsf{Enc}(rpk_E, U||B)$ together with a NIZK proof $\pi$ that a valid signature is encrypted properly. Then it outputs the signature $(C, \pi)$.

**Verification.** To verify that $(C, \pi)$ is a valid signature on $m$ (w.r.t. $pk$), the verification algorithm checks that $\pi$ is a valid proof (w.r.t. $\rho$).

We briefly discuss the security of the scheme and then how to derive a fair partially blind signature scheme out of Fischlin's blind signature scheme.

Unforgeability of the scheme follows from the unforgeability of the regular signature scheme and from the binding property of the commitment. On the other hand, the hiding property of the commitment scheme prevents the malicious signer from learning any information about the message during the issue protocol. Furthermore, the semantically secure encryption scheme hides the encrypted message-signature pair and the non-interactive zero-knowledge proof discloses no information about the plaintext in $C$.

Fair Partially Blind Signatures: a Partial Solution. In order to adapt Fischlin's scheme and avoid the CRS, we have several difficulties to overcome. In the following, we describe these problems together with our solutions.

- Removing CRS $\rho$: We apply the technique of Hazay et al. [16] to remove the CRS. Thus, we rely on ZAPs [9] rather than on NIZKs (a ZAP is a two round witness-indistinguishable proof system, we give a formal definition in the following section).
- Achieving fairness: In order the obtain a "fair" scheme, we have to guarantee that each message-signature pair can, given a revocation key, be related to exactly one execution (and vice versa). To do so, we let the signer choose a random session identifier and (verifiably) include it in the signature as well as in the user's encryption of the signature.
- Achieving partial message control: We allow the signer to control a certain part $\mathsf{info}$ of the signed message by including it in its signature. Simultaneously, the user must include the same $\mathsf{info}$ in the encryption of the message. Thus, one-sided changes to $\mathsf{info}$ result in an invalid signature.

## 4 An Instantiation from General Assumptions

Before presenting our generic construction, we review the required primitives.

Indistinguishable Encryption Under Chosen-Plaintext Attacks (IND-CPA). A public-key encryption scheme $\mathsf{PKE}$ is secure in the IND-CPA model if no efficient adversary $\mathcal{A}$ can distinguish ciphertexts for messages of its choice. The corresponding experiment gives $\mathcal{A}$ access to an encryption oracle, which is initialized with a public key $rpk$ and a bit $b$. The encryption oracle takes as input two equally sized messages $(m_0, m_1)$ and returns $\mathsf{Enc}(rpk, m_b)$. The adversary wins if it is able to guess $b$ with probability noticeable greater than $1/2$. We assume in this construction that the message expansion $c$ is deterministic, i.e., the length of the ciphertext is $c(n)$ for all messages of length $n$.

Strongly Unforgeable Signature Scheme. A digital signature $\mathsf{DS}$ is strongly unforgeable under adaptive chosen message attacks if there is no efficient algorithm, which queries a signing oracle on messages of its choice, that manages to output a *new* signature for a (possibly) queried message. We assume all signature to be encoded as $l(n)$ bit strings.

Proof of Knowledge ($\mathsf{PoK}$). Let's consider an $\mathcal{NP}$-Language $L$ with relation $R_L := \{(s, w) \mid s \in L \text{ and } w \text{ is a witness for } s\}$. Informally, we call an interactive proof protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ a *proof of knowledge* if no malicious prover $\mathcal{P}^*$ can cheat but with negligible probability. If $\mathcal{P}^*$ convinces the verifier with noticeable probability then there exists an efficient algorithm $\mathsf{Extract}$, the extractor, which is able to extract a value $y$ from $\mathcal{P}^*$ such that $(x, y) \in R_L$. We refer the interested reader to [4] for further details.

Witness-Indistinguishability. In the case that an $\mathcal{NP}$-Language $L$ has at least two witnesses $w_1$ and $w_2$ for some string $s \in L$, we can define witness-indistinguishable proofs. Basically, a witness-indistinguishable interactive proof system [12] allows the prover to prove some statement, without revealing the witness ($w_1$ or $w_2$) used during the proof. This condition even holds if the verifier

knows both witnesses. Thus, a witness-indistinguishable proof of knowledge (WI-PoK) hides the used witness and simultaneously allows its extraction.

ZAP. Roughly speaking, a ZAP is a two round public coin witness-indistinguishable protocol [9] with the useful property that the first round (a message from verifier $\mathcal{V}$ to prover $\mathcal{P}$) can be made universal for all executions and therefore be part of the public key of $\mathcal{V}$. Dwork and Naor showed that ZAPs can be build upon any trapdoor permutation [9].

**Definition 7 (ZAP).** *Let $L_{p(n)} := L \cap \{0,1\}^{\leq p(n)}$ for some polynomial $p$. A ZAP is 2-round public coin witness-indistinguishable protocol for some $\mathcal{NP}$-language $L$ with associated relation $R_L$. It consists of two efficient interactive algorithms $\mathcal{P}, \mathcal{V}$ such that*

- *The verifier $\mathcal{V}$ outputs an initial message $\rho$ on input $1^n$;*
- *The prover $\mathcal{P}$ gets as input $\rho$, a statement $s \in L_{p(n)}$, and a witness $w$ such that $(s, w) \in R_L$. It outputs a proof $\pi$;*
- *The verifier $\mathcal{V}$ outputs a decision bit when queried with $\rho, s$ and $\pi$.*

*A ZAP is* complete *if for any $n \in \mathbb{N}$ and any $(s, w) \in R_L$, we have $\mathcal{V}(\rho, s, \mathcal{P}(\rho, s, w)) = 1$. Furthermore, ZAPs satisfy the following properties (see Figure 2 for the experiments):*

**Witness-Indistinguishability.** *A ZAP is* witness-indistinguishable *if for any efficient algorithm $\mathcal{V}^*$ (working in modes* find *and* guess*) the probability that experiment $\mathsf{WitInd}_{\mathcal{V}^*}^{ZAP}$ outputs 1 is negligibly close to $1/2$.*

**Adaptive Soundness.** *A ZAP satisfies* adaptive soundness *if for any algorithm $\mathcal{P}^*$ the probability that experiment $\mathsf{AdSnd}_{\mathcal{P}^*}^{ZAP}$ outputs 1 is negligible.*

**Experiment** $\mathsf{WitInd}_{\mathcal{V}^*}^{\mathrm{ZAP}}$
$b \leftarrow \{0,1\}$
$(\rho, s_1, \ldots, s_\ell, (w_1^0, \ldots, w_\ell^0),$
$\quad (w_1^1, \ldots, w_\ell^1), \mathsf{st}_{\mathsf{find}}) \leftarrow \mathcal{V}^*(\mathsf{find}, 1^n)$
Return 0 if $(s_i, w_i^0), (s_i, w_i^1) \notin R_L$ for $i \in [1, \ell]$.
$\pi_i \leftarrow \mathcal{P}(\rho, s_1, w_i^b)$ for $i \in [1, \ell]$.
$b' \leftarrow \mathcal{V}^*(\mathsf{guess}, \mathsf{st}_{\mathsf{find}}, \rho, (\pi_1, \ldots, \pi_\ell), (s_1, \ldots, s_\ell))$
Return 1 iff $b' = b$.

**Experiment** $\mathsf{AdSnd}_{\mathcal{P}^*}^{\mathrm{ZAP}}$
$\rho \leftarrow \mathcal{V}(1^n)$
$(s, \pi) \leftarrow \mathcal{P}^*(\rho, 1^n)$
Return 1 iff $\mathcal{V}(\rho, s, \pi) = 1$ and $s \notin L$.

**Fig. 2.** Experiments describing witness-indistinguishability and adaptive soundness

### 4.1 Construction

From a high-level point of view, the idea of our construction is as follows. In the first step, the user encrypts a message $m$ together with info under the public key *rpk* of the trusted third party and sends this encryption to the signer. The signer then concatenates the encryption with info and a unique session identifier ssid, signs the resulting string using an ordinary signature scheme and sends the signature back to the user. Finally, the user encrypts (again under the public key of the trusted party) the first message as well as the signature and proves that the valid signature is properly encrypted. Note that we use two (not necessarily distinct) encryption schemes in order to handle the different input/output lengths.

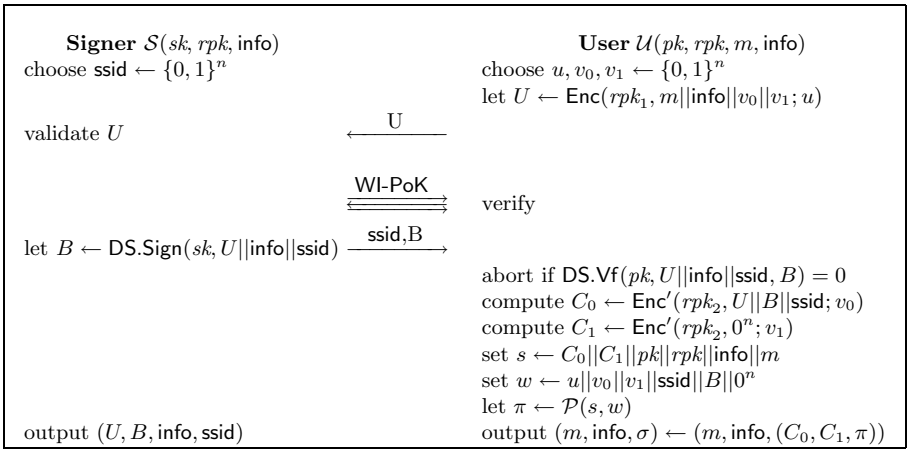



**Fig. 3.** Issue protocol of the fair partially blind signature scheme FPBS

**Key Generation.** $\mathsf{Kg}(1^n)$ executes the key generation algorithm of the signature scheme $(sk, pk) \leftarrow \mathsf{DS.Kg}(1^n)$. It chooses a suitable one-way function $f$, two elements $x_0, x_1 \leftarrow \{0,1\}^n$ and sets $y_0 \leftarrow f(x_0)$ and $y_1 \leftarrow f(x_1)$. Kg also computes the verifier's first message $\rho \leftarrow \mathcal{V}(1^n)$ for the ZAP. The public key is $pk \leftarrow (pk, y_0, y_1, \rho)$ and the corresponding secret key is $sk \leftarrow (sk, x_0, x_1)$.

**Revocation Key Generation.** $\mathsf{RevKg}(1^n)$ runs the key generation algorithm of both public-key encryption schemes $(rsk_1, rpk_1) \leftarrow \mathsf{EncKg}(1^n)$, $(rsk_2, rpk_2) \leftarrow \mathsf{EncKg}'(1^n)$ and outputs the secret revocation key $rsk \leftarrow (rsk_1, rsk_2)$ and the corresponding public key $rpk \leftarrow (rpk_1, rpk_2)$.

**Signature Issue Protocol.** The signature issue protocol is shown in Figure 3. Note that user and signer both include the same info in their computations, thus if one of the parties changes their mind about info during the execution then no valid signature can be produced.

- The user encrypts the message $m$ together with the info element and the randomness for the second encryption under the public key of the trusted party, and sends $U$ to the signer.
- The signer checks that the components $U$ is of the correct form, namely a block-wise encryption of four fields of length $n$ (each). Such an encryption scheme can be realized by concatenating IND-CPA secure encryptions of the individual fields. This check is essential. Otherwise, a malicious user can easily break the scheme.[1] It then initiates an interactive

[1] The malicious user $\mathcal{U}^*$ in the unforgeability experiment executes the signer on a random message $m \in \{0,1\}^n$ and an arbitrary info element $\mathsf{info} \in \{0,1\}^{n/2}$. It selects another $\mathsf{info}' \in \{0,1\}^{n/2}$, computes $U$ according to the protocol and sends $U' \leftarrow U||\mathsf{info}'$ to the signer. Following the protocol, $\mathcal{U}^*$ receives the signature $B$, computes $\sigma \leftarrow (C, \pi)$ and returns the tuple $(m, \mathsf{info}'||\mathsf{info}, \sigma)$. This tuple is a valid forgery because $\mathcal{U}^*$ never queried $\mathcal{S}$ on $\mathsf{info}'' \leftarrow \mathsf{info}'||\mathsf{info}$.

witness-indistinguishable proof of knowledge, where the signer proves knowledge of either $f^{-1}(y_0)$ or $f^{-1}(y_1)$.[2]

- The signer sends its signature $B$ and the randomly chosen session identifier $\mathsf{ssid}$.
- $\mathcal{U}$ encrypts the signature $B$ and computes a ZAP proving that the encryption *either* contains a valid message-signature pair *or* a preimage of $y_0$ or $y_1$. We now define the corresponding $\mathcal{NP}$-relation. Recall that $L$ is an $\mathcal{NP}$-language and that a ZAP is witness-indistinguishable. Thus, we can include a second witness (the "or ..." part below), which allows full simulation during the security reduction. To compute the final proof, the user computes the ZAP $\pi$ with respect to the first message $\rho$ (stored in the public key) for $s \leftarrow C_0||C_1||pk||rpk||\mathsf{info}||m \in L$ with witness $w \leftarrow u||v_0||v_1||\mathsf{ssid}||B||x$ such that membership in $L$ is established by:

$$U \leftarrow \mathsf{Enc}(rpk_1, m||\mathsf{info}||v_0||v_1; u) \wedge C_0 = \mathsf{Enc}'(rpk_2, U||B||\mathsf{ssid}; v_0) \wedge \mathsf{DS.Vf}(pk, U||\mathsf{info}||\mathsf{ssid}, B) = 1$$
$$\text{or } C_1 = \mathsf{Enc}'(rpk_2, x; v_1) \wedge f(x) \in \{y_0, y_1\}.$$

**Signature Verification.** $\mathsf{Vf}(pk, rpk, m, \mathsf{info}, \sigma)$ parses the signature $\sigma$ as $(C_0, C_1, \pi)$. It returns the result of $\mathcal{V}(s, \pi)$ for $s \leftarrow C_0||C_1||pk||rpk||\mathsf{info}||m$.

**Signature Revocation.** $\mathsf{SigRev}(rsk, \mathsf{view})$ parses the view $\mathsf{view}$ as $(U, B, \mathsf{info}, \mathsf{ssid})$. It decrypts $U$, computing $m||\mathsf{info}||v_0||v_1 \leftarrow \mathsf{Dec}(rsk_1, U)$ and extracts the randomness $v_0$. $\mathsf{SigRev}$ then encrypts $U||B||\mathsf{ssid}$ using the randomness $v_0$, obtaining $C_0' \leftarrow \mathsf{Enc}'(rpk_2, U||B||\mathsf{ssid}; v_0)$ and outputs $\mathsf{id}_{\mathsf{Sig}} \leftarrow C_0'$.

**Session Revocation.** $\mathsf{SesRev}(rsk, m, \mathsf{info}, \sigma)$ takes as input a message-signature tuple $(m, \mathsf{info}, (C_0, C_1, \pi))$ and the secret revocation key $rsk = (rsk_1, rsk_2)$. It extracts $B'$, computing $U||B'||\mathsf{ssid} \leftarrow \mathsf{Dec}(rsk_2, C_0)$ and returns $\mathsf{id}_{\mathsf{Ses}} \leftarrow (B', \mathsf{ssid})$.

**Signature Tracing.** $\mathsf{SigVf}(\mathsf{id}_{\mathsf{Sig}}, \sigma)$ parses $\mathsf{id}_{\mathsf{Sig}}$ as $C_0'$ and $\sigma$ as $(C_0, C_1, \pi)$. It outputs 1 iff $C_0' = C_0$.

**Session Tracing.** $\mathsf{SesVf}(\mathsf{id}_{\mathsf{Ses}}, \mathsf{view})$ parses $\mathsf{id}_{\mathsf{Ses}}$ as $(B', \mathsf{ssid}')$ and $\mathsf{view}$ as $(U, B, \mathsf{info}, \mathsf{ssid})$. It returns 1 iff $B' = B$ and $\mathsf{ssid}' = \mathsf{ssid}$.

**Theorem 1 (Partial Blindness).** *Let* $\mathsf{DS}$ *be a strongly unforgeable signature scheme,* $(\mathsf{EncKg}, \mathsf{Enc}, \mathsf{Dec})$ *and* $(\mathsf{EncKg}', \mathsf{Enc}', \mathsf{Dec}')$ *be two IND-CPA secure encryption schemes, and* $(\mathcal{P}, \mathcal{V})$ *be a ZAP. Then* $\mathsf{FPBS}$ *is partially blind.*

*Proof.* We prove partial blindness similar to [11,16]. That is, we transform the way the signatures are computed such that both signatures are, in the end, completely independent of the bit $b$. We then argue that the success probability, while transforming the experiment, does not change the success probability of the adversary more than in a negligible amount.

In the first step, we modify the experiment of fair partially blind signature scheme in the following way: During signature issue, the user algorithm $\mathcal{U}_0$ executes the extraction algorithm $\mathsf{Extract}$ of the witness-indistinguishable proof of knowledge ($\mathsf{WI\text{-}PoK}$) and computes the signature using the extracted witness.

[2] This proof allows the simulator in the blindness experiment to extract the witness and to use it instead of the signature.

In case the proof is valid, but extraction fails, the experiment outputs a random bit. At the end of the issue protocol, the user computes the signature using the extracted witness (all other algorithms remain unchanged). More precisely:

**Signature Issue Protocol ($\mathsf{FPBS}'$).** The issuing protocol is as before, except for the user $\mathcal{U}_0$. It runs the extraction algorithm $x' \leftarrow \mathsf{Extract}$ of the proof of knowledge. It aborts if $f(x') \notin \{y_0, y_1\}$. Otherwise, $\mathcal{U}_0$ executes $\mathcal{P}$ on input $(s, w) \leftarrow (C_0||C_1||pk||rpk||\mathsf{info}||m, 0^{4n}||0^{l(n)}||x')$, where $l(n)$ is the length of a signature. It outputs $\sigma \leftarrow (C_0, C_1, \pi)$. $\mathcal{U}_1$ remains unchanged.

It follows easily from the witness-indistinguishability property that both, experiment $\mathsf{Blind}_{\mathcal{S}^*}^{\mathsf{FPBS}}$ and $\mathsf{Blind}_{\mathcal{S}^*}^{\mathsf{FPBS}'}$, output 1 with the same probability (except for a negligible deviation).

In the next step, we modify the signature issue protocol of $\mathsf{FPBS}'$ such that the user algorithm sends an encryption to all-zero strings in the first round and computes the proof with the previously extracted witness again:

**Signature Issue Protocol ($\mathsf{FPBS}''$).** The signature issue protocol remains the same, except for the user algorithm that computes $U \leftarrow \mathsf{Enc}(rpk_1, 0^{4n}; u)$, and $C_1 \leftarrow \mathsf{Enc}'(rpk_2, x'; v_1)$.

We denote the modified scheme with $\mathsf{FPBS}''$. The IND-CPA property of the encryption scheme guarantees that this modification goes unnoticed (negligible change in $\mathcal{S}^*$' success probability).

In the last step, the user algorithm encrypts only all-zero strings in $C_0$ and the previously extracted witness $x$ in $C_1$.

**Signature Issue Protocol ($\mathsf{FPBS}'''$).** The user $\mathcal{U}$ calculates $C_0 \leftarrow \mathsf{Enc}'(rpk_2, 0^{4c(n)}||0^{l(n)}||0^n; v_0)$ and $C_1 \leftarrow \mathsf{Enc}'(rpk_2, x'; v_1)$.

Denoting the scheme with $\mathsf{FPBS}'''$, we argue by the IND-CPA property that the success probability of $\mathcal{S}^*$ stays essentially the same (except for a negligible deviation). In this protocol, the signature $\sigma = (C_0, C_1, \pi)$ is completely independent of $U$ and $B$. We conclude that the malicious signer $\mathcal{S}^*$ (against $\mathsf{FPBS}$) cannot predict the bit $b$ with noticeable advantage over $1/2$. □

**Theorem 2 (Signature Traceability).** *Let* $\mathsf{DS}$ *be a strongly unforgeable signature scheme,* $(\mathsf{EncKg}, \mathsf{Enc}, \mathsf{Dec})$ *and* $(\mathsf{EncKg}', \mathsf{Enc}', \mathsf{Dec}')$ *be two IND-CPA secure encryption schemes, and* $(\mathcal{P}, \mathcal{V})$ *be a ZAP. Then* $\mathsf{FPBS}$ *is signature traceable.*

*Proof.* First of all recall that an adversary $\mathcal{A}$ has three possibilities to win the signature traceability experiment. The first is to compute a message-signature tuple $(m, \mathsf{info}, \sigma)$ such that there exists no matching view. Note that since only views with respect to $\mathsf{info}$ are considered, the second possibility is to output a message-signature tuple corresponding to an execution with a different information element $\mathsf{info}'$. The third way the attacker $\mathcal{A}$ may win the game is to return two message-signatures related to the same view. In the following proof, we distinguish these three cases:

- An adversary is called a type-1 attacker, denoted by $\mathcal{A}_1$, if it manages to output a valid message-signature tuple $(m, \mathsf{info}, \sigma)$, where $\sigma = (C, \pi)$, such that $\mathsf{SigVf}(\mathsf{id}_{\mathsf{Sig}}, \mathsf{view}) = 0$ for all $\mathsf{id}_{\mathsf{Sig}} \leftarrow \mathsf{SigRev}(rsk, \mathsf{view})$ with $\mathsf{view} \in V_*$.
- An algorithm is called a type-2 attacker, denoted by $\mathcal{A}_2$, if it outputs a valid message-signature tuple $(m, \mathsf{info}, \sigma)$ such that there exists a $\mathsf{view} \in V_* \setminus V_{\mathsf{info}}$ with $\mathsf{SigVf}(\mathsf{id}_{\mathsf{Sig}}, \sigma) = 1$ for $\mathsf{id}_{\mathsf{Sig}} \leftarrow \mathsf{SigRev}(rsk, m, \mathsf{view})$.
- An attacker is called a type-3 attacker, denoted by $\mathcal{A}_3$, if it returns two valid message-signature tuple $(m_1, \mathsf{info}_1, \sigma_1)$ and $(m_2, \mathsf{info}_2, \sigma_2)$ with $m_1 \neq m_2$ such that $\mathsf{SigVf}(\mathsf{id}_{\mathsf{Sig}}, \sigma_1) = \mathsf{SigVf}(\mathsf{id}_{\mathsf{Sig}}, \sigma_2) = 1$ for $\mathsf{id}_{\mathsf{Sig}} \leftarrow \mathsf{SigRev}(rsk, \mathsf{view})$ and some $\mathsf{view} \in V_*$.

Before proving the theorem, recall that the ZAP $\pi$ only guarantees that *either* a valid signature *or* a value $x$, with $f(x) \in \{y_0, y_1\}$, has been encrypted. Thus, we have to distinguish these two cases throughout all proofs. In the first part of the proof, we show how to invert the one-way function $f$ if a preimage $x$ of $y_0$ or $y_1$ is encrypted. We stress that we show the reduction only once because it is the same in all proofs.

Type-1 Attacker. For the first type of adversaries, we have to distinguish, again, two cases: a) the encryption $C$ contains a value $x$ such that $f(x) \in \{y_0, y_1\}$ and b) $C$ contains $U||B||\mathsf{ssid}$. We begin the proof with the type-1a adversaries, denoted with $\mathcal{A}_{1a}$, showing how to invert the one-way function $f$ if $x$ is encrypted by applying the technique of Feige and Shamir [12]. We cover the type-1b adversaries, denoted with $\mathcal{A}_{1b}$, showing how to forge the underlying signature scheme if $U\|B\|\mathsf{ssid}$ is encrypted.

Let $\mathcal{A}_{1a}$ be an efficient algorithm, which succeeds in the signature traceability experiment with noticeable probability. We show how to build an algorithm $\mathcal{B}_{1a}$ that efficiently inverts the one-way function $f$. Note that $\mathcal{A}_{1a}$ may either encrypt the preimage of $y_0$ or the preimage of $y_1$. Thus, we describe the algorithm that extracts the preimage of $y_0$ and argue, in the analysis, that the same algorithm can be used to extract the preimage of $y_1$ by "switching" $y_0$ and $y_1$.

**Setup.** Algorithm $\mathcal{B}_{1a}$ gets as input an image $y$ of a one-way function $f$. It selects a value $x_1 \in \{0,1\}^n$ at random, sets $y_1 \leftarrow f(x_1)$, generates a key pair for the signature scheme $(sk, pk) \leftarrow \mathsf{DS.Kg}(1^n)$, as well as a key pair for the simulation of the trusted party $(rsk, rpk) \leftarrow \mathsf{RevKg}(1^n)$, and computes the first message of the ZAP $\rho \leftarrow \mathcal{V}(1^n)$. $\mathcal{B}_{1a}$ runs $\mathcal{A}_{1a}$ on input $((pk, y, y_1, \rho), rsk, rpk)$ in a black-box simulation.

**Signing Queries.** If $\mathcal{A}_{1a}$ initiates the signing protocol with its signing oracle then $\mathcal{B}_{1a}$ behaves in the following way. It receives the first message $U$ and $\mathsf{info}$ from $\mathcal{A}_{1a}$. It first checks that $U$ has four fields, each of length $c(n)$. If so, it executes the witness-indistinguishable protocol (using the witness $x_1$) and, after terminating the protocol, $\mathcal{B}_{1a}$ randomly selects a session identifier $\mathsf{ssid}$. It returns $\mathsf{ssid}$ and $B \leftarrow \mathsf{DS.Sign}(sk, U||\mathsf{info}||\mathsf{ssid})$.

**Output.** Finally, $\mathcal{A}_{1a}$ stops, outputting a tuple $(m, \mathsf{info}, \sigma)$ with $\sigma = (C, \pi)$. Algorithm $\mathcal{B}_{1a}$ computes $x' \leftarrow \mathsf{Dec}(rsk, C)$ and outputs $x'$.

For the analysis, it is assumed that $\mathcal{A}_{1a}$ outputs a valid message-signature tuple, either containing a witness for $f^{-1}(y_0)$ or for $f^{-1}(y_1)$, with noticeable probability $\epsilon(n)$. With $\epsilon_b(n)$, we denote the probability that $\mathcal{A}_{1a}$ encrypts a witness $x$ for $f^{-1}(y_b)$. Obviously, either $\epsilon_0(n) \geq \epsilon(n)/2$ or $\epsilon_1(n) \geq \epsilon(n)/2$ and, according to the description of $\mathcal{B}_{1a}$, we assume w.l.o.g. that the former holds. But if $\epsilon_0(n)$ is not negligible then we already have a contradiction because $\mathcal{B}_{1a}$ outputs a preimage $x' = f^{-1}(y_0)$. Thus, $\epsilon_0(n)$ must be negligible and $\epsilon_1(n)$ has to be noticeable.

In this case, however, we can execute the same algorithm with switched public keys, i.e., we run $\mathcal{A}_{1a}$ on input $y_1, y$ instead of $y, y_1$. Thanks to the witness-indistinguishability of the proof system, one can show (cf. [10]) that switching $y$ and $y_1$ changes the success probability of $\mathcal{A}_{1a}$ only in a negligible amount. Therefore, $|\epsilon_0(n) - \epsilon_1(n)|$ is negligible.

Next, we show that the second type of adversary ($\mathcal{A}_{1b}$) directly contradicts the unforgeability of the underlying signature scheme. We build an algorithm $\mathcal{B}_{1b}$, which uses $\mathcal{A}_{1b}$ in a black-box simulation.

**Setup.** $\mathcal{B}_{1b}$ takes as input a public verification key $pk$ and has access to a singing oracle. It executes the key generation algorithm of the encryption scheme $(rsk, rpk) \leftarrow \mathsf{EncKg}(1^n)$, selects two elements $x_0, x_1$ at random and sets $y_0 \leftarrow f(x_0), y_1 \leftarrow f(x_1)$. The algorithm then computes the first round of the ZAP $\rho \leftarrow \mathcal{V}(1^n)$ and executes $\mathcal{A}_{1b}$ on input $((pk, y_0, y_1, \rho), rsk, rpk)$.

**Signing Queries.** Whenever $\mathcal{A}_{1b}$ invokes the interactive signing protocol on some message $U$ and some info element $\mathsf{info}$, algorithm $\mathcal{B}_{1b}$ behaves as follows. It first checks that $U$ comprises four fields of length $c(n)$. If the assertion holds, it follows the witness-indistinguishable protocol using the witness $x_1$. Afterwards, $\mathcal{B}_1$ selects a session identifier $\mathsf{ssid}$ at random and invokes its signing oracle on $m \leftarrow U||\mathsf{info}||\mathsf{ssid}$, receiving the signature $\sigma$. Algorithm $\mathcal{B}_{1b}$ sends $\mathsf{ssid}, B \leftarrow \sigma$ back to $\mathcal{A}_{1b}$.

**Output.** Finally, $\mathcal{A}_{1b}$ stops, outputting a possibly valid tuple $(m', \mathsf{info}', \sigma')$ with $\sigma' = (C', \pi')$. $\mathcal{B}_{1b}$ decrypts $C$ obtaining $U'||B'||\mathsf{ssid}'$. It outputs $(m^*, \sigma^*) \leftarrow (U'||\mathsf{info}'||\mathsf{ssid}', B')$.

For the analysis, first note that $\mathcal{B}_{1b}$ is efficient since $\mathcal{A}_{1b}$ is and that $\mathcal{B}_{1b}$ performs a perfect simulation from $\mathcal{A}_{1b}$'s point of view. We assume that $\mathcal{A}_{1b}$ is successful and that $C$ does *not* contain the encryption of a witness. Since there is no matching view $\mathsf{view} = (U, B, \mathsf{info}, \mathsf{ssid})$, it follows that either $\mathcal{B}_{1b}$ never queried $m^*$ to its oracle or never received the same signature $B$ for the message $m^*$. Thus, $\mathcal{B}_{1b}$ succeeds whenever $\mathcal{A}_{1b}$ does.

Type-2 Attacker. The second class of attackers ($\mathcal{A}_2$) manages to output a message-signature tuple $(m, \mathsf{info}, \sigma)$ with $\sigma = (C, \pi)$ such that a "foreign" matching view exists, i.e., $\mathsf{SigVf}(\mathsf{id}_{\mathsf{Sig}}, \sigma) = 1$ for $\mathsf{id}_{\mathsf{Sig}} \leftarrow \mathsf{SigRev}(rsk, \mathsf{view})$ with $\mathsf{view} \in V_* \setminus V_{\mathsf{info}}$. We show that if a matching view exists, then the message, the information element, and the session id have to be the same.

Let the matching view be $\mathsf{view} = (U', B', \mathsf{info}', \mathsf{ssid}')$ and let $U||B||\mathsf{ssid} \leftarrow \mathsf{Dec}(rsk, C)$ be the decryption of the signature received from the malicious user.

Since the verification algorithm evaluates to 1, and because $x$ such that $f(x) \in \{y_0, y_1\}$ is *not* contained in $C$, it follows that $C' = C$ (otherwise $\mathsf{SigVf}(\mathsf{id}_{\mathsf{Sig}}, \sigma) = 0$). This, however, implies that $U = U'$, $\mathsf{ssid} = \mathsf{ssid}'$, and $B = B'$. Thus $m' = m$, $\mathsf{info}' = \mathsf{info}$, and $v' = v$ where $m'||\mathsf{info}'||v' \leftarrow \mathsf{Dec}(rsk, U')$ is the decryption of $U'$ contained in $\mathsf{view}$. Thus, such an attacker does not exist.

Type-3 Attacker. Let $\mathcal{A}_3$ be an adversary, which outputs two valid message-signature tuples $(m_1, \mathsf{info}_1, \sigma_1)$ and $(m_2, \mathsf{info}_2, \sigma_2)$ with $\sigma_i = (C_i, \pi_i)$, for $i = 1, 2$, such that $(m_1, \mathsf{info}_1) \neq (m_2, \mathsf{info}_2)$. We first show that the tuples $(m_1, \mathsf{info}_1, \sigma_1)$ and $(m_2, \mathsf{info}_2, \sigma_2)$ cannot stem from the same view.

W.l.o.g., suppose that $\mathsf{view} = (U, B, \mathsf{info}, \mathsf{ssid}) \in V_*$ is the corresponding view to the message-signature tuple $(m_1, \mathsf{view}_1, \sigma_1)$. Recall that the signature revocation algorithm $\mathsf{SigRev}$ reveals the randomness $v$ from $\mathsf{view}$, computing $m||\mathsf{info}||v \leftarrow \mathsf{Dec}(rsk, U)$, and generates $C' \leftarrow \mathsf{Enc}(rpk, U||B||\mathsf{ssid}; v)$. Since $\mathsf{view}$ is also a matching view for the first message-signature tuple, we infer that $C' = C_1$ and consequently $C' = C_2$. Otherwise, the verification algorithm would not return 1 on both inputs. Since $\sigma_2 = (C_2, \pi_2)$ is a valid signature, the prove $\pi_2$ is valid and it follows that $C_2$ is the ciphertext for $U_2||B_2||\mathsf{ssid}_2$. But this string equals $U_1||B_1||\mathsf{ssid}_1$ and therefore $m_1 = m_2$, $\mathsf{info}_1 = \mathsf{info}_2$, and $v_1 = v_2$. This, however, contradicts the assumption that $(m_1, \mathsf{info}_1) \neq (m_2, \mathsf{info}_2)$ and therefore such a pair cannot exist. □

**Theorem 3 (Session Traceability).** *Let* $\mathsf{DS}$ *be a strongly unforgeable signature scheme,* $(\mathsf{EncKg}, \mathsf{Enc}, \mathsf{Dec})$ *and* $(\mathsf{EncKg}', \mathsf{Enc}', \mathsf{Dec}')$ *be two IND-CPA secure encryption schemes and* $(\mathcal{P}, \mathcal{V})$ *be a ZAP. Then* $\mathsf{FPBS}$ *is session traceable.*

*Proof.* We first show that if a signature verifies then it is always possible to disclose the corresponding session and second, that each signature has a unique identifier and therefore a unique session. Analogously to the proof of signature traceability, we distinguish three cases:

- An algorithm $\mathcal{A}_1$, which outputs a valid message-signature tuple $(m, \mathsf{info}, \sigma)$, where $\sigma = (C, \pi)$, such that $\mathsf{SesVf}(\mathsf{id}_{\mathsf{Ses}}, \mathsf{view}) = 0$ for all $\mathsf{view} \in V_*$ with $\mathsf{id}_{\mathsf{Ses}} \leftarrow \mathsf{SesRev}(rsk, m, \mathsf{info}, \sigma)$.
- An attacker $\mathcal{A}_2$, which returns a valid message-signature tuple $(m, \mathsf{info}, \sigma)$, such that there exists a $\mathsf{view} \in V_* \setminus V_{\mathsf{info}}$ with $\mathsf{SesVf}(\mathsf{id}_{\mathsf{Ses}}, \sigma) = 1$ for $\mathsf{id}_{\mathsf{Ses}} \leftarrow \mathsf{SesRev}(rsk, m, \mathsf{info}, \sigma)$.
- An adversary $\mathcal{A}_3$, which manages to output a valid message-signature tuple $(m, \mathsf{info}, \sigma)$, such that there exist two distinct views $\mathsf{view}_1, \mathsf{view}_2 \in V_*$ with $\mathsf{SesVf}(\mathsf{id}_{\mathsf{Ses}}, \mathsf{view}_1) = \mathsf{SesVf}(\mathsf{id}_{\mathsf{Ses}}, \mathsf{view}_2) = 1$ for $\mathsf{id}_{\mathsf{Ses}} \leftarrow \mathsf{SesRev}(rsk, m, \mathsf{info}, \sigma)$.

The reductions for the first two cases are identical to the first two reductions in the proof of Theorem 2. Thus, we omit it here.

What is left to show is that two different views cannot match to a single execution. This follows from the fact that the signer generates a new session identifier $\mathsf{ssid}$ during each execution. Thus, with overwhelming probability over the choice of $\mathsf{ssid}$, each execution yields a new view. □

By combining the above results with an observation in Section 2, we get the following corollary.

**Corollary 1 (Unforgeability).** *Let* $\mathsf{DS}$ *be a strongly unforgeable signature scheme,* $(\mathsf{EncKg}, \mathsf{Enc}, \mathsf{Dec})$ *be an IND-CPA secure encryption scheme with message expansion function* $c$*, and* $(\mathcal{P}, \mathcal{V})$ *be a ZAP. Then* $\mathsf{FPBS}$ *is unforgeable.*

## Acknowledgments

The authors thank the anonymous reviewers of Africacrypt 2010 for their valuable comments. They are indebted to one particular reviewer who provided exceptionally detailed comments and helped in improving the presentation as well as eliminating inconsistencies. The MD5 hash of the first sentence in his or her review was `3641f58bff4934c06d9e71afcc3e14fe`.

## References

1. Abe, M., Fujisaki, E.: How to Date Blind Signatures. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 244–251. Springer, Heidelberg (1996)
2. Abe, M., Okamoto, T.: Provably Secure Partially Blind Signatures. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 271–286. Springer, Heidelberg (2000)
3. Abe, M., Ohkubo, M.: Provably Secure Fair Blind Signatures with Tight Revocation. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 583–602. Springer, Heidelberg (2001)
4. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
5. Bellare, M., Shi, H., Zhang, C.: Foundations of Group Signatures: The Case of Dynamic Groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
6. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004)
7. Chaum, D.: Blind Signatures for Untraceable Payments. In: Advances in Cryptology — Crypto 1982, pp. 199–203. Plemum, New York (1983)
8. Chow, S.S.M., Hui, L.C.K., Yiu, S.M., Chow, K.P.: Two Improved Partially Blind Signature Schemes from Bilinear Pairings. Cryptology ePrint Archive, Report 2004/108 (2004), `http://eprint.iacr.org/`
9. Dwork, C., Naor, M.: Zaps and Their Applications. SIAM Journal on Computing 36(6), 1513–1543 (2007)
10. Feige, U.: Alternative Models for Zero-Knowledge Interactive Proofs. PhD Thesis. Weizmann Institute of Science. Dept. of Computer Science and Applied Mathematics (1990), `http://www.wisdom.weizmann.ac.il/~feige`
11. Fischlin, M.: Round-Optimal Composable Blind Signatures in the Common Reference String Model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)
12. Feige, U., Shamir, A.: Zero Knowledge Proofs of Knowledge in two Rounds. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 526–544. Springer, Heidelberg (1990)

13. Fischlin, M., Schröder, D.: Security of Blind Signatures Under Aborts. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 297–316. Springer, Heidelberg (2009)
14. Frankel, Y., Tsiounis, Y., Yung, M.: Indirect Discourse Proof: Achieving Efficient Fair Off-Line E-cash. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 286–300. Springer, Heidelberg (1996)
15. Fuchsbauer, G., Vergnaud, D.: Fair Blind Signatures without Random Oracles. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 16–33. Springer, Heidelberg (2010)
16. Hazay, C., Katz, J., Koo, C.-Y., Lindell, Y.: Concurrently-Secure Blind Signatures Without Random Oracles or Setup Assumptions. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 323–341. Springer, Heidelberg (2007)
17. Hufschmitt, E., Traoré, J.: Fair Blind Signatures Revisited. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 268–292. Springer, Heidelberg (2007)
18. Juels, A., Luby, M., Ostrovsky, R.: Security of Blind Digital Signatures. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 150–164. Springer, Heidelberg (1997)
19. Jakobsson, M., Yung, M.: Distributed "Magic ink" signatures. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 450–464. Springer, Heidelberg (1997)
20. Lee, H.-W., Kim, T.-Y.: Message Recovery Fair Blind Signature. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 97–111. Springer, Heidelberg (1999)
21. Miyazaki, S., Sakurai, K.: A More Efficient Untraceable E-Cash System with Partially Blind Signatures Based on the Discrete Logarithm Problem. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 296–307. Springer, Heidelberg (1998)
22. Okamoto, T.: Efficient Blind and Partially Blind Signatures Without Random Oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)
23. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology 13(3), 361–396 (2000)
24. Stadler, M., Piveteau, J.-M., Camenisch, J.: Fair Blind Signatures. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 209–219. Springer, Heidelberg (1995)
25. von Solms, S.H., Naccache, D.: On blind signatures and perfect crimes. Computers & Security 11(6), 581–583 (1992)

# Parallel Shortest Lattice Vector Enumeration on Graphics Cards*

Jens Hermans[1,**], Michael Schneider[2], Johannes Buchmann[2], Frederik Vercauteren[1,***], and Bart Preneel[1]

[1] Katholieke Universiteit Leuven - ESAT/SCD-COSIC and IBBT
{Jens.Hermans,Frederik.Vercauteren,Bart.Preneel}@esat.kuleuven.be
[2] Technische Universität Darmstadt
{mischnei,buchmann}@cdc.informatik.tu-darmstadt.de

**Abstract.** In this paper we present an algorithm for parallel exhaustive search for short vectors in lattices. This algorithm can be applied to a wide range of parallel computing systems. To illustrate the algorithm, it was implemented on graphics cards using CUDA, a programming framework for NVIDIA graphics cards. We gain large speedups compared to previous serial CPU implementations. Our implementation is almost 5 times faster in high lattice dimensions.

Exhaustive search is one of the main building blocks for lattice basis reduction in cryptanalysis. Our work results in an advance in practical lattice reduction.

**Keywords:** Lattice reduction, ENUM, parallelization, graphics cards, CUDA, exhaustive search.

## 1 Introduction

Lattice-based cryptosystems are assumed to be secure against quantum computer attacks. Therefore these systems are promising alternatives to factoring or discrete logarithm based systems. The security of lattice-based schemes is based on the hardness of special lattice problems. Lattice basis reduction helps to determine the actual hardness of those problems in practice. In the past few years there has been increased attention to exhaustive search algorithms for lattices, especially to implementation aspects. In this paper we consider parallelization and special hardware for the exhaustive search.

* The work described in this report has in part been supported by the Commission of the European Communities through the ICT program under contract ICT-2007-216676. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

** Research assistant, sponsored by the Fund for Scientific Research - Flanders (FWO).

*** Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 52–68, 2010.

Lattice reduction is the search for short and orthogonal vectors in a lattice. The algorithm used for lattice reduction in practice today is the BKZ algorithm of Schnorr and Euchner [SE91]. It consists of two main parts, namely an exhaustive search ('enumeration') for shortest, non-zero vectors in lower dimensions and the LLL algorithm [LLL82] for the search for short (not *shortest*) vectors in high dimensions. The BKZ algorithm is parameterized by a blocksize parameter $\beta$, which determines the blocksize of the exhaustive search algorithm inside BKZ.

Algorithms for exhaustive search were presented by Kannan [Kan83] and by Fincke and Pohst [FP83]. Therefore, the enumeration is sometimes referred to as KFP-algorithm. Kannan's algorithm runs in $2^{\mathcal{O}(n \log n)}$ time, where $n$ denotes the lattice dimension. Schnorr and Euchner presented a variant of the KFP exhaustive search, which is called ENUM [SE91]. Roughly speaking, enumeration algorithms perform a depth first search in a search tree that contains all lattice vectors in a certain search space, i.e., all vectors of Euclidean norm less than a specified bound. The main challenge is to determine which branches of the tree can be cut off to speed up the exhaustive search. Enumeration is always executed on lattice bases that are at least LLL reduced in a preprocessing step, as this reduces the runtime significantly compared to non-reduced bases.

The LLL algorithm runs in time polynomial in the lattice dimension and therefore can be applied in high lattice dimensions ($n > 1000$). The runtime of all known exhaustive search algorithms is exponential in the dimension, and therefore can only be applied in blocks of smaller dimension ($n \lessapprox 70$). With this, the runtime of BKZ increases exponentially in the blocksize $\beta$. As in BKZ, enumeration is executed very frequently, it is only practical to choose blocksizes up to 50. For high blocksize, our experience shows that ENUM takes 99% of the time of BKZ.

There are numerous works on parallelization of LLL [Vil92, HT98, RV92, Jou93] [Wet98, BW09]. Parallel versions of lattice enumeration were presented in the masters theses of Pujol [Puj08] and Dagdelen [Dag09] (in french and german language, respectively). Both approaches are not suitable for GPU, since they require dynamic creation of new threads, which is not possible for GPUs.

Being able to parallelize ENUM means to parallelize the second (more time consuming) building block of BKZ, which reduces the runtime of the most promising lattice reduction algorithm in total.

As a platform for our parallel implementation we have chosen graphical processing units (GPUs). Because of their design to perform identical operations on large amounts of graphical data, GPUs can run large numbers of threads in parallel, provided the threads execute similar instructions. We can take advantage of this design and split up the ENUM algorithm over several identical threads. The computation power of GPU rises faster than that of CPUs over the last years, with respect to floating point operations per second (GFlops). This trend is not supposed to stop, therefore using GPUs for computation will be a useful model also in the near future.

*Our Contribution.* In this paper we present a parallel version of the enumeration algorithm of [SE91] that finds a shortest, non-zero vector in a lattice. Since the

enumeration algorithm is tree-based, the main challenge is splitting the tree in some way and executing subtree enumerations in parallel. We use the CUDA framework of NVIDIA for implementing the algorithm on graphics cards. Because of the choice for GPUs, parallelization and splitting are more difficult than for a CPU parallelization. Firstly we explain the ideas of how to parallelize enumeration on GPU. Secondly we present some first experimental results. Using the GPU, we reduce the time required for enumeration of a random lattice in dimensions higher than 50 by a factor of almost 5. We are using random lattices in the sense of Goldstein and Mayer [GM03] for testing our implementation.

The first part of this paper, namely the idea of parallelizing enumeration, can also be applied on multicore CPU. The idea of splitting the search tree into parts and search different subtrees independently in parallel is also applicable on CPU, or other parallel computing frameworks. As mentioned above, BKZ is only practical using blocksizes up to 50. As our GPU version of the enumeration performs best in dimensions $n$ greater than 50, we would expect to speed up BKZ with high blocksizes only.

In contrast to our algorithm, Pujol's idea [Puj08] is to predict the number of enumeration steps in a subtree beforehand, using a volume heuristic. If the number of expected steps in a subtree exceeds some bound, the subtree is split recursively, and enumerated as different threads. Dagdelen [Dag09] bounds the height of subtrees that can be split recursively. Both ideas differ from our approach, as we use a real-time scheduling; when a subtree enumeration has exceeded a specified number of enumeration steps it is stopped, to balance the load of all GPU kernels. This fits best into the SIMD structure of GPUs, as both existing approaches lead to a huge number of diverging subthreads.

*Structure of the Paper.* In Section 2 we introduce the necessary preliminaries on lattices and GPUs. We discuss previous lattice reduction algorithms and the applications of lattices in cryptography. The GPU (CUDA) programming model is shortly introduced, explaining in more detail the memory model and data types which are important for our implementation. Section 3 explains our parallel enumeration algorithm, starting from the ENUM algorithm of Schnorr and Euchner and ending with the iterated GPU enumeration algorithm. Section 4 discusses the results obtained with our algorithm.

## 2 Preliminaries

A lattice is a discrete subgroup of $\mathbb{R}^d$. It can be represented by a basis matrix $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ $(n \leq d)$. We call $\mathcal{L}(\mathbf{B}) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i, x_i \in \mathbb{Z}\}$ the lattice spanned by the column basis vectors $\mathbf{b}_i \in \mathbb{R}^d$ $(i = 1 \ldots n)$. The dimension $n$ of a lattice is the number of linear independent vectors in the lattice, i.e. the number of basis vectors. When $n = d$ the lattice is called full dimensional.

The basis of a lattice is not unique. Every unimodular transformation $\mathbf{M}$, i.e. integer transformation with $\det \mathbf{M} = \pm 1$, turns a basis matrix $\mathbf{B}$ into a second basis $\mathbf{MB}$ of the same lattice.

The determinant of a lattice is defined as $\det(\mathcal{L}(\mathbf{B})) = \sqrt{\det(\mathbf{B}^T\mathbf{B})}$. For full dimensional lattices we have $\det(\mathcal{L}(\mathbf{B})) = |\det(\mathbf{B})|$. The determinant of a lattice is invariant of the choice of the lattice basis, which follows from the multiplicative property of the determinant and the fact that basis transformations have determinant $\pm 1$.

The length of a shortest vector of a lattice $\mathcal{L}(\mathbf{B})$ is denoted $\lambda_1(\mathcal{L}(\mathbf{B}))$ or in short $\lambda_1$ if the lattice is uniquely determined.

The Gram-Schmidt algorithm computes an orthogonalization of a basis. It is an efficient algorithm that outputs $\mathbf{B}^* = [\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*]$ with $\mathbf{b}_i^*$ orthogonal and $\mu_{i,j}$ such that $\mathbf{B} = \mathbf{B}^* \cdot [\mu_{i,j}]$, where $[\mu_{i,j}]$ is an upper triangular matrix consisting of the Gram-Schmidt coefficients $\mu_{i,j}$ for $1 \leq j \leq i \leq n$. The orthogonalized matrix $\mathbf{B}^*$ is not necessarily a basis of the lattice.

### 2.1 Lattice Basis Reduction

*Problems.* Some lattice bases are more useful than others. The goal of lattice basis reduction (or in short lattice reduction) is to find a basis consisting of short and almost orthogonal lattice vectors. More exactly, we can define some (hard) problems on lattices. The most important one is the *shortest vector problem* (SVP), which consists of finding a vector $\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}$ with $\|\mathbf{v}\| = \lambda_1(\mathcal{L}(\mathbf{B}))$. In most cases, the Euclidean norm $\|\cdot\|_2$ is considered. As the SVP is $\mathcal{NP}$-hard (at least under randomized reductions) [Din02, Kho05, RR06] people consider the approximate version $\gamma$-SVP, that tries to find a vector $\mathbf{v} \in \mathcal{L} \setminus \{\mathbf{0}\}$ with $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$.

Other important problems like the *closest vector problem* (CVP) that searches for a nearest lattice vector to a given point in space, its approximation variant $\gamma$-CVP, or the *shortest basis problem* (SBP) are listed and described in detail in [MG02].

*Algorithms.* In 1982 Lenstra, Lenstra, and Lovász [LLL82] introduced the LLL algorithm, which was the first polynomial time algorithm to solve the approximate shortest vector problem in higher dimensions. Another algorithm is the BKZ block algorithm of Schnorr and Euchner [SE91]. In practice, this is the algorithm that gives the best solution to lattice reduction so far. Their paper [SE91] also introduces the enumeration algorithm (ENUM), a variant of the Fincke-Pohst [FP83] and Kannan [Kan83] algorithms. The ENUM algorithm is the fastest algorithm in practice to solve the exact shortest vector problem using complete enumeration of all lattice vectors in a suitable search space. It is used as a black box in the BKZ algorithm. The enumeration algorithm organizes linear combinations of the basis vectors in a search tree and performs a depth first search above the tree.

In [PS08] Pujol and Stehlé analyze the stability of the enumeration when using floating point arithmetic. In [HS07], improved complexity bounds for Kannan's algorithm are presented. This paper also suggests some better preprocessing of lattice bases, i.e., the authors suggest to BKZ reduce a basis before running enumeration. This approach lowers the runtime of enumeration. In this paper we consider both LLL and BKZ pre-reduced bases. [AKS01] show how to solve SVP using a randomized algorithm in time $2^{\mathcal{O}(n)}$, but their algorithm requires

exponential space and is therefore impractical. The papers [NV08] and [MV10] present improved sieving variants, where the Gauss-sieving algorithm of [MV10] is shown to be really competitive to enumeration algorithms in practically interesting dimensions.

Several LLL variants were presented by Schnorr [Sch03], Nguyen and Stehlé [NS05], and Gama and Nguyen [GN08a]. The variant of [NS05] is implemented in the `fpLLL` library of [CPS], which is also the fastest public implementation of ENUM algorithms. Koy introduced the notion of a primal-dual reduction in [Koy04]. Schnorr [Sch03] and Ludwig [BL06] deal with random sampling reduction. Both are slightly different concepts of lattice reduction, where primal-dual reduction uses the dual of a lattice for reducing and random sampling combines LLL-like algorithms with an exhaustive point search in a set of lattice vectors that is likely to contain short vectors.

The papers [SE91, SH95] present a probabilistic improvement of ENUM, called tree pruning. The idea is to prune subtrees that are unlikely to contain shorter vectors. As it leads to a probabilistic variant of the enumeration algorithm, we do not consider pruning techniques here.

In [GN08b] Gama and Nguyen compare the `NTL` implementation [Sho] of floating point LLL, the deep insertion variant of LLL and the BKZ algorithm. It is the first comprehensive comparison of lattice basis reduction algorithms and helps understanding their practical behavior.

In [Vil92, HT93, RV92] the authors present parallel versions for $n$ and $n^2$ processors, where $n$ is the lattice dimension. In [Jou93] the parallel LLL of Villard [Vil92] is combined with the floating point ideas of [SE91]. In [Wet98] the authors present a blockwise generalization of Villards algorithm. Backes and Wetzel worked out a parallel variant of the LLL algorithm for multi-core CPU architectures [BW09]. For the parallelization of lattice reduction on GPU the authors are not aware of any previous work.

*Applications.* Lattice reduction has applications in cryptography as well as in cryptanalysis. The foundation of some cryptographic primitives is based on the hardness of lattice problems. Lattice reduction helps determining the practical hardness of those problems and is a basis for real world application of those hash functions, signatures, and encryption schemes. Well known examples are the SWIFFT hash functions of Lyubashevsky et al. [LMPR08], the signature schemes of [LM08, GPV08, Lyu09, Pei09a], or the encryption schemes of [AD97, Pei09b, SSTX09]. The NTRU [HPS98, otCC09] and GGH [GGH97] schemes do not provide a security proof, but the best attacks are also lattice based.

There are also attacks on RSA and similar systems, using lattice reduction to find small roots of polynomials [CNS99, DN00, May10]. Low density knapsack cryptosystems were successfully attacked with lattice reduction [LO85]. Other applications of lattice basis reduction are factoring numbers and computing discrete logarithms using diophantine approximations [Sch91]. In Operations Research, or generally speaking, discrete optimization, lattice reduction can be used to solve linear integer programs [Len83].

### 2.2 Programming Graphics Cards

A Graphical Processing Units (GPUs) is a piece of hardware that is specifically designed to perform a massive number of specific graphical operations in parallel. The introduction of platforms like CUDA by NVIDIA [Nvi07a] or CTM by ATI [AMD06], that make it easier to run custom programs instead of limited graphical operations on a GPU, has been the major breakthrough for the GPU as a general computing platform. The introduction of integer and bit arithmetic also broadened the scope to cryptographic applications.

*Applications.* Many general mathematical packages are available for GPU, like the BLAS library [NVI07b] that supports basic linear algebra operations.

An obvious application in the area of cryptography is brute force searching using multiple parallel threads on the GPU. There are also implementations of AES [CIKL05, Man07, HW07] and RSA [MPS07, SG08, Fle07] available. GPU implementations can also be used (partially) in cryptanalysis. In 2008, Bernstein et al. use parallelization techniques on graphics cards to solve integer factorization using elliptic curves [BCC+09]. Using NVIDIA's CUDA parallelization framework, they gained a speed-up of up to 6 compared to computation on a four core CPU. However, to date, no applications based on lattices are available for GPU.

*Programming Model.* For the work in this paper the CUDA platform will be used. The GPUs from the Tesla range, which support CUDA, are composed of several multiprocessors, each containing a small number of scalar processors. For the programmer this underlying hardware model is hidden by the concept of SIMT-programming: Single Instruction, Multiple Thread. The basic idea is that the code for a single thread is written, which is then uploaded to the device and executed in parallel by multiple threads.

The threads are organized in multidimensional arrays, called blocks. All blocks are again put in a multidimensional array, called the *grid*. When executing a program (a grid), threads are scheduled in groups of 32 threads, called *warps*. Within a warp threads should not diverge, as otherwise the execution of the warp is serialized.

*Memory Model.* The Tesla GPUs provide multiple levels of memory: registers, shared memory, global memory, texture and constant memory. Registers and shared memory are on chip and close to the multiprocessor and can be accessed with low latency. The number of registers and shared memory is limited, since the number available for one multiprocessor must be shared among all threads in a single block.

Global memory is off-chip and is not cached. As such, access to global memory can slow down the computations drastically, so several strategies for speeding up memory access should be considered (besides the general strategy of avoiding global memory access). By coalescing memory access, e.g. loading the same memory address or a consecutive block of memory from multiple threads, the delay is reduced, since a coalesced memory access has the same cost as a single random memory access. By launching a large number of blocks the latency

introduced by memory loading can also be hidden, since other blocks can be scheduled in the meantime.

The constant and texture memory are cached and can be used for specific types of data or special access patterns.

*Instruction Set.* Modern GPUs provide the full range of (32 and) 64 bit floating point, integer and bit operations. Addition and multiplication are fast, other operations can, depending on the type, be much slower. There is no point in using other than 32 or 64 bit numbers, since smaller types are always cast to larger types. Most GPUs have a specialized FMAD instruction, which performs a floating point multiplication followed by an addition at the cost of only a single operation. This instruction can be used during the BKZ enumeration.

One problem that occurs on GPUs is the fact that today GPUs are not able to deal with higher precision than 64 bit floating point numbers. For lattice reduction, sometimes higher bit sizes are required to guarantee the correct termination of the algorithms. For an $n$-dimensional lattice, using the floating point LLL algorithm of [LLL82], one requires a precision of $\mathcal{O}(n \log B)$ bits, where $B$ is an upper bound for the length of the $d$-dimensional vectors [NS05]. For the $L^2$ algorithm of [NS05], the required bit size is $\mathcal{O}(n \log_2 3)$, which is independent of the norm of the input basis vectors. For more details on the floating point LLL analysis see [NS05] and [NS06].

In [PS08] the authors state that for enumeration algorithms double precision is suitable up to dimension 90, which is beyond the dimensions that are practical today. Therefore enumeration should be possible on actual graphics cards, whereas the implementation of LLL-like algorithms will be more complicated and require some multi-precision framework.

## 3 Parallel Enumeration on GPU

In this section we present our parallel algorithm for shortest vector enumeration in lattices. In Subsection 3.1 we briefly explain the ENUM algorithm of Schnorr and Euchner [SE91], which was used as a basis for our algorithm. Next, we present the basic idea for multi-thread enumeration in Subsection 3.2. Finally, in Subsection 3.3, we explain our parallel algorithm in detail.

The ENUM algorithm of Schnorr-Euchner is an improvement of the algorithms from [Kan83] and [FP83]. The ENUM algorithm is the fastest one today and also the one used in the `NTL` [Sho] and `fpLLL` [CPS] libraries. Therefore we have chosen this algorithm as basis for our parallel algorithm.

### 3.1 Original ENUM Algorithm

The ENUM algorithm enumerates over all linear combinations $[x_1, \ldots, x_n] \in \mathbb{Z}^n$ that generate a vector $\mathbf{v} = \sum_{i=1}^{n} x_i \mathbf{b}_i$ in the search space (i.e., all vectors $\mathbf{v}$ with $\|\mathbf{v}\|$ smaller than a specified bound). Those linear combinations are organized in a tree structure. Leafs of the tree contain full linear combinations, whereas inner nodes contain partly filled vectors. The search for the tree leaf that determines

the shortest lattice vector is performed in a depth first search order. The most important part of the enumeration is cutting off parts of the tree, i.e. the strategy which subtrees are explored and which ones cannot lead to a shorter vector.

Let $i$ be the current level in the tree, $i = 1$ being at the bottom and $i = n$ at the top of the tree (c.f. Figure 1). Each step in the enumeration algorithm consists of computing an *intermediate* squared norm $l_i$, moving one level up or down the tree (to level $i' \in \{i-1, i+1\}$) and determining a new value for the coordinate $x_{i'}$.

Let $r_i = \|\mathbf{b}_i^*\|^2$. We define $l_i = l_{i+1} + y_i^2 r_i$ with $y_i = x_i - c_i$ and $c_i = -\sum_{j=i+1}^{n} \mu_{j,i} x_j$. So, for a certain choice of coordinates $x_i \dots x_n$ it holds that $l_k \geq l_i$ (with $k < i$) for all coordinate vectors $\mathbf{x}$ that end with the same coordinates $x_i \dots x_n$. This implies that the intermediate norm $l_i$ can be used to cut off infeasible subtrees. If $l_i > A$, with $A$ the squared norm of the shortest vector that has been found so far, the algorithm will increase $i$ and move up inside the tree. Otherwise, the algorithm will lower $i$ and move down in the tree. Usually, as initial bound $A$ for the length of the shortest vector, one uses the norm of the first basis vector.

The next value for $x_{i'}$ is selected in an interval of length $\sqrt{\frac{A - l_{i'+1}}{r_{i'}}}$ centered at $c_{i'}$. The interval is enumerated according to the zig-zag pattern described in [SE91]. Starting from a central value $\lfloor c_{i'} \rceil$, ENUM will generate a sequence $\lfloor c_{i'} \rceil + 1, \lfloor c_{i'} \rceil - 1, \lfloor c_{i'} \rceil + 2, \lfloor c_{i'} \rceil - 2, \dots$ for the coordinate $x_{i'}$. To be able to generate such a pattern, helper vectors $\mathbf{\Delta x} \in \mathbb{Z}^n$ are used. We do not require to store $\mathbf{\Delta^2 x}$ as in the orginal algorithm [SE91, PS08], as the computation of the zigzag pattern is done in a slightly different way as in the original algorithm. For a more detailed description of the ENUM algorithm we refer to [PS08].

### 3.2 Multi-thread Enumeration

Roughly speaking, the parallel enumeration works as follows. The search tree of combinations that is explored in the enumeration algorithm can be split at a high level, distributing subtrees among several threads. Each thread then runs an enumeration algorithm, keeping the first coefficients fixed. These fixed coefficients are called *start vectors*. The subtree enumerations can run independently, which limits communication between threads. The top level enumeration is performed on CPU and outputs start vectors for the GPU threads.

When the number of postponed subtrees is higher than the number of threads that we can start in parallel, then we copy the start vectors to the GPU and let it enumerate the subtrees. After all threads have finished enumerating their subtrees we proceed in the same manner: caching start vectors on CPU and starting a batch of subtree enumerations on GPU. Figure 1 illustrates this approach. The variable $\alpha$ defines the region where the initial enumeration is performed. The subtrees where GPU threads work are also depicted in Figure 1.

If a GPU subtree enumeration finds a new optimal vector, it writes back the coordinates $\mathbf{x}$ and the squared norm $A$ of this vector to the main memory. The other GPU threads will directly receive the new value for $A$, which will allow them to cut away more parts of the subtree.

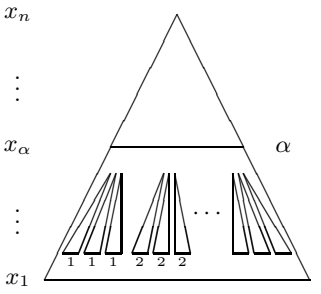



**Fig. 1.** Illustration of the algorithm flow. The top part is enumerated on CPU, the lower subtrees are explored in parallel on GPU. The tiny numbers illustrate which subtrees are enumerated in the same iteration.

*Early Termination.* The computation power of the GPU is used best when as many threads as possible are working at the same time. Recall that the GPU uses warps as the basic execution units: all threads in a warp are running the same instructions (or some of the threads in the warp are stalled in the case of branching).

In general, more starting vectors than there are GPU threads are uploaded in each run of the GPU kernel. This allows us to do some load balancing on the GPU, to make sure all threads are busy. To avoid the GPU being stalled by a few long running subtree enumerations, the GPU stops when just a few subtrees are left. We call this process, by which the GPU stops some subtrees even though they are not finished, *early termination*.

At the end of Section 3.3 details are included on the exact way early termination and our load balancing algorithm works. For now it suffices to know that, because of early termination, some of the subtree enumerations are not finished after a single launch of the GPU kernel. This is the main reason why the entire algorithm is iterated several times. At each iteration the GPU launches a mix of enumerations: new subtrees (start vectors) from the top enumeration and subtrees that were not finished in one of the previous GPU launches.

### 3.3 The Iterated Parallel ENUM Algorithm

Algorithm 1 shows the high-level layout of the GPU enumeration algorithm. Details concerning the updating of the bound $A$, as well as the write-back of newly discovered optimal vectors have been omitted. The actual enumeration is also not shown: it is part of several subroutines which are called from the main algorithm.

The whole process of launching a grid of GPU threads is iterated several times (line 2), until the whole search tree has been enumerated either on GPU or CPU.

In line 3, the top of the search tree is enumerated, to generate a set $S$ of starting vectors $\mathbf{x}_k$ for which enumeration should be started at level $\alpha$. More detailed, the top enumeration in the region between $\alpha$ and $n$ outputs distinct vectors

$$\mathbf{x}_k = [0, \ldots, 0, x_{k,\alpha}, \ldots, x_{k,n}] \qquad \text{for } k = 1 \ldots \text{NUMSTARTPOINTS} - \#T\,.$$

**Algorithm 1.** High-level Iterated Parallel ENUM Algorithm

**Input**: $\mathbf{b}_i (i = 1, \ldots, n)$, $A$, $\alpha$, $n$

1 Compute the Gram-Schmidt orthogonalization of $\mathbf{b}_i$
2 **while** *true* **do**
3 $S = \{(\mathbf{x}_k, \mathbf{\Delta x}_k, L_k = \alpha, s_k = 0)\}_k \leftarrow$ Top enum: generate at most NUMSTARTPOINTS $- \#T$ vectors
4 $R = \{(\bar{\mathbf{x}}_k, \mathbf{\Delta x}_k, L_k, s_k)\}_k \leftarrow$ GPU enumeration, starting from $S \cup T$
5 $T \leftarrow \{R_k : \text{subtree } k \text{ was not finished}\}$
6 **if** $\#T <$ CPUTHRESHOLD **then**
7 Enumerate the starting points in $T$ on the CPU.
8 Stop
9 **end**
10 **end**

**Output**: $(x_1, \ldots, x_n)$ with $\left\| \sum_{i=1}^{n} x_i \mathbf{b}_i \right\| = \lambda_1(\mathcal{L})$

The top enumeration will stop automatically if a sufficient number of vectors from the top of the tree have been enumerated. The rest of the top of the tree is enumerated in the following iterations of the algorithm.

Line 4 performs the actual GPU enumeration. In each iteration, a set of starting vectors and starting levels $\{\mathbf{x}_k, L_k\}$ is uploaded to the GPU. These starting vectors can be either vectors generated by the top enumeration in the region between $\alpha$ and $n$ (in which case $L_k = \alpha$) or the vectors (and levels) written back by the GPU because of early termination, so that the enumeration will continue. In total NUMSTARTPOINTS vectors (a mix of new and old vectors) are uploaded at each iteration. For each starting vector $\mathbf{x}_k$ (with associated starting level $L_k$) the GPU outputs a vector

$$\bar{\mathbf{x}}_k = [\bar{x}_{k,1}, \ldots, \bar{x}_{k,\alpha-1}, x_{k,\alpha}, \ldots, x_{k,n}] \qquad \text{for } k = 1 \ldots \text{NUMSTARTPOINTS}$$

(which describes the current position in the search tree), the current level $L_k$, the number of enumeration steps $s_k$ performed and also part of the internal state of the enumeration. This state $\{\bar{\mathbf{x}}_k, \mathbf{\Delta x}_k, L_k\}$ can be used to continue the enumeration later on. The vectors $\mathbf{\Delta x}_k$ are used in the enumeration to generate the zig-zag pattern and are part of the internal state of the enumeration [SE91]. This state is added to the output to be able to efficiently restart the enumeration at the point it was terminated.

Line 5 will select the resulting vectors from the GPU enumeration that were terminated early. These will be added to the set $T$ of *leftover* vectors, which will be relaunched in the next iteration of the algorithm. If the set of leftover vectors is too small to get an efficient GPU enumeration, the CPU takes over and finishes off the last part of the enumeration. This final part only takes limited time.

*GPU Threads and Load Balancing.* In Section 3.2 the need for a load balancing algorithm was introduced: all threads should remain active and to ensure this, each thread in the same warp should run the same instruction. One of the

problems in achieving this, is the length difference of each subtree enumeration. Some very long subtree enumeration can cause all the other threads in the warp to become idle after they finish their subtree enumeration.

Therefore the number of enumeration steps that each thread can perform on a subtree is limited by **M**. When **M** is exceeded, a subtree enumeration is forced to stop. After this, all threads in the same warp will reinitialise: they will either continue the previous subtree enumeration (that was terminated by reaching **M**) or they will pick a new starting vector of the list $S \cup T$ delivered by the CPU. Then the enumeration starts again, limited to **M** enumeration steps.

In our experiments, NUMSTARTPOINTS was around 20-30 times higher than NUMTHREADS, which means that on average every GPU thread enumerated 20-30 subtrees in each iteration. **M** was chosen to be around 50-200.

## 4 Experimental Results

In this section we present some results of the CUDA implementation of our algorithm. For comparison we used the highly optimized ENUM algorithm of the `fpLLL` library in version 3.0.11 from [CPS]. `NTL` does not allow to run ENUM as a standalone SVP solver, but [Puj08] and the ENUM timings of [GN08b] show that `fpLLL`'s ENUM runs faster than `NTL`'s (the bit size of the lattice bases used in [GN08b] is higher than what we used, therefore a comparison with those timings is to be drawn carefully).

The CUDA program was compiled using `nvcc`, for the CPU programs we used `g++` with compiler flag `-O2`. The tests were run on an Intel Core2 Extreme CPU X9650 (using one single core) running at 3 GHz, and an NVIDIA GTX 280 graphics card. We run up to 100000 threads in parallel on the GPU. The code of our program can be found online.[1]

We chose random lattices following the construction principle of [GM03] with bit size of the entries of $10 \cdot n$. This type of lattices was also used in [GN08b] and [NS06]. We start with the basis in Hermite normal form and LLL-reduce them with $\delta = 0.99$. At the end of this section, we present some timings using BKZ-20 reduced bases, to show the capabilities of stronger pre-reduction.

Both algorithms, the enum of `fpLLL` (run with parameter *-a svp*) and our CUDA version, always output the same coefficient vectors and therefore a lattice vector with shortest possible length. We compare now the throughput of GPU and CPU concerning enumerations steps. Section 3.1 gives the explanation what is computed in each enumeration step. On the GPU, up to 200 million enumeration steps per second can be computed, while similar experiments on CPU only yielded 25 million steps per second. We choose $\alpha = n - 11$ for our experiments, this shapes up to be a good choice in practice. Table 1 and Figure 2 illustrate the experimental results. The figure shows the runtimes of both algorithms when applied to five different lattices of each dimension. One can notice that in dimension above 44, our CUDA implementation always outperforms the `fpLLL` implementation.

[1] `http://homes.esat.kuleuven.be/~jhermans/gpuenum/index.html`

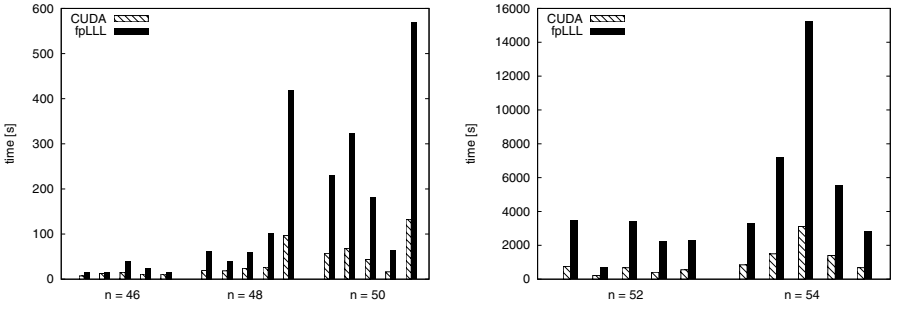



**Fig. 2.** Timings for enumeration. The graph shows the time needed for enumerating five different random lattices in each dimension $n$. It compares the ENUM algorithm of the `fpLLL`-library with our parallel CUDA version.

**Table 1.** Average time needed for enumeration of LLL pre-reduced lattices in each dimension n. The table presents the percentage of time that the GPU version takes compared to the `fpLLL` version.

| n | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 60 |
|---|---|---|---|---|---|---|---|---|---|---|
| `fpLLL` - ENUM | 0.96s | 2.41s | 17.7s | 22.0s | 136s | 273s | 2434s | 6821s | 137489s | - |
| CUDA - ENUM | 2.75s | 4.29s | 11.7s | 11.4s | 37.0s | 63.5s | 520s | 1504s | 30752s | 274268s |
| | 286% | 178% | 66% | 52% | 27% | 23% | 21% | 22% | 22% | - |

Table 1 shows the average value over all five lattices in each dimension. Again one notices that the GPU algorithm demonstrates its strength in dimensions above 44, where the time goes down to 22% in dimensions 54 and 56 and down to 21% in dimension 52. Therefore we state that the GPU algorithm gains big speedups in dimensions higher than 45, which are the interesting ones in practice. In dimension 60, `fpLLL` did not finish the experiments in time, therefore only the average time of the CUDA version is presented in the table.

Table 2 presents the timing of the same bases, pre-reduced using BKZ algorithm with blocksize 20. The time of the BKZ-20 reduction is not included in the timings shown in the table. For dimension 64 we changed $\alpha$ (the subtree dimension) from the usual $n - 11$ to $\alpha = n - 14$, as this leads to lower timings in high dimensions. First, one can notice that both algorithms run much faster

**Table 2.** Average time needed for enumeration of BKZ-20 pre-reduced lattices in each dimension n. The time for pre-reduction is omitted in both cases.

| n | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 |
|---|---|---|---|---|---|---|---|---|---|
| `fpLLL` - ENUM | 2.96s | 7.30s | 36.5s | 79.2s | 190s | 601s | 1293s | 7395s | 15069s |
| CUDA - ENUM | 3.88s | 5.42s | 16.9s | 27.3s | 56.8s | 119s | 336s | 986s | 4884s |
| | 131% | 74% | 46% | 34% | 30% | 20% | 26% | 13% | 32% |

when using stronger pre-processing, a fact that was already mentioned in [HS07]. Second, we see that the speedup of the GPU version goes down to 13% in the best case (dimension 62).

As pruning would speed up both the serial and the parallel enumeration, we expect the same speedups with pruning.

It is hard to give an estimate of the achieved speedup compared to the number of threads used: since GPUs have hardware-based scheduling, it is not possible to know the number of active threads exactly. Other properties, like memory access and divergent warps, have a much greater influence on the performance and cannot be measured in thread counts or similar figures. When comparing only the number of double fmadds, the GTX 280 should be able to do 13 times more fmadd's than a single Core2 Extreme X9650.[2] Based on our results we fill only 30 to 40% of the GPUs ALUs. Using the CUDA Profiler, we determine that in our experiments around 12% of branches was divergent, which implies a loss of parallelism and also some ALUs being left idle. There is also a high number of warp serializations due to conflicting shared and constant memory access. The ratio warp serializations/instructions is around 35%.

To compare CPUs and GPUs, we can have a look at the cost of both platforms in dollardays, similar to the comparison in [BCC+09]. We assume a cost of around $2200 for our CPU (quad core) + 2x GTX295 setup. For a CPU-only system, the cost is only around $900. Given a speedup of 5 for a GPU compared to a CPU, we get a total speedup of 24 (4 CPU cores + 4 GPUs) in the $2200 machines and only a speedup of 4 in the CPU-only machine, assuming we can use all cores. This gives $225 \cdot t$ dollardays for the CPU-only system and only $91 \cdot t$ dollardays for the CPU+GPU system, where $t$ is the time. This shows that even in this model of expense, the GPU implementation gains an advantage of around 2.4.

## 5 Further Work

Further improvements are possible using multiple CPU cores. Our implementation only uses one CPU core for the top enumeration and the rest of the outer loop of the enumeration. During the subtree enumerations on the GPU, the main part of the algorithm, the CPU is not used. When the GPU starts a batch of subtree enumerations it would be possible to start threads on the CPU cores as well. We expect a speedup of two compared to our actual implementation using this idea.

It is possible to start enumeration using a shorter starting value than the first basis vectors norm. The Gaussian heuristic can be used to predict the norm of the shortest basis vector $\lambda_1$. This can lead to enormous speedups in the algorithm. We did not include this improvement into our algorithm so far to get comparable results to `fpLLL`.

[2] A GTX280 can do 30 double fmadds in a 1.3GHz cycle, a single Core2 core can do 2 double fmadds in every two 3GHz cycle, which gives us a speedup of 13 for the GTX280.

## Acknowledgments

We thank the anonymous referees for their valuable comments. We thank Özgür Dagdelen for creating some of the initial ideas of parallelizing lattice enumeration and Benjamin Milde, Chen-Mou Cheng, and Bo-Yin Yang for the nice discussions and helpful ideas.

We would like to thank ECRYPT II[3] and CASED[4] for providing the funding for the visits during which this work was prepared. Part of the work was done during the authors' visit to Center for Information and Electronics Technologies, National Taiwan University.

## References

[AD97] Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: Proceedings of the Annual Symposium on the Theory of Computing — STOC 1997, pp. 284–293 (1997)

[AKS01] Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the Annual Symposium on the Theory of Computing — STOC 2001, pp. 601–610. ACM Press, New York (2001)

[AMD06] Advanced Micro Devices. ATI CTM Guide. Technical report (2006)

[BCC+09] Bernstein, D.J., Chen, T.-R., Cheng, C.-M., Lange, T., Yang, B.-Y.: ECM on graphics cards. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 483–501. Springer, Heidelberg (2009)

[BL06] Buchmann, J., Ludwig, C.: Practical lattice basis sampling reduction. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 222–237. Springer, Heidelberg (2006)

[BW09] Backes, W., Wetzel, S.: Parallel lattice basis reduction using a multi-threaded Schnorr-Euchner LLL algorithm. In: Sips, H., Epema, D., Lin, H.-X. (eds.) Euro-Par 2009 Parallel Processing. LNCS, vol. 5704, pp. 960–973. Springer, Heidelberg (2009)

[CIKL05] Cook, D.L., Ioannidis, J., Keromytis, A.D., Luck, J.: Cryptographics: Secret key cryptography using graphics cards. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 334–350. Springer, Heidelberg (2005)

[CNS99] Coupé, C., Nguyen, P.Q., Stern, J.: The effectiveness of lattice attacks against low exponent RSA. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 204–218. Springer, Heidelberg (1999)

[CPS] Cadé, D., Pujol, X., Stehlé, D.: fpLLL - a floating point LLL implementation. Available at Damien Stehlé's homepage at école normale supérieure de Lyon, http://perso.ens-lyon.fr/damien.stehle/english.html

[Dag09] Dagdelen, Ö.: Parallelisierung von Gitterbasisreduktionen. Masters thesis, TU Darmstadt (2009)

[Din02] Dinur, I.: Approximating $\mathrm{SVP}_\infty$ to within almost-polynomial factors is NP-hard. Theoretical Computer Science 285(1), 55–71 (2002)

[3] http://www.ecrypt.eu.org/

[4] http://www.cased.de

[DN00] Durfee, G., Nguyen, P.Q.: Cryptanalysis of the RSA schemes with short secret exponent from Asiacrypt 1999. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 14–29. Springer, Heidelberg (2000)

[Fle07] Fleissner, S.: GPU-Accelerated Montgomery Exponentiation. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4487, pp. 213–220. Springer, Heidelberg (2007)

[FP83] Fincke, U., Pohst, M.: A procedure for determining algebraic integers of given norm. In: van Hulzen, J.A. (ed.) ISSAC 1983 and EUROCAL 1983. LNCS, vol. 162, pp. 194–202. Springer, Heidelberg (1983)

[GGH97] Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (1997)

[GM03] Goldstein, D., Mayer, A.: On the equidistribution of hecke points. Forum Mathematicum 2003 15(2), 165–189 (2003)

[GN08a] Gama, N., Nguyen, P.Q.: Finding short lattice vectors within Mordell's inequality. In: Proceedings of the Annual Symposium on the Theory of Computing — STOC 2008, pp. 207–216. ACM Press, New York (2008)

[GN08b] Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)

[GPV08] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the Annual Symposium on the Theory of Computing — STOC 2008, pp. 197–206. ACM Press, New York (2008)

[HPS98] Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)

[HS07] Hanrot, G., Stehlé, D.: Improved analysis of kannan's shortest lattice vector algorithm. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 170–186. Springer, Heidelberg (2007)

[HT93] Heckler, C., Thiele, L.: A parallel lattice basis reduction for mesh-connected processor arrays and parallel complexity. In: IEEE Symposium on Parallel and Distributed Processing — SPDP, pp. 400–407. IEEE Computer Society Press, Los Alamitos (1993)

[HT98] Heckler, C., Thiele, L.: Complexity analysis of a parallel lattice basis reduction algorithm. SIAM J. Comput. 27(5), 1295–1302 (1998)

[HW07] Harrison, O., Waldron, J.: AES Encryption Implementation and Analysis on Commodity Graphics Processing Units. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 209–226. Springer, Heidelberg (2007)

[Jou93] Joux, A.: A fast parallel lattice reduction algorithm. In: Proceedings of the Second Gauss Symposium, pp. 1–15 (1993)

[Kan83] Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proceedings of the Annual Symposium on the Theory of Computing — STOC 1983, pp. 193–206. ACM Press, New York (1983)

[Kho05] Khot, S.: Hardness of approximating the shortest vector problem in lattices. J. ACM 52(5), 789–808 (2005)

[Koy04] Koy, H.: Primale-duale Segment-Reduktion (2004), http://www.mi.informatik.uni-frankfurt.de/research/papers.html

[Len83] Lenstra, H.W.: Integer programming with a fixed number of variables. Math. Oper. Res. 8, 538–548 (1983)

[LLL82] Lenstra, A., Lenstra, H., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261(4), 515–534 (1982)

[LM08] Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 37–54. Springer, Heidelberg (2008)

[LMPR08] Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: Swifft: A modest proposal for fft hashing. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 54–72. Springer, Heidelberg (2008)

[LO85] Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. Journal of the ACM 32(1), 229–246 (1985)

[Lyu09] Lyubashevsky, V.: Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (2009)

[Man07] Manavski, S.A.: Cuda Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography. In: IEEE International Conference on Signal Processing and Communications — ICSPC, pp. 65–68. IEEE Computer Society Press, Los Alamitos (2007)

[May10] May, A.: Using LLL-reduction for solving RSA and factorization problems. In: Nguyen, P.Q., Vallée, B. (eds.) The LLL algorithm, pp. 315–348. Springer, Heidelberg (2010)

[MG02] Micciancio, D., Goldwasser, S.: Complexity of Lattice Problems: a cryptographic perspective. The Kluwer International Series in Engineering and Computer Science, vol. 671. Kluwer Academic Publishers, Boston (2002)

[MPS07] Moss, A., Page, D., Smart, N.P.: Toward Acceleration of RSA Using 3D Graphics Hardware. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 364–383. Springer, Heidelberg (2007)

[MV10] Micciancio, D., Voulgaris, P.: Faster exponential time algorithms for the shortest vector problem. In: Proceedings of the Annual Symposium on Discrete Algorithms — SODA 2010 (2010)

[NS05] Nguyen, P.Q., Stehlé, D.: Floating-point LLL revisited. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 215–233. Springer, Heidelberg (2005)

[NS06] Nguyen, P.Q., Stehlé, D.: LLL on the average. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 238–256. Springer, Heidelberg (2006)

[NV08] Nguyen, P.Q., Vidick, T.: Sieve algorithms for the shortest vector problem are practical. J. of Mathematical Cryptology 2(2) (2008)

[Nvi07a] Nvidia. Compute Unified Device Architecture Programming Guide. Technical report (2007)

[NVI07b] NVIDIA. CUBLAS Library (2007)

[otCC09] 1363 Working Group of the C/MM Committee. IEEE P1363.1 Standard Specification for Public-Key Cryptographic Techniques Based on Hard Problems over Lattices (2009), http://grouper.ieee.org/groups/1363/

[Pei09a] Peikert, C.: Bonsai trees (or, arboriculture in lattice-based cryptography). Cryptology ePrint Archive, Report 2009/359 (2009), http://eprint.iacr.org/

[Pei09b] Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Proceedings of the Annual Symposium on the Theory of Computing — STOC 2009, pp. 333–342 (2009)

[PS08] Pujol, X., Stehlé, D.: Rigorous and efficient short lattice vectors enumeration. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 390–405. Springer, Heidelberg (2008)

[Puj08] Pujol, X.: Recherche efficace de vecteur court dans un réseau euclidien. Masters thesis, ENS Lyon (2008)

[RR06] Regev, O., Rosen, R.: Lattice problems and norm embeddings. In: Proceedings of the Annual Symposium on the Theory of Computing — STOC 2006, pp. 447–456. ACM Press, New York (2006)

[RV92] Roch, J.-L., Villard, G.: Parallel gcd and lattice basis reduction. In: Bougé, L., Robert, Y., Trystram, D., Cosnard, M. (eds.) CONPAR 1992 and VAPP 1992. LNCS, vol. 634, pp. 557–564. Springer, Heidelberg (1992)

[Sch91] Schnorr, C.-P.: Factoring integers and computing discrete logarithms via diophantine approximations. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 281–293. Springer, Heidelberg (1991)

[Sch03] Schnorr, C.-P.: Lattice reduction by random sampling and birthday methods. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 145–156. Springer, Heidelberg (2003)

[SE91] Schnorr, C.-P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In: Budach, L. (ed.) FCT 1991. LNCS, vol. 529, pp. 68–85. Springer, Heidelberg (1991)

[SG08] Szerwinski, R., Guneysu, T.: Exploiting the Power of GPUs for Asymmetric Cryptography. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 79–99. Springer, Heidelberg (2008)

[SH95] Schnorr, C.-P., Hörner, H.H.: Attacking the Chor-Rivest cryptosystem by improved lattice reduction. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 1–12. Springer, Heidelberg (1995)

[Sho] Shoup, V.: Number theory library (NTL) for C++, `http://www.shoup.net/ntl/`

[SSTX09] Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 617–635. Springer, Heidelberg (2009)

[Vil92] Villard, G.: Parallel lattice basis reduction. In: International Symposium on Symbolic and Algebraic Computation — ISSAC, pp. 269–277. ACM Press, New York (1992)

[Wet98] Wetzel, S.: An efficient parallel block-reduction algorithm. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 323–337. Springer, Heidelberg (1998)

# Flexible Partial Enlargement to Accelerate Gröbner Basis Computation over $\mathbb{F}_2$

Johannes Buchmann[1], Daniel Cabarcas[2], Jintai Ding[3,⋆], and Mohamed Saied Emam Mohamed[1]

[1] TU Darmstadt, FB Informatik
Hochschulstrasse 10, 64289 Darmstadt, Germany
{mohamed,buchmann}@cdc.informatik.tu-darmstadt.de
[2] Department of Mathematical Sciences, University of Cincinnati
cabarcd@mail.uc.edu
[3] Department of Mathematical Sciences, University of Cincinnati, South China University of Technology
jintai.ding@uc.edu

**Abstract.** Recent developments in multivariate polynomial solving algorithms have made algebraic cryptanalysis a plausible threat to many cryptosystems. However, theoretical complexity estimates have shown this kind of attack unfeasible for most realistic applications. In this paper we present a strategy for computing Gröbner basis that challenges those complexity estimates. It uses a flexible partial enlargement technique together with reduced row echelon forms to generate lower degree elements–mutants. This new strategy surpasses old boundaries and obligates us to think of new paradigms for estimating complexity of Gröbner basis computation. The new proposed algorithm computed a Gröbner basis of a degree 2 random system with 32 variables and 32 equations using 30 GB which was never done before by any known Gröbner bases solver.

**Keywords:** Algebraic cryptanalysis, Gröbner basis, Complexity, HFE.

## 1 Introduction

The standard way to represent the polynomial ideal is to compute a Gröbner basis of it. Solving a system of multivariate polynomial equations is one of the most important application of Gröbner bases. The complexity of algorithms for computing a Gröbner basis of an ideal depends on how large the subset of elements of the ideal used during the computation is. Minimizing the size of this subset is an extremely important target for research in this area.

Suppose $P$ is a finite set of degree 2 polynomials. For $d \geq 2$ consider the set

$$H_d := \{t \cdot p \mid t \text{ is a term}, p \in P, \deg(tp) \leq d\} \tag{1}$$

It is well known that for $d$ large enough, a row echelon form of $H_d$ is a Gröbner basis for $\langle P \rangle$ (see Section 1.1 for a definition of row echelon form and other

⋆ Corresponding author.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 69–81, 2010.

notations used in this introduction). However, systems that appear in algebraic cryptanalysis and their corresponding degrees needed to solve them are so large, that constructing $H_d$ is unfeasible. Usually memory is the main constraint. A key strategy we have been and are following is to explore and do computation in $H_d$ for large enough $d$, but without storing the whole set $H_d$ at any given time.

A first breakthrough came with the concept of mutant polynomials [4,5]. Informally, a polynomial $p$ is called mutant, if $p \in \text{span}(H_d) \setminus \text{span}(H_{d-1})$ but its degree is strictly less than $d$. If $x$ is a variable, then the polynomial $xp$ belongs to $\text{span}(H_{d+1})$, however, since its degree is less than $d+1$ we can reduce it using only elements from $H_d$.

A second breakthrough came with the partial enlargement technique [9]. By partitioning the set $X \times H_d$ according to leading variables, it is possible to explore $H_{d+1}$ one step at a time without being forced to store the whole set at once. In [8] the $\text{MXL}_3$ algorithm was introduced which uses an optimized mutant strategy, the partial enlargement technique and a mutant based termination criterion.

In this paper we introduce an efficient algorithm, called Mutant-based Gröbner Basis algorithm (MGB), that uses a more flexible partial enlargement to omit some parts of $H_d$ and still satisfies $\text{MXL}_3$'s criterion. Similar heuristics were attempted by Gotaishi and Tsujii in [7]. The MGB algorithm is fully implemented in C++. We give experimental results that compare the performance of MGB with both $\text{MXL}_3$ and the Magma's implementation of $\text{F}_4$ algorithm [6]. Our experiments are based on randomly generated MQ systems and HFE cryptosystems. We show that MGB computed a Gröbner basis of a degree 2 random system with 32 variables and equations in 2.3 days using only 30 Gigabytes.

This paper is organized as follows. In Section 2 we discuss the theoretical foundation of the new proposed algorithm. In Section 3 we present the MGB algorithm and exemplify its operation in Section 4. A complexity analysis of the algorithm is discussed in Section 5 and experimental results are presented in Section 6. Finally, we conclude the paper and give the future work in Section 7. Before continuing let us introduce the necessary notation.

### 1.1 Notation

Let $X := \{x_1, \ldots, x_n\}$ be a set of variables, upon which we impose the following order: $x_1 > x_2 > \ldots > x_n$. (Note the counterintuitive $i < j$ imply $x_i > x_j$.) Let

$$R = \mathbb{F}_2[x_1, \ldots, x_n]/\langle x_1^2 - x_1, ..., x_n^2 - x_n \rangle$$

be the Boolean polynomial ring in $X$ with the terms of $R$ ordered by the graded lexicographical order $<_{glex}$. We represent an element of $R$ by its minimal representative polynomial over $\mathbb{F}_2$ where degree of each term w.r.t any variable is 0 or 1. We denote by $T_d(x_{j_1}, \ldots, x_{j_s})$ the set of terms of degree $d$ in the variables $x_{j_1}, \ldots, x_{j_s}$, and by $T_d$ all the terms of degree $d$.

Let $P = \{p_1, \ldots, p_m\}$ be set of polynomials in $R$. A row echelon form is simply a basis for $\text{span}(P)$ with pairwise distinct head terms, (see [8] for definition). We will denote by $P_{(op)d}$ the subset of all the polynomials of degree $(op)d$ in $P$, where

$(op)$ is any of $\{=, <, >, \leq, \geq\}$. We define the leading variable of $p \in P$, denoted by $\mathrm{LV}(p)$, as the largest variable in $\mathrm{HT}(p)$ according to the order defined on the variables set.

Suppose that all polynomials in $P$ are of degree $d$. We define $P_{x_j}$ as the set of all polynomials in $P$ with leading variable $x_j$, so that $P$ is partitioned in $n$ sets $P_{x_1}, \ldots, P_{x_n}$. We refer to these sets as *variable-partitions*. Given a pair $(x, p)$ in the Cartesian product $X \times P$, we define its leading variable to be $\mathrm{LV}(x, p) := \max(x, \mathrm{LV}(p))$ (note that in general $\mathrm{LV}(x, p) \neq \mathrm{LV}(xp)$). Consider also the partition of the Cartesian product $X \times P$ in $n$ sets, $L_1(P), \ldots, L_n(P)$ defined by

$$L_j(P) := \{(x, p) \in X \times P \mid \mathrm{LV}(x, p) = x_j\} , \tag{2}$$

and note that that for $(x, p) \in L_j(P)$, $\mathrm{LV}(xp) \leq x_j$ or $\deg(xp) < d + 1$.

## 2 A More Flexible Partial Enlargement

The partial enlargement technique introduced first in [9] is effective in reducing the number of polynomials needed at the highest degree $D$ where mutants start to appear. However, it offers no advantage over generating the whole set $H_d$ for $d < D$ (as defined in equation (1)). We propose a method for systematically omitting subsets of $H_d$ whenever a given condition is met. In this section we provide the theoretical foundation for this method.

The partial enlargement technique introduced in [9] can be described as follows. Let $P_2$ be a finite set of degree 2 polynomials in $R$ and assume $P_2$ is in row echelon form. Initialize the set $G$ with $P_2$. Then, enlarge $P_2$ one variable at a time, i.e., for $j = n, n-1, \ldots, 1$ (in that order), construct the set $L_j(P_2)$, then append the set $\{xp \mid (x, p) \in L_j(P_2)\}$ to $G$ and compute a row echelon form of $G$. If no mutants are found in $G$, make $P_3$ the set of new polynomials in $G$ and repeat the process for $P_3$ only.

While no mutants are produced, we produce a sequence of sets $P_2, P_3, \ldots$ such that for $p \in P_d$, $\deg(p) = d$ and $\mathrm{span}(P_d) = \mathrm{span}(H_d)$. We have observed, that often, the largest variable-partitions of each $P_d$ are *full*, meaning that they have as many linearly independent polynomials as they can possibly have. So, there exist $J_d < n$ such that for all term $t \in T_d$ such that $\mathrm{LV}(t) \geq x_{J_d}$, there exist $p \in P_d$ such that $t = \mathrm{HT}(p)$. Note that if the $j$ partition of $P_d$ is full, necessarily the $j$ partition for $P_{d+1}$ is also full. Hence, the sequence $J_2, J_3, \ldots$ is non-decreasing.

If the $j$ partition of $P_d$ is full, we propose to omit the $j$ partition from any subsequent $P_D$ $(D > d)$, under the certainty that it will be full. The problem is that when mutants appear, it is possible that the head term of a mutant happens to be precisely one of the omitted head terms, thus failing to be completely reduced, in the sense that its head term is already in the ideal generated by the head terms of the elements in $G$.

To remedy this situation, we compute row reduced echelon forms instead of just any row echelon form, by means of which, we are able to perform reductions

in advance, and ensure polynomials are fully reduced with respect to the omitted partitions. The following proposition states this result precisely.

**Proposition 1.** *Let $P_d$ be a finite subset of degree $d$ polynomials from $K[\underline{x}]$ in row-echelon form. Suppose there exist $J < n$ such that for all term $t \in T_d$ such that $\mathrm{LV}(t) \geq x_J$, there exist $p \in P_d$ such that $t = \mathrm{HT}(p)$. Let*

$$G := P_d \cup \bigcup_{j=J+1}^{n} \{xp \mid (x,p) \in L_j(P_d)\} ,$$

*$\widetilde{G}$ the row reduced echelon form of $G$ and*

$$P_{d+1} := \{p \in \widetilde{G} \mid \deg(p) = d+1\} .$$

*Then for $j > J$, $(x,p) \in L_j(P_{d+1})$ and any term $s \in T_{d+1}$ such that $\mathrm{LV}(s) \geq x_J$, the term $s$ is not a term in $xp$.*

*Proof.* Let $j > J$, $(x,p) \in L_j(P_{d+1})$ and $s \in T_{d+1}$ such that $\mathrm{LV}(s) \geq x_J$. From the definitions of $G$ and $P_{d+1}$, all terms of degree $d+1$ in $p$ belong to $T_{d+1}(x_{J+1}, \ldots, x_n)$ thus $s$ is not a term in $p$.

Since $P_{d+1}$ is a subset of $\widetilde{G}$, the row reduced echelon form of $G$, and all terms of degree $d$ with leading variables $\geq x_J$ appear as head term of an element of $P_d \subset G$, then such terms do not appear in $p$, i.e. all terms of degree $d$ in $p$ are elements of $T_d(x_{J+1}, \ldots, x_n)$. Moreover, $x < x_J$ since $j > J$, hence, there is no term $t$ in $p$ with degree $d$ that satisfies $xt = s$. Therefore, $s$ is not a term in $xp$.

The importance of this proposition is that the polynomials of degree $d+1$ with leading variables $x_1, \ldots, x_J$ are not needed to reduce polynomials coming from $L_j(P_{d+1})$ for $j > J$. Hence, when enlarging $P$ we may omit the polynomials coming from $L_1(P), \ldots, L_J(P)$. This does not imply that this polynomials are not needed ever again, but just that their computation can be postponed yet obtaining sets of polynomials row reduced with respect to this missing polynomials. This observation leads to the heuristic strategy described in the following section. Section 4 illustrates the operation of the algorithm.

## 3 The MGB Algorithm

In this section we introduce the MGB algorithm to compute Gröbner bases over the function ring $R$ under the graded lexicographic order. The MGB is based on the MXL3 algorithm [8], and differs in a heuristic method to omit some variable-partitions based on the flexible partial enlargement explained in Section 2. The heuristic consists in enlarging up to the first full partition. By this way many partitions are omitted, yet enough polynomials are enlarged.

Algorithms 1, 2, 3 and 4 provide a detailed description of the MGB algorithm. The most important difference with $\mathrm{MXL}_3$ is the extra condition

$$|P_{=D-1}(x_1, \ldots, x)| = |T_{=D-1}(x_1, \ldots, x)|$$

in line 20 of the *Enlarge* subroutine (Algorithm 4). It means that all the terms of degree $D-1$ with leading variable $\in x_1, \ldots, x$ appear as leading terms in $P$. When the condition is met, the flag *newExtend* is set to true, which forces $D$ to be increased in the next call to *Enlarge*.

Throughout the operation of the algorithm a degree bound $D$ is used. This degree bound denotes the maximum degree of the polynomials contained in $P$. An elimination degree bound $ED$ is used as a bound for the degrees of polynomials in $P$ that are being eliminated. RREF$(P, ED)$ means the reduced row echelon form of $P_{\leq ED}$. We mean by the new elements the set of all polynomials produced during the previous enlargement. The set $M$ stores the mutants obtained during the *Echelonization* process. We define the array $S$ to keep the the largest leading variable at each degree level. Note that the content of $P$ is changed throughout the operation of the algorithm.

**Algorithm 1.** Algorithm1$(P)$

**Require:** $P$ is a finite sequence from $R$
1: $D = \max\{\deg(p) \mid p \in P\}$
2: $ED = D$
3: $M = \emptyset$
4: $S = \{s_i = x_1 : 1 \leq i \leq D\}$
5: $x = x_1$
6: $newExtend = true$
7: **loop**
8: *Echelonize*$(P, M, ED)$
9: $G =$*Gröbner*$(P, M, D, ED)$
10: **if** $G \neq \emptyset$ **then**
11: **return** $G$
12: **end if**
13: *Enlarge*$(P, M, S, x, D, ED, newExtend)$
14: **end loop**

**Algorithm 2.** Echelonize$(P, M, ED)$

1: Consider each term in $P$ as a new variable
2: $P =$ RREF$(P, ED)$
3: $M = \{p \in P : P$ is a new element and $\deg(p) < ED\}$

On an actual implementation we do not compute the full row reduced echelon form of the constructed matrix in each step. For the columns corresponding to highest degree terms we only clear entries under the pivot. For the rest of the columns we clear above and below the pivot. The idea of using the full reduction in this case is creating the vertical hole of the unextended partitions as explained in the previous section and illustrated in the next section.

**Algorithm 3.** Gröbner$(P, M, S, D, ED)$

1: $G = \emptyset$
2: **if** $M_{<ED} = \emptyset$ **and** $(ED < D$ **or** $newExtend = true)$ **then**
3: $s = S[ED - 1]$
4: **if** $|P_{=ED-1}| = |T_{=ED-1}(s, \ldots, x_n)|$ **then**
5: **if** $s < x_1$ **then**
6: Recover$(P, ED, S)$
7: **end if**
8: $G = P_{<ED}$
9: **end if**
10: **end if**
11: **return** $G$

**Algorithm 4.** Enlarge$(P, M, S, x, D, ED, newExtend)$

1: **if** $M \neq \emptyset$ **then**
2: $k = \min\{\deg(p)\colon p \in M\}$
3: Select a necessary number of mutants $NM$ from $M_{=k}$
4: $y = \max\{\mathrm{LV}(p) : p \in NM\}$
5: Multiply selected mutants by all variables $\leq y$
6: Remove the selected mutants from $M$
7: Add the new polynomials to $P$
8: $ED = k + 1$
9: **else**
10: **if** $newExtend$ **then**
11: $D = D + 1$
12: $x = \min\{\mathrm{LV}(p) : p \in P_{=D-1}\}$
13: $newExtend = false$
14: **else**
15: $x = \min\{\mathrm{LV}(p) : p \in P_{=D-1}$ and $\mathrm{LV}(p) > x\}$
16: **end if**
17: $S[D] = x$
18: Multiply all $p \in P_{D-1}$ that has leading variable $x$ by all the variables $\leq x$ without redundancy
19: Add the newly obtained polynomials to $P$
20: **if** $|P_{=D-1}(x_1, \ldots, x)| = |T_{=D-1}(x_1, \ldots, x)|$ **or** $(x = x_1)$ **then**
21: $newExtend = true$
22: **end if**
23: Set $ED = D$
24: **end if**

The *Recover* function in line 6 of Algorithm 3 enlarges all partitions of $P_{\leq Ed}$ that were omitting during the enlargement process and echelonizes $P$. Also, it multiplies any mutants found.

The correctness of the MGB algorithm is guaranteed by the following proposition, first stated and proved in [8].

**Proposition 2.** *Let $G$ be a finite subset of $R$ with $D$ being the highest degree of its elements. $G$ is a Gröbner basis if it satisfies the following conditions:*

1. *$G$ contains all the terms of degree $D$ as leading terms; and*
2. *if $H := G \cup \{t \cdot g \mid g \in G, t$ a term and $\deg(t \cdot g) \leq D + 1\}$, there exists $\widetilde{H}$, a row echelon form of $H$, such that $\widetilde{H}_{\leq D} = G$,*

The MGB algorithm terminates only when it returns the set $G = P_{<ED}$ that is computed by the *Gröbner* subroutine. We are going now to prove that $G$ is a Gröbner basis of the ideal generated by $P$.

The first if statement of Algorithm 3, line 2, guarantees the second condition of Proposition 2 since all the polynomials up to degree $ED - 1$ are extended one degree more without producing any mutants ($M_{<ED} = \emptyset$). The second if statement of Algorithm 3, line 4, represents a second difference from $\text{MXL}_3$. It means that all the terms of leading variable $\in s, \ldots, x_n$ appear as leading terms in $P_{=Ed-1}$. This will guarantee the first condition of Proposition 2 as follows.

In case of $s = x_1$, $P_{<ED}$ contains all terms of degree $ED-1$ as leading terms. In case of $s < x_1$, MGB needs to recover $P_{\leq ED}$ as explained above. This leads to satisfying the two conditions of Proposition 2 since these unextended partitions have full rank.

Therefore MGB returns the set $G = P_{<ED}$ that satisfies the two conditions of Proposition 2. Then it is a Gröbner basis. The worst case of MGB is to reproduce the $\text{MXL}_3$ algorithm. So MGB terminates since $\text{MXL}_3$ terminates, theorem 1 in [8]. As an important note, for the experiments run so far, the recovering process was never necessary.

## 4 The Algorithm in Action

We describe the behavior of the algorithm in a concrete example, a sequence of 24 degree 2 polynomials in 24 variables. We refer to Figure 1 for a schematic representation of the process.

After *Echelonization*, the leading variables of the original polynomials range from $x_1$ to $x_2$ as depicted at the bottom of Figure 1. Then, the $x_1$ and $x_2$ partitions are enlarged and echelonized to obtain degree 3 polynomials with leading variables ranging from $x_1$ to $x_3$. Here we encounter that the variable partitions for $x_1$ and $x_2$ are full and we represent it with the darker shading in Figure 1.

So in the next step, only the $x_2$ and $x_3$ partitions are enlarged to degree 4. After *Echelonization* we obtain polynomials of degree 4 with leading variables ranging from $x_2$ to $x_5$. Note that since the two partitions $x_1$ and $x_2$ of degree 3 polynomials are full, no term with leading variable $x_1$ or $x_2$ of degree 3 appears in the degree 4 polynomials. This is depicted in Figure 1 with vertical stripes.

At degree 4, the variable partitions corresponding to $x_2$, $x_3$ and $x_4$ are full, so only the $x_4$ and $x_5$ partitions are enlarged to degree 5. After *Echelonization*, we obtain polynomials of degree 5 with leading variables ranging from $x_4$ to $x_{12}$. Note that by Proposition 1, no term with leading variable $x_1$ of degree 4 appears in the degree 5 polynomials, and *Echelonization* clears $x_2$ through $x_4$.

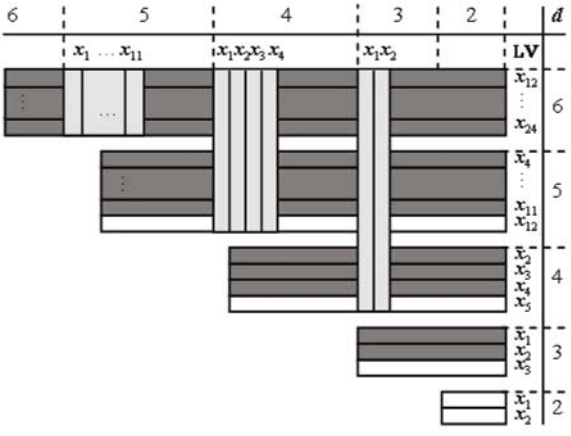



**Fig. 1.** Behavior of the algorithm for a sequence of 24 degree 2 equations in 24 variables. Horizontal stripes represent variable-partitions, darker ones are full. Vertical stripes represent terms that do not appear in the given polynomials.

At degree 5, the variable partitions from $x_4$ to $x_{11}$ are full, so only the $x_{11}$ and $x_{12}$ partitions are enlarged to degree 6. In fact, once the $x_{12}$ partition is enlarged and echelonized, mutants are produced and after a few steps of enlarging mutants and *echelonization*, we arrive at a situation in which the $\mathrm{MXL}_3$ criterion is satisfied and the algorithm terminates.

## 5 Complexity Analysis

Studies of the complexity of Gröbner basis computation have mostly focused on the maximum degree of the polynomials that occur during computation. For example, [2] provides an exact formula for computing the degree of regularity ($D_{reg}$) of a homogeneous semi-regular sequence. In the case of a somehow generic sequence, this degree coincides with the maximum degree of the polynomials that occur during computation using the $\mathrm{F}_5$ algorithm. If the system is homogeneous, they argue that linear algebra over a square matrix of size the number of terms of degree $D_{reg}$ would solve the system and use this as an upper bound. This line of argument is sound in the case of a homogeneous semi-regular sequence.

For non-homogeneous semi-regular sequences we should add the lower degree terms to obtain

$$\sum_{d=0}^{D_{reg}} \binom{n}{d}$$

The algorithm proposed in this paper puts in question the sharpness of this bound. For example, in Section 4 we described the behavior of the algorithm for a random system of 24 degree 2 polynomials in 24 variables, in which the size of the largest matrix produced by the new algorithm was $26\,409 \times 33\,245$. This contrasts with

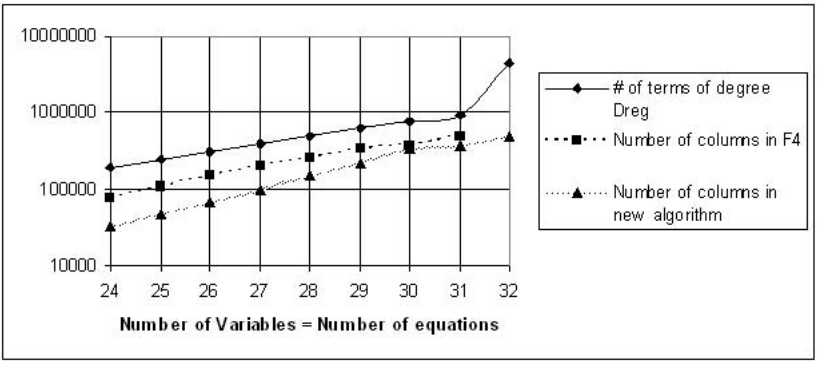



**Fig. 2.** Experimental results compared with number of terms at the degree of regularity

the number of terms up to degree 6 in 24 variables which is 190 051. Such small size was achieved because some partitions were omitted at different degrees and as a consequence some terms never appeared in the computation.

The complexity of the algorithm presented in this paper depends not only on the highest degree $D$ of the system and the number of variables $n$, but also on the sequence $n_1, n_2, \ldots, n_D$ of variables omitted at each degree $d$. The number of columns of the largest matrix is given by

$$\sum_{d=1}^{D} \binom{n - n_d}{d} + 1$$

Figure 2 compares matrix size estimated solely based on degree of regularity with experimental results. It shows a significant gap between the number of terms up to degree $D_{reg}$ and the number of columns of the new algorithm and even that of the $F_4$ algorithm (for a more complete report on the results see Section 6).

Although we don't have a way to predict or even estimate the sequence $n_1, n_2, \ldots, n_D$, It is very clear in this case that omitting substantial number of partitions could drastically change the complexity to solve the corresponding system, however it remains very speculative how this will really work and we will explore this case in a subsequent paper.

## 6 Experimental Results

We present our experiments to compare the efficiency of MGB with both $MXL_3$ and $F_4$ algorithms. We tested them with random systems generated by Courtois [3] and HFE systems generated by the code of John Baena. We run all the experiments in a Sun X4440 server, with four "Quad-Core AMD Opteron™ Processor 8356" CPUs and 128 GB of main memory. Each CPU is running at 2.3 GHz. We used only one out of the 16 cores.

Tables 1 and 2 show the main experiments of dense random systems with many solutions and the experiments of HFE systems of univariate degree 288,

**Table 1.** Experiments for dense random systems

| | $F_4$ | | $MXL_3$ | | MGB | |
|---|---|---|---|---|---|---|
| $n$ | $D$ | max. matrix | $D$ | max. matrix | $D$ | max. matrix |
| 24 | 6 | 207150×78637 | 6 | 50367×57171 | 6 | 26409×33245 |
| 25 | 6 | 248495×108746 | 6 | 66631×76414 | 6 | 37880×47594 |
| 26 | 6 | 298592×148804 | 6 | 88513×102246 | 6 | 55063×67815 |
| 27 | 6 | 354189×197902 | 6 | 123938×140344 | 6 | 92296×99518 |
| 28 | 6 | 420773×261160 | 6 | 201636×197051 | 6 | 132918×148976 |
| 29 | 6 | 499222×340254 | 6 | 279288×281192 | 6 | 173300×224941 |
| 30 | 6 | 1283869×374081 | 6 | 332615×351537 | 6 | 265298×339236 |
| 31 | 6 | 868614×489702 | 6 | 415654×436598 | 6 | 349778×381382 |
| 32 | ran out of memory | | ran out of memory | | 7 | 437172×507294 |

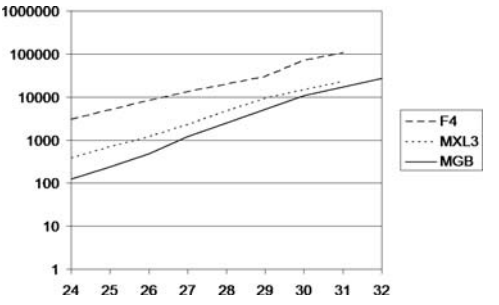
**Table 2.** Experiments for HFE(288,n) systems

| | $F_4$ | | $MXL_3$ | | MGB | |
|---|---|---|---|---|---|---|
| $n$ | $D$ | max. matrix | $D$ | max. matrix | $D$ | max. matrix |
| 30 | 5 | 149532×136004 | 5 | 86795×130211 | 5 | 68468×109007 |
| 35 | 5 | 200302×321883 | 5 | 155914×296872 | 5 | 116737×254928 |
| 36 | 5 | 219438×382252 | 5 | 173439×344968 | 5 | 125133×297503 |
| 37 | 5 | 247387×444867 | 5 | 192805×399151 | 5 | 142460×345635 |
| 38 | 5 | 274985×512311 | 5 | 212271×459985 | 5 | 153181×399855 |
| 39 | 5 | 305528×588400 | 5 | 234111×528068 | 5 | 171985×460727 |
| 40 | ran out of memory | | 5 | 258029×604033 | 5 | 192506×528849 |
| 49 | ran out of memory | | 5 | 561972×1765465 | 5 | 371368×1584984 |
| 50 | ran out of memory | | ran out of memory | | 5 | 382392×1766691 |
| 51 | ran out of memory | | ran out of memory | | 5 | 410169×1964756 |

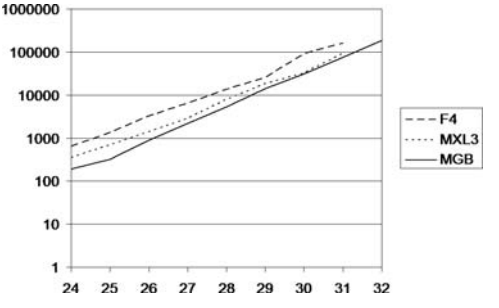
respectively. We denote the number of variables and equations by $n$ and the highest degree of the iteration steps by $D$. We also show the maximum matrix size. It is evident from Table 1 and 2 how the new strategy improves $MXL_3$.

Figure 3 displays a comparison between MGB, $MXL_3$ and $F_4$ in terms of space and time. It is clear from Figure 3(a) that the MGB algorithm uses less memory than both $MXL_3$ and Magma's $F_4$ since it constructs smallest matrices. However, it is not much faster than $MXL_3$ in terms of the size of the system becomes bigger as shown in Figure 3(b). The reason is that the new algorithm uses a row-reduced echelon form while $MXL_3$ uses only the row echelon form. Also, the gap between MGB and $MXL_3$ becomes a little smaller as the size of the system increased.

Table 3 shows the detailed result of computing a Gröbner basis of a dense random system with 32 variables by MGB. For each step we give the degree ($D$), the matrix size, the rank of the matrix (Rank), a set of leading variable of the level partitions, the number of variables in the degree $D$ terms ($n_D$), the number of lowest degree mutants found (NM), the number of used mutants (UM), and finally the lowest degree of mutants found (MD).




(a) Memory in megabyte




(b) Time in seconds

**Fig. 3.** Comparison between MGB, $\text{MXL}_3$, and $\text{F}_4$ for dense random systems

**Table 3.** Results for the system Random-32 with the MGB algorithm

| Step | D | Matrix Size | Rank | partitions | $n_D$ | NM | UM | MD |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 32×529 | 32 | $\{x_1, x_2\}$ | 32 | 0 | 0 | - |
| 2 | 3 | 1056×5489 | 1056 | $\{x_1, x_2, x_3\}$ | 32 | 0 | 0 | - |
| 3 | 4 | 11798×36954 | 11776 | $\{x_2, x_3, x_4\}$ | 31 | 0 | 0 | - |
| 4 | 5 | 93534×179460 | 91378 | $\{x_3, \ldots, x_7\}$ | 30 | 0 | 0 | - |
| 5 | 6 | 389286×475470 | 372679 | $\{x_6, \ldots, x_{16}\}$ | 27 | 0 | 0 | 0 |
| 6 | 7 | **437172×507294** | 437172 | $\{x_{15}, \ldots, x_{26}\}$ | 18 | 21445 | 2158 | 5 |
| 7 | 6 | 305685×314056 | 305685 | $\{x_9, \ldots, x_{27}\}$ | 24 | 18589 | 199 | 4 |
| 8 | 5 | 175490×179460 | 175490 | $\{x_3, \ldots, x_{28}\}$ | 30 | 3910 | 16 | 3 |
| 9 | 4 | 36875×36954 | 36875 | $\{x_2, \ldots, x_{29}\}$ | 31 | 6 | 1 | 1 |
| 10 | 2 | 535×529 | 528 | $\{x_1, \ldots, x_{31}\}$ | 32 | 25 | 0 | 1 |

Table 3 explains how the MGB algorithm works. As the degree $D$ is going up the number of variables in degree $D$ terms goes down. Starting from step 3, the number of variables of degree 4 terms starts to decrease. At step 6 the degree of the system reaches 7 by extending only two partitions of degree 6 polynomials $(x_{15}, x_{16})$, while the number of variables in degree 7 terms is only 18. The system starts to generate mutants. It generates 21445 mutants of degree 5. Only 2158 of

them are used. All of these mutants have leading variable $x_9$. So by multiplying them with variables $\geq x_9$, we have new polynomials of degree 6 with leading variables at most $x_9$. We do not multiply mutants by $x_6, x_7$, and $x_8$ since their partitions are not needed in the Gaussian elimination of step 7. This leads to decreasing the dimension of the matrix of the step. The system continuously generates low degree mutants until 6 linear mutants are produced at step 9. Another 25 linear ones are generated at step 10. By multiplying all mutants of degree $\leq 2$, the system does not produce more mutants which in turn leads to a Gröbner basis of the ideal generated by the initial 32 polynomials.

For $F_4$, we used Magma version V2.13-10 implementation of Faugère's $F_4$ algorithm which is considered the best available implementation of $F_4$. When we use the new version of Magma (V2.16), we found no big difference between them. the new version is worse in terms of memory, while it is a little bit faster. For both, the MGB algorithm and the $MXL_3$ algorithm, we used our C++ implementation. For the *Echelonization* step, we used an adapted version of M4RI [1], the dense matrix linear algebra over $\mathbb{F}_2$ library. Our adaptation was in changing the strategy of selecting the pivot during Gaussian elimination to keep the old elements in the system intact. We use the M4RI method that has complexity $O(n^3/logn)$ [1].

## 7 Conclusions and Future Work

This paper presents a new strategy to improve algorithms to compute efficiently Gröbner bases. This new strategy is to use a more flexible partial enlargement technique that avoids computing polynomials at different degrees. As the first application of this strategy, we produced a new algorithm that has the ability of computing Gröbner bases more efficiently than the $MXL_3$ algorithm, which already performed better than the $F_4$ in Magma. Our experiments confirm that the new proposed algorithm is substantially better than $MXL_3$ and $F_4$ algorithms in both randomly generated instances of MQ and HFE systems and the experiment data also suggests that the complexity of the new algorithm challenges known theoretical estimates. Our preliminary complexity analysis of this new algorithm suggested that this new strategy may change substantially our thinking on the hardness of computing Gröbner bases and this new strategy of flexible partial enlargement may leads to new paradigms in Gröbner bases computation.

We plan to study the connection between the complexity of the algorithm presented in this paper and the complexity of other Gröbner bases algorithms. We are working on a priory complexity estimates for the algorithm and security levels of various cryptosystems based on this new algorithm. We are also experimenting with other heuristics that exploit the new strategy.

## Acknowledgments

J. Ding and D. Cabarcas are especially grateful for the insightful and critical discussions with Professor Shigeo Tsujii and Mr. Masahito Gotaishi during a

2009 visit to Japan, which was supported by the "Strategic information and COmmunications R & D Promotion programmE" (SCOPE) from the Ministry of Internal Affairs and Communications of Japan. We also acknowledge useful comments from Stanislav Bulygin on a preliminary draft and valuable suggestions by anonymous referees. J. Ding would also like to thank partial support from NSF, NSF China and Taft Foundation for this project. D. Cabarcas is also partially supported by the Taft Foundation.

## References

1. Albrecht, M., Bard, G.: M4RI – linear algebra over GF(2) (2008), http://m4ri.sagemath.org/index.html
2. Bardet, M., Faugère, J.-C., Salvy, B., Yang, B.-Y.: Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In: MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry, Porto Conte, Alghero, Sardinia (Italy), May 27-June 1 (2005)
3. Courtois, N.T.: Experimental algebraic cryptanalysis of block ciphers (2007), http://www.cryptosystem.net/aes/toyciphers.html
4. Ding, J.: Mutants and its impact on polynomial solving strategies and algorithms. Privately distributed research note, University of Cincinnati and Technical University of Darmstadt (2006)
5. Ding, J., Buchmann, J., Mohamed, M.S.E., Moahmed, W.S.A., Weinmann, R.-P.: MutantXL. In: Proceedings of the 1st international conference on Symbolic Computation and Cryptography (SCC 2008), Beijing, China, April 2008, pp. 16–22. LMIB (2008)
6. Faugère, J.-C.: A new efficient algorithm for computing Gröbner bases (F4). Pure and Applied Algebra 139(1-3), 61–88 (1999)
7. Gotaishi, M., Tsujii, S.: Hxl -a variant of xl algorithm computing gröbner bases. In: Presented in Special Track on Symbolic Computation and Cryptology of the 4th International Conference on Information Security and Cryptology (Inscrypt 2008) (December 2008)
8. Mohamed, M.S.E., Cabarcas, D., Ding, J., Buchmann, J., Bulygin, S.: MXL3: An efficient algorithm for computing gröbner bases of zero-dimensional ideals. In: ICISC 2009. LNCS. Springer, Heidelberg (2009) (accepted for publication)
9. Mohamed, M.S.E., Mohamed, W.S.A.E., Ding, J., Buchmann, J.: MXL2: Solving polynomial equations over GF(2) using an improved mutant strategy. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 203–215. Springer, Heidelberg (2008)

# Factoring RSA Modulus Using Prime Reconstruction from Random Known Bits

Subhamoy Maitra, Santanu Sarkar, and Sourav Sen Gupta

Applied Statistics Unit, Indian Statistical Institute,
203 B T Road, Kolkata 700 108, India
subho@isical.ac.in, santanu_r@isical.ac.in, sg.sourav@gmail.com

**Abstract.** This paper discusses the factorization of the RSA modulus $N$ (i.e., $N = pq$, where $p, q$ are primes of same bit size) by reconstructing the primes from randomly known bits. The reconstruction method is a modified brute-force search exploiting the known bits to prune wrong branches of the search tree, thereby reducing the total search space towards possible factorization. Here we revisit the work of Heninger and Shacham in Crypto 2009 and provide a combinatorial model for the search where some random bits of the primes are known. This shows how one can factorize $N$ given the knowledge of random bits in the least significant halves of the primes. We also explain a lattice based strategy in this direction. More importantly, we study how $N$ can be factored given the knowledge of some blocks of bits in the most significant halves of the primes. We present improved theoretical result and experimental evidences in this direction.

**Keywords:** Factorization, Prime reconstruction, Random known bits, RSA.

## 1 Introduction

The RSA [12] public key cryptosystem uses two primes $p, q$ (usually of the same bit size, i.e., $q < p < 2q$ or $p < q < 2p$). The RSA modulus is $N = pq$. The factorization of $N$ cannot be done efficiently on classical computational model without the knowledge of $p, q$, and this provides the security of RSA. However, there may be different possibilities to know partial information regarding the secret parameters (through side channel attacks, say) and it is necessary to study how that can affect the security of a cryptosystem. In addition, the basic problem of integer factorization is of great interest in literature.

An extensive amount of research has been done in RSA factorization and we refer the reader to the survey papers by Boneh [1] and May [10] for a complete account. One major class of RSA attacks exploit partial knowledge of the RSA secret keys or the primes. Rivest and Shamir [11] pioneered these attacks using Integer Programming and factored RSA modulus given two-third of the LSBs of a factor. This result was improved in the seminal paper [4] by Coppersmith, where factorization of the RSA modulus could be achieved given half of the MSBs of

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 82–99, 2010.

a factor. His method used LLL [9] lattice reduction technique to solve for small solutions to modular equations. This method triggered a host of research in the field of lattice based factorization, e.g., the works by Howgrave-Graham [7], Jochemsz and May [8].

These attacks require knowledge of contiguous blocks of bits of the RSA secret keys or the primes. In a different model, one may not get contiguous blocks, but may gain the knowledge of random bits of the RSA secret keys through cold boot or other side channel attacks. In [6], it has been shown how $N$ can be factored with the knowledge of a random subset of the bits (distributed over small contiguous blocks) in one of the primes. Later, a similar result has been studied by Heninger and Shacham [5] to reconstruct the RSA private keys given a certain fraction of the bits, distributed at random. This is the work [5] where the random bits of both the primes are considered unlike the earlier works (e.g., [4,2,6]) where knowledge of the bits of a single prime have been exploited.

This paper studies how the least (respectively most) significant halves of the RSA primes can be completely recovered from some amount of randomly chosen bits from the least (respectively most) significant halves of the same. Thereafter one can exploit the existing lattice based results towards factoring the RSA modulus $N = pq$ when $p, q$ are of the same bit size. It is possible to factor $N$ in any one of the following cases in poly($\log N$) time: (i) when the most significant half of any one of the primes is known [4, Theorem 4], (ii) when the least significant half of any one of the primes is known [2, Corollary 2.2].

Road Map. In Section 2, we analyze the algorithm of [5] using a combinatorial model for reconstruction. The knowledge of random prime bits and existing lattice based method [2] allows us to factor $N$ efficiently given certain fraction of the bits of $p$ and $q$, namely about 0.5 fraction of the bits from the least significant halves of the primes when $N$ is 1024 bits. In certain cases, the strategy presented in Section 2 does not work well. To overcome this, we present a lattice based strategy in Section 3. More importantly, we propose in Section 4 an idea to reconstruct the upper half of a prime using the knowledge of certain fraction of bits in $p$ and $q$. Once one obtains the top half of any one of the primes, the factorization of $N$ is possible using existing lattice based method [4]. Theoretical results as well as extensive experimental evidences are presented to corroborate our claims.

## 2 The LSB Case: Combinatorial Analysis of [5]

In this section, we analyze the reconstruction algorithm by Heninger and Shacham [5, Section 3] from combinatorial point of view. Though the algorithm extends to all relations among the RSA secret keys, we shall concentrate our attention to the primary relation $N = pq$ for the sake of factorization. The algorithm is a smart brute-force method on the total search space of unknown bits of $p$ and $q$, which prunes the solutions those are infeasible given the knowledge of $N$ and some random bits of the primes. Henceforth, we shall denote by $l_N$ the bit size of $N$, i.e, $l_N = \lceil \log_2 N \rceil$.

### 2.1 The Reconstruction Algorithm

**Definition 1.** *Let us define $X[i]$ to be the $i$-th bit of $X$ with $X[0]$ being the LSB. Also define $X_i$ to be the partial approximation of $X$ through the bits 0 to $i$.*

Then Algorithm 1 (described below) creates all possible pairs $(p_i, q_i)$ by appending $(p[i], q[i])$ to the partial solutions $(p_{i-1}, q_{i-1})$ and prunes the incorrect ones by checking the validity of the available relation. A formal outline of Algorithm 1, which retrieves the least significant $t$ many bits of both $p, q$, is as follows. It is easy to see that the correct partial solution till the $t$ many LSBs will exist in the set of all pairs $(p_{t-1}, q_{t-1})$ found from Algorithm 1.

```
   Input: N, t and p[i], q[j], for some random values of i, j
   Output: Contiguous t many LSBs of p, q
 1 Initialize: i = 1 and p_0 = p[0] = 1, q_0 = q[0] = 1 (as both are odd);
 2 for all (p_{i-1}, q_{i-1}) do
 3 |   for all possible (p[i], q[i]) do
 4 |   |   p_i := APPEND(p[i], p_{i-1});
 5 |   |   q_i := APPEND(q[i], q_{i-1});
 6 |   |   if N ≡ p_i q_i (mod 2^{i+1}) then
 7 |   |   |   ADD the pair (p_i, q_i) at level i;
   |   |   end
   |   end
   end
 8 if i < t − 1 then
 9 |   i := i + 1;
10 |   GOTO Step 2;
   end
11 REPORT all (p_{t-1}, q_{t-1}) pairs;
```

**Algorithm 1.** The search algorithm

As one may notice, there are at most 4 possible choices for $(p[i], q[i])$ branches at any level $i$. Algorithm 1 works with all possible combinations of the bits $p[i], q[i]$ at level $i$ and hence one may want to obtain a relation between $p[i]$ and $q[i]$ in terms of the known values of $N, p_{i-1}, q_{i-1}$ so that it poses a constraint on the possibilities. Heninger and Shacham [5, Section 4] uses Multivariate Hensel's Lemma to obtain such a relation

$$p[i] + q[i] \equiv (N - p_{i-1}q_{i-1})[i] \pmod 2. \tag{1}$$

Now, this linear relation between $p[i], q[i]$ restricts the possible choices for the bits. Thus, at any level $i$, instead of 4 possibilities, the number cuts down to 2.

If we construct the search tree, then these possibilities for the bits at any level give rise to new branches in the tree. The tree at any level $i$ contains all the partial solutions $p_i, q_i$ up to the $i$-th LSB (the correct partial solution is one

among them). It is quite natural to restrict the number of potential candidates (i.e., the partial solutions) at any level so that the correct one can be found easily by exhaustive search among all the solutions and the space to store all these solutions is within certain feasible limit. This calls for restricting the width of the search tree at each level. Let us denote the width of the tree at level $i$ by $W_i$. Now we take a look at the situations (depending on the knowledge of the random bits of the primes) that control the branching behavior of the tree.

### 2.2 Growth of the Search Tree

Consider the situation where we have a pair of partials $(p_{i-1}, q_{i-1})$ and do not have any information of $(p[i], q[i])$ in Step 3 of Algorithm 1. Naturally there are 4 options, $(0,0), (0,1), (1,0)$ and $(1,1)$ for getting $(p_i, q_i)$. However, Equation (1) and the knowledge of $N$, $p_{i-1}$, $q_{i-1}$ impose a linear dependence between $p[i], q[i]$ and hence restrict the number of choices to exactly 2. If $(N - p_{i-1}q_{i-1})[i] = 0$ then we have $p[i] + q[i] \equiv 0 \pmod 2$ and $(N - p_{i-1}q_{i-1})[i] = 1$ implies $p[i] + q[i] \equiv 1 \pmod 2$. Hence the width of the tree at this level will be twice the width of the tree at the previous one, as shown in Figure 1.

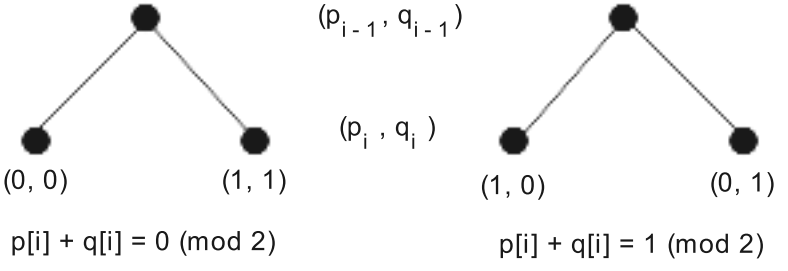



**Fig. 1.** Branching when both the bits $p[i], q[i]$ are unknown

Next, let us have a look at the situation when exactly one of $p[i], q[i]$ is known. First, the number of branches restricts to 2 by Equation (1), as discussed before. Moreover, the knowledge of one bit fixes the other in this relation. For example, if one knows the value of $p[i]$ along with $N, p_{i-1}, q_{i-1}$ in Equation (1), then $q[i]$ gets determined. Thus the number of choices for $p[i], q[i]$ and hence the number of $p_i, q_i$ branches reduces to a single one in this case. This branching, which keeps the tree-width fixed, may be illustrated as in Figure 2 ($p[i] = 0$ is known, say).

Though the earlier two cases are easy to understand, the situation is not so simple when both $p[i], q[i]$ are known. In this case, the validity of Equation (1) comes under scrutiny. If we fit in all the values $p[i], q[i], N, p_{i-1}, q_{i-1}$ in Equation (1) and it is satisfied, then we accept the new partial solution $p_i, q_i$ at level $i$ and otherwise we do not. In the case where neither of the possibilities for $p_i, q_i$ generated from $p_{i-1}, q_{i-1}$ satisfy the relation, we discard the whole subtree

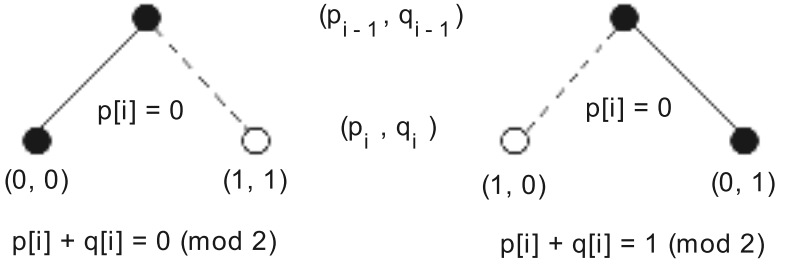



**Fig. 2.** Branching when exactly one bit of $p[i], q[i]$ is known

rooted at $p_{i-1}, q_{i-1}$. Thus, the pruning procedure not only discards the wrong ones at level $i$, but also discards subtrees from level $i-1$, thereby narrowing down the search tree. An example case ($p[i] = 0$ and $q[i] = 1$ are known, say) may be presented as in Figure 3.

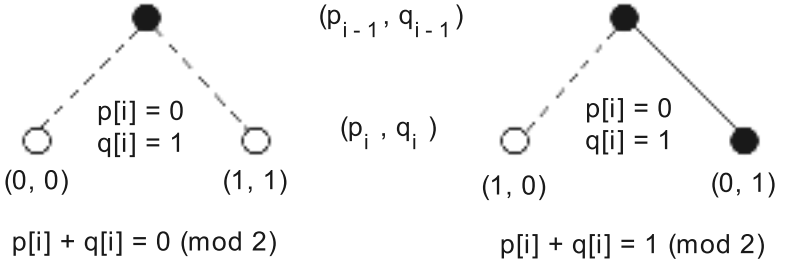



**Fig. 3.** Branching when both the bits $p[i], q[i]$ are known

Based on our discussion so far, let us try to model the growth of the search tree following Algorithm 1. As both $p, q$ are odd, we have $p[0] = 1$ and $q[0] = 1$. Thus the tree starts from $W_0 = 1$ and the expansion or contraction of the tree at each level can be modeled as follows.

- $p[i]$ = UNKNOWN, $q[i]$ = UNKNOWN: $W_i = 2W_{i-1}$.
- $p[i]$ = KNOWN, $q[i]$ = UNKNOWN: $W_i = W_{i-1}$.
- $p[i]$ = UNKNOWN, $q[i]$ = KNOWN: $W_i = W_{i-1}$.
- $p[i]$ = KNOWN, $q[i]$ = KNOWN: $W_i = \gamma_i W_{i-1}$.

Here, we assume that the tree narrows down to a $\gamma_i$ fraction ($0 < \gamma_i \leq 1$) from the earlier level if both the bits of the primes are known. One may note that Heninger and Shacham [5, Conjecture 4.3] conjectures the average value of $\gamma_i$ (call it $\gamma$) to be $\frac{1}{2}$. We shall discuss this in more details later.

Suppose that randomly chosen $\alpha$ fraction of bits of $p$ and $\beta$ fraction of bits of $q$ are known (by some side channel attack, e.g., cold boot). Then the joint probability distribution table for the bits of the primes will be as follows.

| $\downarrow q[i],\ p[i] \rightarrow$ | UNKNOWN | KNOWN |
|---|---|---|
| UNKNOWN | $(1-\alpha)(1-\beta)$ | $\alpha(1-\beta)$ |
| KNOWN | $(1-\alpha)\beta$ | $\alpha\beta$ |

As shown before, the growth of the search tree depends upon the knowledge of the bits in the primes. Hence, we can model the growth of the tree as a recursion on the level index $i$:

$$\begin{aligned} W_i &= (1-\alpha)(1-\beta)2W_{i-1} + \alpha(1-\beta)W_{i-1} + (1-\alpha)\beta W_{i-1} + \alpha\beta\gamma_i W_{i-1} \\ &= (2-\alpha-\beta+\alpha\beta\gamma_i)W_{i-1}. \end{aligned}$$

If we want to restrict $W_i$ (that is the growth of the tree) as a polynomial of $i$ (that is the number of level), we would like (roughly speaking) the value of $(2-\alpha-\beta+\alpha\beta\gamma_i)$ close to 1 on an average. Considering the average value $\gamma$ (instead of $\gamma_i$ at each level), we get, $2-\alpha-\beta+\alpha\beta\gamma \approx 1$ which implies $1-\alpha-\beta+\alpha\beta\gamma \approx 0$. If we assume that the same fraction of bits are known for $p$ and $q$, then $\alpha = \beta$ and we get $1-2\alpha+\alpha^2\gamma \approx 0 \quad \Rightarrow \quad \alpha \approx \frac{1-\sqrt{1-\gamma}}{\gamma}$. If we assume [5, Conjecture 4.3], then $\gamma \approx 0.5$ and hence $\alpha \approx 2-\sqrt{2} \approx 0.5858$, as obtained in [5, Section 4.4]. One may note that our idea is simpler compared to the explanation in [5]. This simplification is achieved here by using average value for $\gamma_i$ in the recurrence relation of $W_i$.

The most natural strategy is to first apply Algorithm 1 to retrieve the least significant half of any one of the primes and then apply the result of Boneh et. al. [2, Corollary 2.2] to factorize $N$. One may note that [5] utilizes their prime reconstruction algorithm to reconstruct the whole primes $p, q$ whereas our idea is to use lattice based results after reconstructing just one half of any prime. This is more practical as it requires the knowledge of lesser number of random bits of the primes, namely, just about $0.5858 \times 0.5 \approx 0.3$ fraction of bits (from the LSB half) instead of 0.5858 fraction of the primes as explained in [5]. Moreover, factorization being the main objective, one need not reconstruct the primes completely, but just requires to obtain enough information that suffices for factoring the product $N$ based on the existing efficient techniques. In this direction, let us first present the following result.

### 2.3 Known Prime Bits: Complementary Sets for $p, q$

**Theorem 1.** *Let $N = pq$, when $p, q$ are primes of same bit size. Let $S = \{0, \ldots, \lceil l_N/4 \rceil\}$. Consider $U \subseteq S$ and $V = S \setminus U$. Assume that $p[i]$'s for $i \in U$ and $q[j]$'s for $j \in V$ are known. Then one can factor $N$ in poly$(\log N)$ time.*

*Proof.* Let us apply Algorithm 1 in this case to retrieve the bits of the primes at each level. We shall use induction on the index of levels in this case.

For level 0, we know that $p[0] = 1$ and $q[0] = 1$. Hence, the width of the search tree is $W_0 = 1$ and we have a single correct partial $(p_0, q_0)$. Let us suppose that we have possible pairs of partials $(p_{i-1}, q_{i-1})$ at level $i-1$, generated by Algorithm 1. At level $i$, two cases may arise. If $i \in U$ then we know $p[i], N, p_{i-1}, q_{i-1}$ which restricts the branching to a single branch and keeps the width of the tree fixed ($W_i = W_{i-1}$). Else one must have $i \in V$ ($V = S \setminus U$) and we know $q[i], N, p_{i-1}, q_{i-1}$. This restricts the branching to a single branch as well and keeps the width fixed ($W_i = W_{i-1}$). Hence, by induction on $i$, $W_i = W_{i-1}$ for $i = 0, \ldots, \lceil l_N/4 \rceil$. As $W_0 = 1$, this boils down to $W_i = 1$ for $i \leq \lceil l_N/4 \rceil$.

Thus we obtain a single correct partial pair $p_i, q_i$ at level $i = \lceil l_N/4 \rceil$ using Algorithm 1 in $O(\log^3 N)$ time ($\lceil l_N/4 \rceil$ iterations and $O(\log^2 N)$ computing time for each iteration) and $O(l_N/2)$ space (actually we need space to store a single partial pair at the current level). This provides us with the least significant half of both the primes and using any one of those two, the lattice based method of [2, Corollary 2.2] completes the factorization of $N = pq$ in poly($\log N$) time. □

It is interesting to analyze the implications of this result in a few specific cases. An extreme case may be $U = S$, that is we know all the bits in the least significant half of a single prime $p$ and do not know any such bits for $q$. In this scenario, one need not apply Algorithm 1 at all and the lattice based method in [2, Corollary 2.2] suffices for factorization. Second case is when $|U| = |S| - x$, i.e., missing $x$ bits of $p$ at random positions. In such a case, one can use a brute force search for these missing bits and apply lattice based factoring method [2, Corollary 2.2] for all of the $2^x$ possibilities, if $x$ is small. However, for large $x$, e.g., $x \approx |U|$, i.e., around half of the random bits from the least significant halves of $p$ as well as $q$ are known, then the brute force strategy fails, and one must use Algorithm 1 before applying the lattice based method in [2, Corollary 2.2].

### 2.4 Known Prime Bits: Distributed at Random

Here we consider the case when random bits of $p, q$ are available, lifting the constraint $V = S \setminus U$. That is, here we only consider $U, V$ to be random subsets of $S$. For 512-bit primes, we observed that knowledge of randomly chosen half of the bits from least significant halves of $p, q$ is sufficient to recover the complete least significant halves of $p$ as well as $q$ using Algorithm 1.

Now let us present a select few of our experimental results in Table 1. The first column represents the size of the RSA primes and the second column gives the fraction of bits known randomly from the least significant halves of the primes (call these $\alpha_p, \beta_q$ respectively). The value of $t$ in the third column is the target level we need to run Algorithm 1 for, and is half the size of the primes. $W_t$ is the final width of the search tree at the target level $t$. This denotes the number of possible partial solutions for $p, q$ at the target bit level $t$, whereas the next column gives us the maximum width of the tree observed during the run of Algorithm 1. The last column depicts the average value of the shrink ratio $\gamma$, as we have defined earlier.

**Table 1.** Experimental results corresponding to Algorithm 1

| Size $\|p\|, \|q\|$ | Known $\alpha_p, \beta_q$ | Target $t$ | Final $W_t$ | $\max_{i=1}^{t} W_i$ | Average $\gamma$ |
|---|---|---|---|---|---|
| 256, 256 | 0.5, 0.5 | 128 | 30 | 60 | 0.56 |
| 256, 256 | 0.5, 0.5 | 128 | 2816 | 5632 | 0.52 |
| 256, 256 | 0.47, 0.47 | 128 | 106 | 1508 | 0.54 |
| 256, 256 | 0.45, 0.45 | 128 | 6144 | 6144 | 0.49 |
| 512, 512 | 0.5, 0.5 | 256 | 352 | 928 | 0.53 |
| 512, 512 | 0.5, 0.5 | 256 | 8 | 256 | 0.55 |
| 512, 512 | 0.5, 0.5 | 256 | 716 | 3776 | 0.53 |
| 512, 512 | 0.5, 0.5 | 256 | 152 | 2240 | 0.59 |
| 512, 512 | 0.55, 0.45 | 256 | 37 | 268 | 0.51 |
| 512, 512 | 0.55, 0.45 | 256 | 64 | 334 | 0.51 |
| 512, 512 | 0.6, 0.4 | 256 | 1648 | 13528 | 0.55 |
| 512, 512 | 0.6, 0.4 | 256 | 704 | 5632 | 0.56 |
| 512, 512 | 0.7, 0.3 | 256 | 158 | 1344 | 0.53 |
| 512, 512 | 0.7, 0.3 | 256 | 47 | 4848 | 0.52 |
| 1024,1024 | 0.55, 0.55 | 512 | 1 | 352 | 0.53 |
| 1024,1024 | 0.53, 0.53 | 512 | 16 | 764 | 0.53 |
| 1024,1024 | 0.51, 0.51 | 512 | 138 | 15551 | 0.54 |
| 1024,1024 | 0.51, 0.5 | 512 | 17 | 4088 | 0.52 |

A few crucial observations can be made from the data presented in Table 1. We have run the experiments for different sizes of RSA keys, and though the theoretical requirement for the fraction of known bits $(\alpha, \beta)$ is 0.5858, we have obtained better results when $l_N \leq 2048$. For 512 bit $N$, the knowledge of just 0.45 fraction of random bits from the least significant halves of the primes proves to be sufficient for Algorithm 1 to retrieve the halves, whereas for 1024 and 2048 bit $N$, we require about 0.5 fraction of such bits. The main reason is that the growth of the search tree increases with increasing size of the target level $t$. As we have discussed before, the growth will be independent of the target if we know 0.5858 fraction of bits instead. One may also note that for 1024 bit $N$, we have obtained successful results when the fraction of bits known is not the same for the two primes. For such skewed cases, the average requirement of known bits stay the same, i.e, 0.5 fraction of the least significant halves. The examples for (0.7, 0.3) in such skewed cases provides interesting results compared to the result by Herrmann and May [6]. Knowing about 70% of the bits of one prime is sufficient for their method to factorize $N$, but the runtime is exponential in the number of blocks over which the bits are distributed. By knowing 35% of one prime (70% from the least significant half) and 15% of the other (30% of the least significant half), Algorithm 1 can produce significantly better results in the same direction.

Another important point to note from the results is the average value of the shrink ratio $\gamma$. It is conjectured in [5] that $\gamma = 0.5$. However, our experiments clearly show that the value of $\gamma$ is more than 0.5 in *most* (17 out of 18) of the cases. A theoretical explanation of this anomaly may be of interest.

### 2.5 Known Prime Bits: Distributed in a Pattern

In addition to these results, some interesting cases appear when we consider the knowledge of the bits to be periodic in a systematic pattern, instead of being totally random. Suppose that the bits of the primes $p, q$ are available in the following pattern: none of the bits is known over a stretch of $U$ bits, only $q[i]$ is known for $Q$ bits, only $p[i]$ is known for $P$ bits and both $p[i], q[i]$ are known for $K$ bits. This pattern of length $U + P + Q + K$ repeats over the total number of bits. In such a case, one may expect the growth of the tree to obey the following heuristic model – grows in doubles for $U$ bits, stays the same for $Q+P$ length and shrinks thereafter (approximately by halves, considering $\gamma = 0.5$) for a stretch of $K$ bits. If this model is followed strictly, one expects the growth of the tree by a factor of $2^U 2^{-K} = 2^{U-K}$ over each period of the pattern. The total number of occurrences of this pattern over the stretch of $T$ bits is roughly $\frac{T}{U+Q+P+K}$. Hence the width of the tree at level $T$ may be roughly estimated by $W_T \approx \left[2^{U-K}\right]^{\frac{T}{U+Q+P+K}} = 2^{\frac{T(U-K)}{U+Q+P+K}}$. A closer look reveals a slightly different observation. We have expected that the tree shrinks in half if both bits are known, which is based on the conjecture that $\gamma \approx 1/2$ on an average. But in practical scenario, this is not the case. So, the width $W_T$ at level $T$, as estimated above, comes as an underestimate in most of the cases.

Let us consider a specific example for such a band-LSB case. The pattern followed is $[U = 5, Q = 3, P = 3, K = 5]$. Using the estimation formula above, one expects the final width of the tree at level 256 to be 1, as $U = K$. But in this case, the final width turns out to be 8 instead. The reason behind this is that the average value of $\gamma$ in this experiment is 0.55 instead of 0.5.

It is natural for one to notice that the fraction of bits to be known in this band-LSB case is $(P+K)/(U+Q+P+K)$ for the prime $p$ and $(Q+K)/(U+Q+P+K)$ for the prime $q$. If we choose $Q = P$ and $U = K$, then this fraction is 0.5. Thus, by knowing 50% of the bits from the least significant halves of the primes, that is, knowing just 0.25 fraction of bits in total, Algorithm 1 can factorize $N = pq$ in this case. One may note that the result by Herrmann and May [6] requires the knowledge of about 70% of the bits distributed over arbitrary number of small blocks of a single prime. Thus, in terms of total number of bits to be known (considering both the primes), our result is clearly better.

An extension of this idea may be applied in case of MSBs. Though we can retrieve information about the primes from random bits at the least significant side, we could not exploit similar information from the most significant part. But we could do better if bands of bits are known instead of isolated random bits. A novel idea for reconstructing primes based on such knowledge is presented in Section 4.

## 3 The LSB Case: Lattice Based Technique

Consider the scenario when a long run (length $u$) of $p[i], q[i]$ is not known, for $k < i \leq k + u$ say, starting at the $(k+1)$-th bit level. In such a case, Algorithm 1

will require large memory as the width of the tree will be at least $2^u$ at the $u$-th level. If $u$ is large, say $u \geq 50$, then it will be hard to accommodate the number of options, which is greater than $2^{50}$. We describe a lattice based method below to handle such situation.

For basics related to lattices and solution to modular equations using lattice techniques, one may refer to [4,7,8]. First we recall the following result from [7].

**Lemma 1.** *Let $g(x, y) \in \mathbb{Z}[x, y]$ be a polynomial which is the sum of $\omega$ many monomials. Suppose $g(x_1, y_1) \equiv 0 \bmod n$, where $|x_1| < X_1$ and $|y_1| < Y_1$. If $\| g(xX_1, yY_1) \|_2 < \frac{n}{\sqrt{\omega}}$, then $g(x_1, y_1) = 0$ holds over integers.*

We apply resultant techniques to solve for the roots of the bivariate polynomials. It may sometimes happen that the resultant between the two bivariate polynomials is zero. There is no way to avoid this and it is a common problem in bivariate Coppersmith method. Thus one cannot always find common roots using this method. Though our technique works in practice as noted from the experiments we perform, we formally state the following assumption, which proves to be crucial in Theorem 2.

**Assumption 1.** *Let $\{f_1, f_2\}$ be two polynomials in two variables sharing common roots of the form $(x_1, y_1)$. Then it is possible to find the roots efficiently by calculating the resultant of $\{f_1, f_2\}$.*

Now we will state and prove the main result of this section.

**Theorem 2.** *Let $N = pq$ where $p, q$ are of same bit size. Suppose $\tau l_N$ many least significant bits (LSBs) of $p, q$ are unknown but the subsequent $\eta l_N$ many LSBs of both $p, q$ are known. Then, under Assumption 1, one can recover the $\tau l_N$ many unknown LSBs of $p, q$ in poly$(\log N)$ time, if $\tau < \frac{\eta}{2}$.*

*Proof.* Let $p_0$ correspond to the known $\eta l_N$ many bits of $p$ and $q_0$ correspond to the known $\eta l_N$ bits of $q$. Let $p_1$ correspond to the unknown $\tau l_N$ many bits of $p$ and $q_1$ correspond to the unknown $\tau l_N$ bits of $q$. Then we have $(2^{\tau l_N} p_0 + p_1)(2^{\tau l_N} q_0 + q_1) \equiv N \bmod (2^{(\tau+\eta) l_N})$. Let $T = 2^{(\tau+\eta) l_N}$. Hence we are interested to find the roots $(p_1, q_1)$ of $f(x, y) = (2^{\tau l_N} p_0 + x)(2^{\tau l_N} q_0 + y) - N$ over $\mathbb{Z}_T$.

Let us take $X = 2^{\tau l_N}$ and $Y = 2^{\tau l_N}$. One may note that $X, Y$ are the upper bounds of the roots $(p_1, q_1)$ of $f(x, y)$, neglecting small constants. For a non negative integer $m$, we define two sets of polynomials

$g_{i,j}(x, y) = x^i f^j(x, y) T^{m-j}$, where $j = 0, \ldots, m,\ i = 0, \ldots, m - j$ and

$h_{i,j}(x, y) = y^i f^j(x, y) T^{m-j}$, where $j = 0, \ldots, m,\ i = 1, \ldots, m - j$.

Note that $g_{i,j}(p_1, q_1) \equiv 0 \bmod (T^m)$ and $h_{i,j}(p_1, q_1) \equiv 0 \bmod (T^m)$. We call $g_{i,j}$ the $x$-shift and $h_{i,j}$ the $y$-shift polynomials, as per their respective constructions following the idea of [8].

Next, we form a lattice $L$ by taking the coefficient vectors of the shift polynomials $g_{i,j}(xX, yY)$ and $h_{i,j}(xX, yY)$ as basis. One can verify that the dimension of the lattice $L$ is $\omega = (m + 1)^2$. The matrix containing the basis vectors of $L$ is lower triangular and has diagonal entries of the form $X^{i+j} Y^j T^{m-j}$, for

$j = 0, \ldots, m$ and $i = 0, \ldots, m-j$, and $X^j Y^{i+j} T^{m-j}$ for $j = 0, \ldots, m$ and $i = 1, \ldots, m-j$, coming from $g_{i,j}$ and $h_{i,j}$ respectively. Thus, one can calculate

$$\det(L) = \left[\prod_{j=0}^{m}\prod_{i=0}^{m-j} X^{i+j}Y^{j}T^{m-j}\right]\left[\prod_{j=0}^{m}\prod_{i=1}^{m-j} X^{j}Y^{i+j}T^{m-j}\right] = X^{s_1}Y^{s_2}T^{s_3}$$

where $s_1 = \frac{1}{2}m^3 + m^2 + \frac{1}{2}m$, $s_2 = \frac{1}{2}m^3 + m^2 + \frac{1}{2}m$, and $s_3 = \frac{2}{3}m^3 + \frac{3}{2}m^2 + \frac{5}{6}m$.

To utilize resultant techniques and Assumption 1, we need two polynomials $f_1(x, y)$, $f_2(x, y)$ which share the roots $(p_1, q_1)$ over integers. From Lemma 1, we know that one can find such $f_1(x, y), f_2(x, y)$ using LLL lattice reduction algorithm [9] over $L$ when $det(L) < T^{m\omega}$, neglecting the small constants. Given $det(L)$ and $\omega$ as above, the condition becomes $X^{s_1}Y^{s_2}T^{s_3} < T^{m((m+1)^2)}$, i.e., $X^{s_1}Y^{s_2} < T^{s_0}$, where $s_0 = m((m+1)^2) - s_3 = \frac{1}{3}m^3 + \frac{1}{2}m^2 + \frac{1}{6}m$. Putting the values of the bounds $X = Y = 2^{\tau l_N}$, and neglecting $o(m^3)$ terms, we get $\frac{\tau}{2} + \frac{\tau}{2} < \frac{\tau+\eta}{3}$ and thus get the required bound for $\tau$. Now, one can find the root $(p_1, q_1)$ from $f_1, f_2$ under Assumption 1. The claimed time complexity of $\text{poly}(\log N)$ can be achieved as

- the time complexity of the LLL lattice reduction is $\text{poly}(\log N)$; and
- given a fixed lattice dimension of small size, we get constant degree polynomials and the complexity of resultant calculation is polynomial in the sum-of-degrees of the polynomials.

This completes the proof of Theorem 2. □

This lattice based technique complements Algorithm 1 by overcoming one of its limitations. As we discussed before, the search tree grows two-folds each time we do not have any information about the bits of the primes. Hence in a case where an initial chunk of LSBs is unknown for both the primes, one can not use Algorithm 1 for reconstruction as it would require huge storage space for the search tree. This lattice based technique provides a feasible solution in such a case. We present a few experimental results in Table 2 to illustrate the operation of this technique. All the experiments have been performed in SAGE 4.1 over Linux Ubuntu 8.04 on a Compaq machine with Dual CORE Intel(R) Pentium(R) D CPU 1.83 GHz, 2 GB RAM and 2 MB Cache.

**Table 2.** Experimental runs of the Lattice Based Technique with lattice dimension 64

| # of Unknown bits ($\tau l_N$) | # of Known bits ($\eta l_N$) | Time in Seconds | | |
|---|---|---|---|---|
| | | LLL Algorithm | Resultant Calculation | Root Extraction |
| 40 | 90 | 36.66 | 25.67 | $< 1$ |
| 50 | 110 | 47.31 | 35.20 | $< 1$ |
| 60 | 135 | 69.23 | 47.14 | $< 1$ |
| 70 | 155 | 73.15 | 58.04 | $< 1$ |

The limitation of this technique is that it asks for double or more the number of missing bits for both the primes. If one misses 60 LSBs for the primes say, this method requires the next 120 or more bits of both the primes to be known to reconstruct all $60 + 120 = 180$ LSBs. In the practical scenario, the requirement of bits to be known is 135 (shown in Table 2), instead of 120, as we use limited lattice dimensions in the experiments. In all the cases mentioned above, we miss the first $\tau l_N$ LSBs of the primes. If we miss the information of the bits of the prime in a contiguous block of size $\tau l_N$ somewhere in the middle, after the $i$-th level say, then this method offers similar solution if we have $\eta > 2\tau + 2i/l_N$.

## 4 The MSB Case: Our Method and Its Analysis

In this section, we put forward a novel idea of reconstructing the most significant half of the primes $p, q$ given the knowledge of some blocks of bits. To the best of our knowledge, this has not been studied in a disciplined manner in the existing literature. As before, $N = pq$, and the primes $p, q$ are of the same bit size. For this section of MSB reconstruction, let us propose the following definition to make notations simpler.

**Definition 2.** *Let us define $X[i]$ to be the $i$-th most significant bit of $X$ with $X[0]$ being the MSB. Also define $X_i$ to be the partial approximation of $X$ where $X_i$ shares the most significant bits 0 through $i$ with $X$. Let $l_X$ denote the bit size of $X$, i.e, $l_X = \lceil \log_2 X \rceil$.*

### 4.1 The Reconstruction Idea

The idea for reconstructing the most significant halves of the primes is quite simple. We shall use the basic relation $N = pq$. If one of the primes, $p$ say, is known, the other one is easy to find by $q = N/p$. Now, if a few MSBs of one of the primes, $p$ say, is known, then we may obtain an approximation $p'$ of $p$. This allows us to construct an approximation $q' = \lceil N/p' \rceil$ of the other prime $q$ as well. Our idea is to use the known blocks of bits of the primes in a systematic order to repeat this approximation process until we recover half of one of the primes. A few obvious questions may be as follows.

- How accurate are the approximations?
- How probable is the success of the reconstruction process?
- How many bits of the primes do we need to know?

We answer these questions by describing the reconstruction algorithm in Section 4.2 and analyzing the same in Section 4.3. But first, let us present an outline of our idea.

Suppose that we have the knowledge of MSBs $\{0, \ldots, a\}$ of prime $p$. This allows us to construct an approximation $p_a$ of $p$, and hence an approximation $q' = \lceil N/p_a \rceil$ of $q$. Lemma 2, discussed later in Section 4.3, tells us that $q'$ matches $q$ through MSBs $\{0, \ldots, a-t-1\}$, i.e, $q' = q_{a-t-1}$, with some probability

depending on $t$. Now, if one knows the MSBs $\{a-t,\ldots,2a\}$ of $q$, then a better approximation $q_{2a}$ may be constructed using $q_{a-t-1}$ and these known bits. Again, $q_{2a}$ facilitates the construction of a new approximation $p' = \lceil N/q_{2a} \rceil$, which by Lemma 2, satisfies $p' = p_{2a-t-1}$ with some probability depending on $t$. With the knowledge of MSBs $\{2a-t,\ldots,3a\}$ of $p$, it is once again possible to construct a better approximation $p_{3a}$ of $p$. This process of constructing approximations is recursed until one reconstructs the most significant half of one of the primes. A graphical representation of the reconstruction process is illustrated in Figure 4.

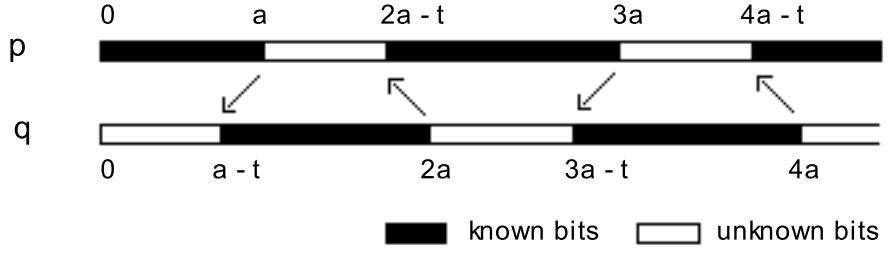



**Fig. 4.** The feedback mechanism in MSB reconstruction

### 4.2 The Reconstruction Algorithm

Let $S = \{0,\ldots,T\}$ denote the set of bit indices from the most significant halves of the primes. Let us assume that $k = \lfloor T/a \rfloor$ is odd in this case. Consider $U, V \subseteq S$ such that $U = \{0,\ldots,a\} \cup \{2a-t,\ldots,3a\} \cup \cdots \cup \{(k-1)a-t,\ldots,ka\}$, $V = \{a-t,\ldots,2a\} \cup \{3a-t,\ldots,4a\} \cup \cdots \cup \{ka-t,\ldots,T\}$. Also consider that $p[i]$'s are known for $i \in U$ and $q[j]$'s are known for $j \in V$. Then, Algorithm 2 reconstructs $T$ many contiguous most significant bits of the prime $q$.

The subroutine CORRECT used in Algorithm 2 (and Algorithm 4 later) takes as input a partial approximation $Y$ of $X$ and a set of contiguous known bits, $X[i]$ for $i \in \Sigma$, say. It outputs a better approximation $Z$ of $X$ by correcting the bits of the partial approximation $Y$ using the knowledge of the known bits. Formally, the subroutine works as described is Algorithm 3.

In the case where $k = \lfloor T/a \rfloor$ is even, Algorithm 2 needs to be tweaked a little to work as expected. One may consider a slightly changed version of $U, V \subseteq S$ such that $U = \{0,\ldots,a\} \cup \{2a-t,\ldots,3a\} \cup \cdots \cup \{ka-t,\ldots,T\}$ and $V = \{a-t,\ldots,2a\} \cup \{3a-t,\ldots,4a\} \cup \cdots \cup \{(k-1)a-t,\ldots,ka\}$. As before, $p[i]$'s are known for $i \in U$ and $q[j]$'s are known for $j \in V$. Then, Algorithm 4 reconstructs $T$ many contiguous most significant bits of the prime $p$.

### 4.3 Analysis of the Reconstruction Algorithm

Algorithm 2 and Algorithm 4 follow the same basic idea of reconstruction as discussed in Section 4.1, and differs only in a minor issue regarding the practical implementation. We have stated both the algorithms in Section 4.2 for the sake

**Input**: $N, T$ and $p[i], q[j]$ for all $i \in U$ and $j \in V$
**Output**: Contiguous $T$ many MSBs of $q$
Initialize: $p_0 := 2^{l_p-1}$, $q_0 := 2^{l_q-1}$;
$p_a :=$ CORRECT($p_0$, $p[j]$ for $j \in \{1, \ldots, a\} \subset U$);
$q_{a-t} := \lceil \frac{N}{p_a} \rceil$;
**for** *i from* 2 *to* $k-1$ *in steps of* 2 **do**
  $q_{ia} :=$ CORRECT($q_{(i-1)a-t}$, $q[j]$ for $j \in \{(i-1)a-t, \ldots, ia\} \subset V$);
  $p_{ia-t-1} := \lceil \frac{N}{q_{ia}} \rceil$;
  $p_{(i+1)a} :=$ CORRECT($p_{ia-t-1}$, $p[j]$ for $j \in \{ia-t, \ldots, (i+1)a\} \subset U$);
  $q_{(i+1)a-t-1} := \lceil \frac{N}{p_{(i+1)a}} \rceil$;
**end**
$q_T :=$ CORRECT($q_{ka-t-1}$, $q[j]$ for $j \in \{ka-t, \ldots, T\} \subset V$);
REPORT $q_T$;

**Algorithm 2.** The MSB reconstruction algorithm [$k$ odd]

**Input**: $Y$ and $X[i]$ for $i \in \Sigma$
**Output**: $Z$, the correction of $Y$
**for** *j from* 0 *to* $l_X$ **do**
  **if** $j \in \Sigma$ **then** $Z[j] = X[j]$; `// Correct the j-th MSB if the bit X[j] is known`
  **else** $Z[j] = Y[j]$; `// Keep the j-th MSB of Y as X[j] is not known`
**end**
REPORT $Z$;

**Algorithm 3.** Subroutine CORRECT

of completeness. But in case of the analysis and the experimental results, we shall consider only one of them, Algorithm 2 say, without loss of generality.

Algorithm 2 requires the knowledge of at most $(T - ka + 1) + k(a + t + 1) \leq k(a+t) + (k+a) \leq T(1 + \frac{t}{a}) + (k+a)$ many bits of $p$ and $q$ to (probabilistically) reconstruct $T$ contiguous MSBs of one prime. The runtime of Algorithm 2 is linear in terms of the number of known blocks, i.e, linear in terms of $k = \lfloor T/a \rfloor$. If we set the target $T = l_N/4$, then Algorithm 2 outputs the most significant half of one of the primes in $O(k)$ steps with some probability of success depending on $a$ and $t$. In this context, we propose Theorem 3 to estimate the probability of success of Algorithm 2. Before that, let us introduce the following technical result (Lemma 2) which is necessary to prove Theorem 3.

**Lemma 2.** *If $X$ and $X'$ are two integers with same bit size and $|X - X'| < 2^H$, then the probability that $X$ and $X'$ share $l_X - H - t$ many MSBs for some $0 \leq t \leq H$ is at least $P_t = 1 - \frac{1}{2^t}$.*

*Proof.* We know that $|X - X'| < 2^H$, i.e, $X = X' + Y$ or $X' = X + Z$ where $0 \leq Y, Z < 2^H$, say. Let us consider the case $X = X' + Y$ first, and the other case will follow by symmetry between $X$ and $X'$.

**Input**: $N, T$ and $p[i], q[j]$ for all $i \in U$ and $j \in V$
**Output**: Contiguous $T$ many MSBs of $p$

Initialize: $p_0 := 2^{l_p - 1}$, $q_0 := 2^{l_q - 1}$;
$p_a :=$ CORRECT($p_0$, $p[j]$ for $j \in \{1, \ldots, a\} \subset U$);
**for** $i$ *from* 1 *to* $k-3$ *in steps of* 2 **do**
  $q_{ia-t-1} := \lceil \frac{N}{p_{ia}} \rceil$;
  $q_{(i+1)a} :=$ CORRECT($q_{ia-t-1}$, $q[j]$ for $j \in \{ia - t, \ldots, (i+1)a\} \subset V$);
  $p_{(i+1)a-t-1} := \lceil \frac{N}{q_{(i+1)a}} \rceil$;
  $p_{(i+2)a} :=$ CORRECT($p_{(i+1)a-t-1}$, $p[j]$ for $j \in \{(i+1)a - t, \ldots, (i+2)a\} \subset U$);
**end**
$q_{(k-1)a-t-1} := \lceil \frac{N}{p_{(k-1)a}} \rceil$;
$q_{ka} :=$ CORRECT($q_{(k-1)a-t-1}$, $q[j]$ for $j \in \{(k-1)a - t, \ldots, ka\} \subset V$);
$p_{ka-t-1} := \lceil \frac{N}{q_{ka}} \rceil$;
$p_T :=$ CORRECT($p_{ka-t-1}$, $p[j]$ for $j \in \{ka - t, \ldots, T\} \subset U$);
REPORT $p_T$;

**Algorithm 4.** The MSB reconstruction algorithm [$k$ even]

Let us split $X' = 2^H X_0 + X_1$. Then, clearly $X = 2^H X_0 + (X_1 + Y)$. The addition of $Y < 2^H$ affects the lower part $X_1$ directly and the carry from the sum $(X_1 + Y)$ affects the first half $X_0$ up to a certain level. Our goal is to find out the probability that the carry affects less than or equal to $t$ bits of $X_0$ from the lower side. Let us assume that the probability of $(X_1 + Y)$ generating a carry bit is $p_c$. We also know that this carry bit will propagate through the lower bits of $X_0$ until it hits a 0, and we can assume any bit of $X_0$ to be 0 or 1 randomly with equal probabilities of 1/2 each. Then, the probability of the carry bit to propagate less than or equal to $t$ bits of $X_0$ from the lower side is
$P[\text{carry propagation} \leq t]$

$= P[\text{no carry}] + \sum_{i=1}^{t} P[\text{carry}] P[\text{carry propagation} = i]$
$= P[\text{no carry}] + \sum_{i=1}^{t} P[\text{carry}] P[\text{first 0 occurs at } i\text{-th LSB of } X_0]$
$= (1 - p_c) + \sum_{i=1}^{t} p_c \frac{1}{2^i} = 1 - \frac{p_c}{2^t}$.

Now, one may expect the probability of carry generated from the sum $(X_1 + Y)$ to be $p_c \approx 1/2$. A careful statistical modelling of the difference $Y$ will reveal a better estimate of $p_c$. As we do not assume any distribution of $Y$ here, we consider the trivial bound $p_c \leq 1$. Thus the probability of $X$ and $X_0$, and hence $X$ and $X'$, sharing $l_X - H - t$ many MSBs is $1 - \frac{p_c}{2^t} \geq 1 - \frac{1}{2^t}$. □

At this point, we can state and prove the main result of this section, the following theorem.

**Theorem 3.** *Let* $S = \{0, \ldots, T\}$ *and* $k = \lfloor T/a \rfloor$ *is odd. Suppose* $U, V \subseteq S$ *such that* $U = \{0, \ldots, a\} \cup \{2a - t, \ldots, 3a\} \cup \cdots \cup \{(k-1)a - t, \ldots, ka\}$, $V = \{a - t, \ldots, 2a\} \cup \{3a - t, \ldots, 4a\} \cup \cdots \cup \{ka - t, \ldots, T\}$, *where* $p[i]$*'s are known for* $i \in U$ *and* $q[j]$*'s are known for* $j \in V$*, as discussed before. Then, Algorithm 2*

*reconstructs $T$ many contiguous most significant bits of one of the primes correctly in $O(k)$ steps with probability at least* $P_{a,t}(T) = \left(1 - \frac{1}{2^t}\right)^{\lfloor T/a \rfloor}$.

*Proof.* The success of Algorithm 2 relies on the correct construction of the approximations at various levels. The CORRECT function produces correct approximations with probability 1 given the known sets of bits $U, V$, as mentioned before. Hence, success probability of Algorithm 2 depends on the correctness of $\{q_{a-t-1}, p_{2a-t-1}, q_{3a-t-1}, \ldots, p_{(k-1)a-t-1}, q_{ka-t-1}\}$.

Let us first consider the case $p > q$. We know that in such a case, as $p, q$ are of the same bit size, one must have $\sqrt{N/2} < q < \sqrt{N} < p < \sqrt{2N}$. Suppose that there exists an approximation $p_{ha}$ of $p$, sharing the MSBs $\{0, \ldots, ha\}$ for some $1 \leq h \leq k$. In this case, $|p - p_{ha}| < 2^{l_p - ha}$. Using $p_{ha}$, one constructs an approximation $q' = \lceil N/p_{ha} \rceil$ of $q$. Then we have $|q - q'| \approx \left|\frac{N}{p} - \frac{N}{p_{ha}}\right| = \frac{N}{pp_{ha}} |p - p_{ha}| < |p - p_{ha}| < 2^{l_p - ha}$, as $p, p_{ha} > \sqrt{N}$. If $p_{ha} < \sqrt{N}$ from the initial approximation, we reassign $p_{ha} = \lceil \sqrt{N} \rceil$ as a better approximation to $p$. The case $p < q$ produces an approximation $q'$ of $q$ with $|q - q'| < 2|p - p_{ha}| < 2^{l_p - ha + 1}$.

Then, we know for sure that $|q - q'| < 2^{l_p - ha + 1}$. Thus, by Lemma 2, setting $H = l_p - ha + 1$, we get that $q$ and $q'$ share the first $l_p - (l_p - ha + 1) - t = ha - t - 1$ MSBs with probability at least $P_t = 1 - \frac{1}{2^t}$. In other words, the probability that $q'$ correctly represents $q_{ha-t-1}$ is at least $P_t = 1 - \frac{1}{2^t}$. The probability of correctness is the same in case of the approximations of $p$ by $p_{ga-t-1}$ for all $1 < g < k$.

Now, the $k$ approximations of $p, q$ at different bit levels, as mentioned above, can be considered independent. Hence, the probability of success of Algorithm 2 in constructing $T$ many contiguous MSBs of $q$ (or $p$ in another case) is at least $P_{a,t} = P_t^k = \left(1 - \frac{1}{2^t}\right)^k = \left(1 - \frac{1}{2^t}\right)^{\lfloor T/a \rfloor}$. □

Once the most significant half of any one of the primes is known using Algorithm 2, one may use a lattice based method of to factorize $N = pq$. In this context, let us present the following result for factoring the RSA modulus $N$ using Algorithm 2.

**Corollary 1.** *Let $S = \{0, \ldots, l_N/4\}$ and $k = \lfloor l_N/4a \rfloor$ is odd. Suppose $U, V \subseteq S$ such that $U = \{0, \ldots, a\} \cup \{2a - t, \ldots, 3a\} \cup \cdots \cup \{(k-1)a - t, \ldots, ka\}$, $V = \{a - t, \ldots, 2a\} \cup \{3a - t, \ldots, 4a\} \cup \cdots \cup \{ka - t, \ldots, l_N/4\}$, where $p[i]$'s are known for $i \in U$ and $q[j]$'s are known for $j \in V$, as discussed before. Then, one can factor $N$ in poly$(\log N)$ time with probability at least* $P_{a,t} = \left(1 - \frac{1}{2^t}\right)^{\lfloor l_N/4a \rfloor}$.

*Proof.* By setting $T = l_N/4$ in Theorem 3 we obtain that Algorithm 2 is able to recover contiguous $l_N/4$ many MSBs of one of the primes $p, q$ in $O(l_N/4)$ steps with probability at least $P_{a,t} = \left(1 - \frac{1}{2^t}\right)^{\lfloor l_N/4a \rfloor}$. Since one call of CORRECT costs $O(l_N)$, thus the total time complexity is $O(\log^2 N)$.

Once we get these $l_N/4$ MSBs, that is the complete most significant half, of one of the primes, one can use the existing lattice based method [4] by Coppersmith to factor $N = pq$ in poly$(\log N)$ time. □

### 4.4 Experimental Results for the Reconstruction Algorithm

We present some experimental results in Table 3 to support the claim in Theorem 3. The blocksize for known bits, i.e, $a$, and the approximation offset $t$ are varied to obtain these results for $l_N = 1024$. The target size for reconstruction is $T = 256$ as the primes are 512 bits each. We have run the experiment 1000 times for each pair of fixed parameters $a, t$. The first value in each cell represents the experimental percentage of success in these cases and illustrate the practicality of our method. The second value in each cell is the theoretical probability of success obtained from Theorem 3.

**Table 3.** Percentage of success of Algorithm 2 with 512-bit $p, q$, i.e., $l_N = 1024$

| $a$ | $t=1$ | $t=2$ | $t=3$ | $t=4$ | $t=5$ | $t=6$ | $t=7$ | $t=8$ | $t=9$ | $t=10$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0, 0 | 2.5, 0.07 | 16.8, 3.55 | 41.5, 19.9 | 64.5, 45.2 | 82.1, 67.5 | 90.6, 82.2 | 95.0, 90.7 | 97.2, 95.2 | - |
| 20 | 1.8, 0.02 | 18.7, 3.17 | 44.5, 20.1 | 65.7, 46.1 | *81.9, 68.3* | *90.6, 82.8* | *94.8, 91.0* | *97.5, 95.4* | 98.5, 97.7 | 99.3, 97.6 |
| 40 | 15.5, 1.6 | 42.8, 17.8 | 66.7, 44.9 | *81.8, 67.9* | *90.8, 82.7* | *95.2, 91.0* | *97.8, 95.4* | *98.6, 97.7* | *99.3, 98.8* | *99.9, 99.4* |
| 60 | 29.1, 6.3 | 55.6, 31.6 | *75.7, 58.6* | *86.6, 77.2* | *91.7, 88.1* | *95.3, 93.9* | *97.4, 96.9* | *98.9, 98.4* | *99.5, 99.2* | *99.9, 99.7* |
| 80 | 41.9, 12.5 | 66.4, 42.2 | *82.9, 67.0* | *91.0, 82.4* | *95.7, 90.9* | *98.3, 95.4* | *99.1, 97.7* | *99.4, 98.8* | *99.7, 99.4* | *100, 99.7* |
| 100 | 50.6, 25.0 | *74.4, 56.2* | *86.6, 76.6* | *93.7, 87.9* | *97.1, 93.8* | *98.8, 96.9* | *99.6, 98.4* | *99.8, 99.2* | *99.9, 99.6* | *100, 99.8* |

One may note that our theoretical bounds on the probability (second value in each cell) is an underestimate compared to the experimental evidences (first value in each cell) in all the cases. This is because we have used the bound on the probability of carry loosely as $p_c \leq 1$ in Lemma 2, whereas a better estimate should have been $p_c \approx \frac{1}{2}$. As an example, let us check the case with $a = 40, t = 3$. Here, the theoretical bound on the probability with $p_c \leq 1$ is 44.9% whereas with $p_c = 0.5$, the same bound comes as 67.9%. The experimental evidence of 66.7% is quite clearly closer to the second one. But we could not correctly estimate the value of $p_c$ and hence opted for a safe (conservative) margin.

The results in *italic font* are of special interest. In these cases one can factorize $N$ in poly(log $N$)time, with probability greater than $\frac{1}{2}$ by knowing less than 70% of the bits of both the primes combined, that is, by knowing approximately just 35% of the bits of each prime $p, q$. Note that the result by Herrmann and May [6] requires about 70% of the bits of one of the primes in a similar case where the known bits are distributed over small blocks. Their result factorizes $N$ in time exponential in the number of such blocks, whereas our method produces the same result in time polynomial in the number of blocks.

## 5 Conclusion

Our work discusses the factorization of RSA modulus $N$ by reconstructing the primes from randomly known bits. The reconstruction method exploits the known bits to prune wrong branches of the search tree and reduces the total search space. We have revisited the work of Heninger and Shacham [5] in Crypto 2009 and provided a combinatorial model for the search where certain bits of the primes are known at random. This, combined with existing lattice based techniques, can factorize $N$ given the knowledge of randomly chosen prime bits in the

least significant halves of the primes. We also explain a lattice based strategy to remedy one of the shortcomings of the reconstruction algorithm. Moreover, we study how $N$ can be factored given the knowledge of some blocks of bits in the most significant halves of the primes. We propose an algorithm that recovers the most significant halves of one or both the primes exploiting the known bits. An obvious open question in this direction is to attack this problem when random MSBs (as in the case for LSBs) instead of certain blocks are available.

**Acknowledgments.** The authors are grateful to the reviewers for their invaluable comments and suggestions. The second and third authors would like to acknowledge the Council of Scientific and Industrial Research (CSIR) and the Department of Information Technology (DIT), India, for supporting their respective research. The authors are thankful to Amrita Saha of Jadavpur University for implementing the code for the LSB case in the preliminary phase of the work.

## References

1. Boneh, D.: Twenty Years of Attacks on the RSA Cryptosystem. Notices of the AMS 46(2), 203–213 (1999)
2. Boneh, D., Durfee, G., Frankel, Y.: Exposing an RSA Private Key Given a Small Fraction of its Bits. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 25–34. Springer, Heidelberg (1998)
3. Cohen, H.: A Course in Computational Algebraic Number Theory. Springer, Heidelberg (1996)
4. Coppersmith, D.: Small Solutions to Polynomial Equations and Low Exponent Vulnerabilities. Journal of Cryptology 10(4), 223–260 (1997)
5. Heninger, N., Shacham, H.: Reconstructing RSA Private Keys from Random Key Bits. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 1–17. Springer, Heidelberg (2009)
6. Herrmann, M., May, A.: Solving Linear Equations Modulo Divisors: On Factoring Given Any Bits. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 406–424. Springer, Heidelberg (2008)
7. Howgrave-Graham, N.: Finding Small Roots of Univariate Modular Equations Revisited. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997)
8. Jochemsz, E., May, A.: A Strategy for Finding Roots of Multivariate Polynomials with new Applications in Attacking RSA Variants. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 267–282. Springer, Heidelberg (2006)
9. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring Polynomials with Rational Coefficients. Mathematische Annalen 261, 513–534 (1982)
10. May, A.: Using LLL-Reduction for Solving RSA and Factorization Problems: A Survey. In: LLL+25 Conference in honour of the 25th birthday of the LLL algorithm (2007), http://www.cits.rub.de/personen/may.html
11. Rivest, R.L., Shamir, A.: Efficient Factoring based on Partial Information. In: Pichler, F. (ed.) EUROCRYPT 1985. LNCS, vol. 219, pp. 31–34. Springer, Heidelberg (1986)
12. Rivest, R.L., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of ACM 21(2), 158–164 (1978)

# Proofs of Restricted Shuffles

Björn Terelius and Douglas Wikström

CSC KTH Stockholm, Sweden
{terelius,dog}@csc.kth.se

**Abstract.** A proof of a shuffle is a zero-knowledge proof that one list of ciphertexts is a permutation and re-encryption of another list of ciphertexts. We call a shuffle restricted if the permutation is chosen from a public subset of all permutations. In this paper, we introduce a general technique for constructing proofs of shuffles which restrict the permutation to a group that is characterized by a public polynomial. This generalizes previous work by Reiter and Wang [22], and de Hoogh et al. [7].

Our approach also gives a new efficient proof of an unrestricted shuffle that we think is conceptually simpler and allow a simpler analysis than all previous proofs of shuffles.

**Keywords:** cryptographic protocols, election schemes, mix-nets, proof of a shuffle.

## 1 Introduction

*Mix-Nets.* Suppose that $N$ senders $S_1, \ldots, S_N$ each wish to send a message, but remain anonymous within the group of senders, e.g., the senders could be voters in an election. Chaum [5] introduced the notion of a mix-net (or anonymous channel) to solve this problem. A mix-net is a cryptographic protocol executed by $k$ mix-servers $M_1, \ldots, M_k$, where $k$ typically is much smaller than $N$. All provably secure mix-nets proposed in the literature take as input a list $L_0$ of ciphertexts and order the mix-servers in a chain. Each mix-server $M_j$ in the chain takes as input the output $L_{j-1}$ of the previous mix-server in the chain. It processes each ciphertext in $L_{j-1}$ by decrypting and/or re-encrypting it, and then forms its output $L_j$ as the processed ciphertexts in random order. This operation is called a *shuffle*. If the ciphertexts are not decrypted during processing, the mix-servers then perform a joint verifiable decryption of the final list $L_k$. In some applications, the final output of the mix-net may be a list of the ciphertexts themselves, in which case no joint decryption takes place.

In Chaum's original construction a generic cryptosystem is used. A sender encrypts its message $m_i$ as $\mathsf{E}_{pk_1}(\mathsf{E}_{pk_2}(\cdots \mathsf{E}_{pk_k}(m_i)\cdots))$, where $pk_j$ is the public key of the $j$th mix-server, and the mix-servers use standard decryption when processing the ciphertexts. Park et al. [18] observed that if a homomorphic cryptosystem is used, the increase in the size of the submitted ciphertext can be avoided. Another, perhaps more important, consequence of using a homomorphic cryptosystem is that it simplifies the construction of a zero-knowledge proof

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 100–113, 2010.

that a mix-server shuffles its input correctly, a so called *proof of a shuffle*. We give a detailed account of previous work on such protocols later, but first we conclude the discussion on mix-nets.

Although the mix-servers use the homomorphic properties of the cryptosystem constructively, Pfitzmann [20] points out that a non-malleable cryptosystem must be used in the submission phase. There are several ways to achieve this in a provably secure way.

The security of a mix-net as a whole was first formalized by Abe and Imai [1] in a standalone model, but they did not provide a construction that satisfied their definition. The first definition of security in the UC framework [4] and the first mix-net provably secure as a whole was given by Wikström [25]. Later work [26] gave simpler and more efficient constructions.

*Previous Work On Proofs of Shuffles.* As far as we know the first proof of a shuffle appears in Sako and Kilian [24] based on a previous protocol by Park et al. [18]. They give a cut-and-choose based zero-knowledge proof of a shuffle with soundness 1/2. Its soundness can be increased using repetition, but this becomes computationally expensive if the number of ciphertexts is large. The first efficient proofs of shuffles were given independently by Neff [17] and Furukawa and Sako [11]. Both approaches were subsequently optimized and generalized, in particular by Groth [13] and Furukawa [9] respectively. Wikström [26] presented a proof of a shuffle based on an idea distinct from both that of Neff and that of Furukawa and Sako.

These shuffles have all been optimized and/or generalized in various ways, e.g., for proving re-encryption and partial decryption shuffling [10], for shuffling hybrid ciphertexts [12], or to reduce communication complexity [15].

A different approach to mix-nets was introduced by Adida and Wikström[3,2]. They investigate to what extent a shuffle can be pre-computed in such a way that the shuffling can be done in public and the online processing of the mix-servers can be reduced to joint decryption. The construction in [2] is conceptually appealing, but inefficient, whereas the construction in [3] is very efficient as long as the number of senders is relatively small.

In a recent paper, Wikström [27] shows that any proof of a shuffle can be split in an offline part and an extremely efficient online part. The offline part is executed before, or at the same time, that senders submit their inputs and consists of a commitment scheme for permutations and a zero-knowledge proof of knowledge of correctly opening such a commitment. The online part is a commitment-consistent proof of a shuffle, where the prover shows that it correctly uses the committed permutation to process the input. This drastically reduces the online complexity of provably secure mix-nets.

Motivated by insider attacks, Reiter and Wang [22] construct a proof of a rotation (they use the term "fragile mixing"). A rotation is a shuffle, where the permutation used is restricted to a random rotation of the ciphertexts. Their idea is to reduce the incentive of insiders to reveal any input/output-correspondence, since this would reveal the complete permutation used, which is assumed to be associated with a cost for the insider. Recently, a more efficient proof of a rotation

is given by de Hoogh et al. [7]. In fact, they give two protocols: a general protocol for any homomorphic cryptosystem and rotation, and a more efficient solution which requires some mild assumptions on the homomorphic cryptosystem used.

De Hoogh et al. lists several possible applications of proofs of rotations beyond the classical application of proofs of shuffles for mix-nets, e.g., secure integer comparison [21], secure function evaluation [16], and submission schemes in electronic election schemes [23].

### 1.1 Our Contribution

We introduce a novel technique for restricting the class of permutations in a proof of a shuffle of $N$ ciphertexts by showing that $\pi$ is chosen such that $F(x_{\pi(1)}, \ldots, x_{\pi(N)}) = F(x_1, \ldots, x_N)$ for some public polynomial $F$. In particular, we can prove that the permutation is contained in the automorphism group of a (directed or undirected) graph on $N$ elements.

A concrete general proof of rotation with efficiency comparable to that of de Hoogh et al. [7] is trivially derived from our technique, but several other natural concrete examples are derived just as easily, e.g. the list of ciphertexts may be viewed as a complete binary tree and the set of permutations restricted to isomorphisms of the tree.

Furthermore, the basic principle behind our technique can be used in a natural way to construct a novel efficient proof of an *unrestricted* shuffle with an exceptionally simple analysis. Given the large and unwieldy literature on how to construct efficient proofs of shuffles we think this conceptual simplification is of independent interest.

### 1.2 Informal Description of Our Technique

We briefly describe our results and the technique we use, but before we do so we recall the definition of Pedersen's perfectly hiding commitment scheme [19], or more precisely a well known generalization thereof. The commitment parameters consist of $N+1$ randomly chosen generators $g, g_1, \ldots, g_N$ in a group $G_q$ of prime order $q$ in which the discrete logarithm assumption holds. To commit to an array $(e_1, \ldots, e_N) \in \mathbb{Z}_q^N$, the committer forms $g^s \prod_{i=1}^N g_i^{e_i}$. Below we use the fact that sigma proofs can be used to efficiently prove any polynomial relation between the values $e_1, \ldots, e_N$.

*A New Proof of a Shuffle.* We describe how to prove that a matrix $M \in \mathbb{Z}_q^{N \times N}$ over a finite field $\mathbb{Z}_q$ hidden in a Pedersen commitment is a permutation matrix. Let $\langle \cdot, \cdot \rangle$ denote the standard inner product in $\mathbb{Z}_q^N$ and let $\overline{x} = (x_1, \ldots, x_N)$ be a list of variables. If $M$ does not have exactly one non-zero element in each column and each row, then $\prod_{i=1}^N \langle \overline{m}_i, \overline{x} \rangle \neq \prod_{i=1}^N x_i$ where $\overline{m}_i$ denotes the $i$th row of $M$. This can be tested efficiently by Schwarz-Zippel's lemma, which we recall states that if $f(x_1, \ldots, x_N)$ is a non-zero polynomial of degree $d$ and we pick a point $\overline{e}$ from $\mathbb{Z}_q^N$ randomly, then the probability that $f(\overline{e}) = 0$ is at most $d/q$.

To prove that $M$ is a permutation matrix, given that it has exactly one non-zero element in each row and column, is of course trivial; simply show that the elements of each row sum to one.

We observe that proving both these properties can be done efficiently using the matrix commitment scheme used in [11,27]. The commitment parameters of this scheme consists of independently chosen generators $g, g_1, \ldots, g_N$ of a group $G_q$ of prime order $q$. To commit to a matrix $M$, the committer computes $(a_1, \ldots, a_N) = (g^{s_1} \prod_{i=1}^{N} g_i^{m_{i,1}}, \ldots, g^{s_N} \prod_{i=1}^{N} g_i^{m_{i,N}})$. which allows publicly computing a commitment of all $\langle \overline{m}_i, \overline{e} \rangle$ as $\prod_{j=1}^{N} a_j^{e_j} = g^{\langle s, \overline{e} \rangle} \prod_{j=1}^{N} g_i^{\langle \overline{m}_i, \overline{e} \rangle}$. The prover may then use standard sigma proofs to show that $\prod_{i=1}^{N} \langle \overline{m}_i, \overline{e} \rangle = \prod_{i=1}^{N} e_i$. To show that the sum of each row is one it suffices to prove that $\prod_{j=1}^{N} a_j / \prod_{j=1}^{N} g_j$ is on the form $g^s$ for some $s$.

At this point we may invoke the commitment-consistent proof of a shuffle in [27] to extend the above to a complete proof of an unrestricted shuffle, but we also present a more direct construction.

*Restricting the Set of Permutations.* In the interest of describing the techniques, we consider how to restrict the set of permutations to the automorphism group of an undirected graph $\mathscr{G}$. Let the graph have vertices $V = \{1, 2, 3, \ldots, N\}$ and edges $E \subseteq V \times V$ and encode the graph as a polynomial $F_{\mathscr{G}}(x_1, \ldots, x_N) = \sum_{(i,j) \in E} x_i x_j$ in $\mathbb{Z}_q[x_1, \ldots, x_N]$. Observe that $\pi$ is contained in the automorphism group of $\mathscr{G}$ if and only if $F_{\mathscr{G}}(x_{\pi(1)}, \ldots, x_{\pi(N)}) = F_{\mathscr{G}}(x_1, \ldots, x_N)$.

Suppose that a prover has shown that a committed matrix $M \in \mathbb{Z}_q^{N \times N}$ is a permutation matrix corresponding to a permutation $\pi$. If $\pi$ is not contained in the automorphism group of $\mathscr{G}$, it follows from Schwartz-Zippel's lemma that $\Pr\left[F_{\mathscr{G}}(e_{\pi(1)}, \ldots, e_{\pi(N)}) = F_{\mathscr{G}}(e_1, \ldots, e_N)\right]$ is exponentially small, when $\overline{e} \in \mathbb{Z}_q$ is randomly chosen by the verifier. Since $M\overline{e} = (e_{\pi(1)}, \ldots, e_{\pi(N)})$ the verifier can easily compute a commitment of the permuted random exponents as $\prod_{j=1}^{N} a_j^{e_j} = g^{\langle \overline{s}, \overline{e} \rangle} \prod_{j=1}^{N} g_i^{e_{\pi(i)}}$. The prover can then use a standard sigma proof to prove the equality, for example by proving correct computation of each gate in the arithmetic circuit for the polynomial.

Note that it is not important that the polynomial comes from a graph. Hence, we can apply the same technique to prove that $\pi$ satisfies $F(e_{\pi(1)}, \ldots, e_{\pi(N)}) = F(e_1, \ldots, e_N)$ for any public polynomial $F$.

## 2 Notation and Tools

We use $n$ as the security parameter and let $q$ denote an $n$-bit prime integer. The field with $q$ elements is denoted by $\mathbb{Z}_q$. Our protocols are employed in a group $G_q$ of prime order $q$ with standard generator $g$, in which the discrete logarithm problem is assumed to be hard. We use bars over vectors to distinguish them from scalars. The angle bracket $\langle \overline{a}, \overline{b} \rangle$ denotes the inner product $\sum_{i=1}^{N} a_i b_i$ of two vectors $\overline{a}, \overline{b} \in \mathbb{Z}_q^N$. For a list $u = (u_1, \ldots, u_N) \in G_q^N$ and a vector $\overline{e} \in \mathbb{Z}_q^N$ we abuse notation and write $u^{\overline{e}} = \prod_{i=1}^{N} u_i^{e_i}$. Throughout, we use $S \subseteq \mathbb{Z}_q$ to denote the set from which the components of our random vectors are chosen.

If $\mathscr{R}_1$ and $\mathscr{R}_2$ are relations we denote by $\mathscr{R}_1 \vee \mathscr{R}_2$ the relation consisting of pairs $((x_1, x_2), w)$ such that $(x_1, w) \in \mathscr{R}_1$ or $(x_2, w) \in \mathscr{R}_2$.

*Matrix Commitments.* We use a variation of Pedersen commitments [19] in a group $G_q$, to commit to a matrix over $\mathbb{Z}_q$. The commitment parameter $ck$ needed to commit to an $N \times 1$ matrix consists of a description of the group $G_q$ with its standard generator $g$ and randomly chosen group elements $g_1, \ldots, g_N \in G_q$. An $N \times 1$-matrix $M$ over $\mathbb{Z}_q$ is committed to by computing $a = \mathcal{C}_{ck}(M, s) = g^s \prod_{i=1}^{N} g_i^{m_i}$, where $s \in \mathbb{Z}_q$ is chosen randomly. We abuse notation and omit the commitment parameter when it is clear from the context. An $N \times N$-matrix $M$ is committed to column-wise, i.e., $\mathcal{C}(M, \overline{s}) = \left(\mathcal{C}\left((m_{i,1})_{i=1}^{N}, s_1\right), \ldots, \mathcal{C}\left((m_{i,N})_{i=1}^{N}, s_N\right)\right)$, where in this case $\overline{s}$ is chosen randomly in $\mathbb{Z}_q^N$. By committing to a matrix $M$ column-wise we get the useful identity

$$\mathcal{C}(M, \overline{s})^{\overline{e}} = \prod_{j=1}^{N} g^{s_j e_j} \prod_{i=1}^{N} g_i^{m_{i,j} e_j} = g^{\langle \overline{s}, \overline{e} \rangle} \prod_{i=1}^{N} g_i^{\sum_{j=1}^{N} m_{i,j} e_j} = \mathcal{C}(M\overline{e}, \langle \overline{s}, \overline{e} \rangle) \quad .$$

This feature plays a central role in our approach. It is easy to see that the commitment scheme is perfectly hiding. The binding property is known to hold under the discrete logarithm assumption in $G_q$, see [11] for a proof.

We construct protocols that are sound under the assumption that the binding property of the above protocol is not violated. We define $\mathscr{R}_{com}$ to be the relation consisting of pairs $((ck, a), (M, \overline{s}, M', \overline{s}'))$ such that $a = \mathcal{C}_{ck}(M, \overline{s}) = \mathcal{C}_{ck}(M', \overline{s}')$, i.e., finding a witness corresponds to violating the binding property of the commitment scheme.

*$\Sigma$-proofs.* Recall that a sigma proof is a three-message protocol that is both special sound and special honest verifier zero-knowledge [6]. The first property means that the view of the verifier can be simulated for a given challenge, and the second property means that a witness can be computed from any pair of accepting transcripts with identical first messages and distinct challenges. It is well known that if a prover $\mathcal{P}^*$ convinces the verifier with probability $\delta$, there exists an extractor running in expected time $\mathcal{O}\big(T/(\delta - \epsilon)\big)$ for some polynomial $T$ and some negligible knowledge error $\epsilon$. Given a statement $x$, we denote the execution of a sigma proof of knowledge of a witness $w$ such that $(x, w) \in \mathscr{R}$ by $\Sigma\text{-}\mathsf{proof}\left[w \,|\, (x, w) \in \mathscr{R}\right]$.

We need to prove knowledge of how to open commitments such that the committed values satisfy a public polynomial relation, i.e. to construct a sigma proof

$$\Sigma\text{-}\mathsf{proof}\left[\overline{e} \in \mathbb{Z}_q^N, s \in \mathbb{Z}_q \,\middle|\, a = \mathcal{C}(\overline{e}, s) \wedge f(\overline{e}) = e'\right]$$

given a commitment $a = \mathcal{C}(\overline{e}, s)$, a polynomial $f \in \mathbb{Z}_q[x_1, \ldots, x_N]$, and a value $e' \in \mathbb{Z}_q$. We remind the reader how this can be done.

The parties agree on an arithmetic circuit over $\mathbb{Z}_q$ that evaluates the polynomial $f$. The prover constructs new commitments $a_i$ to each individual value $e_i$

hidden in $a$ and proves that it knows how to open all commitments consistently with a proof of the form

$$\Sigma\text{-proof}\left[\overline{e} \in \mathbb{Z}_q^N, s, s_1, \ldots, s_N \in \mathbb{Z}_q \,\middle|\, a = \mathcal{C}\left(\overline{e}, s\right) \wedge \forall_{i=1}^{N}(a_i = \mathcal{C}\left(e_i, s_i\right))\right] \quad .$$

The resulting commitments $a_1, \ldots, a_N$ are considered the input of the arithmetic circuit. For each summation gate, the two input commitments of the gate are multiplied to form the output of the gate. For each product gate with input commitments $a_1 = \mathcal{C}\left(e_1, s_1\right)$ and $a_2 = \mathcal{C}\left(e_2, s_2\right)$, the prover forms a new commitment $a_3 = \mathcal{C}\left(e_3, s_3\right)$ and proves that $e_3 = e_1 e_2$ with a sigma protocol

$$\Sigma\text{-proof}\left[e_2, s_2, s_3' \in \mathbb{Z}_q \,\middle|\, a_3 = g^{s_3'} a_1^{e_2} \wedge a_2 = \mathcal{C}\left(e_2, s_2\right)\right] \quad .$$

Finally, the output $a$ of the entire circuit is shown to be a commitment of $e'$ using a protocol of the form

$$\Sigma\text{-proof}\left[s \in \mathbb{Z}_q \,\middle|\, a/g_1^{e'} = g^s\right] \quad .$$

Special soundness and special honest verifier zero-knowledge allow us to execute all these protocols in parallel using a single challenge from the verifier, thus forming a new sigma protocol. Together with the binding property of the commitment scheme, this implies that the prover knows $\overline{e} \in \mathbb{Z}_q^N$ and $s \in \mathbb{Z}_q$ such that $\mathcal{C}\left(\overline{e}, s\right) = a \wedge f(\overline{e}) = e'$.

We remark that it is sometimes possible to do better than the general technique above. In particular when proving that a shuffle is a rotation, one has to prove that a polynomial of the form $\sum_{i=1}^{N} x_i y_i$ has a certain value. This can be done efficiently by evaluating the polynomial as an inner product between the vectors $\overline{x}$ and $\overline{y}$ using a linear algebra protocol from [14].

*Polynomial Equivalence Testing.* We use the Schwartz-Zippel lemma to analyze the soundness of our protocols. The lemma gives an efficient, probabilistic test of whether a polynomial is identically zero.

**Lemma 1 (Schwartz-Zippel).** *Let $f \in \mathbb{Z}_q[x_1, \ldots, x_N]$ be a non-zero multivariate polynomial of total degree $d \geq 0$ over $\mathbb{Z}_q$, let $S \subseteq \mathbb{Z}_q$, and let $e_1, \ldots, e_N$ be chosen randomly from $S$. Then*

$$\Pr\left[f(e_1, \ldots, e_N) = 0\right] \leq \frac{d}{|S|} \quad .$$

## 3 Proof of Knowledge of Permutation Matrix

We show how to prove knowledge of how to open a Pedersen commitment of a matrix such that the matrix is a permutation matrix. Wikström [27] constructs a commitment-consistent proof of a shuffle for any shuffle-friendly map, based on the same permutation commitment we use here. Thus, it is trivial to construct

a proof of a shuffle by combining the protocol below with the online protocol in [27].

Our protocol is based on a simple probabilistic test that accepts a non-permutation matrix with negligible probability.

**Theorem 1 (Permutation Matrix).** *Let $M = (m_{i,j})$ be an $N \times N$-matrix over $\mathbb{Z}_q$ and $\overline{x} = (x_1, \ldots, x_N)$ a vector of $N$ independent variables. Then $M$ is a permutation matrix if and only if $\prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle = \prod_{i=1}^{N} x_i$ and $M\overline{1} = \overline{1}$.*

*Proof.* Consider the polynomial $f(\overline{x}) = \prod_{i=1}^{N} \langle \overline{m}_i, \overline{x} \rangle$ in the multivariate polynomial ring $R = \mathbb{Z}_q[x_1, \ldots, x_N]$, where $\overline{m}_i$ is the $i$th row of $M$. It is clear that $M\overline{1} = \overline{1}$ and $f(\overline{x}) = \prod_{i=1}^{N} x_i$ if $M$ is a permutation matrix. Conversely, suppose that $M\overline{1} = \overline{1}$ and $f(\overline{x}) = \prod_{i=1}^{N} x_i$. If any row $\overline{m}_i$ were the zero vector, then $f$ would be the zero polynomial. If all rows of $M$ were non-zero, but some row $\overline{m}_i$ contained more than one non-zero element, then $f$ would contain a factor of the form $\sum_{j \in J} m_{i,j} x_j$ with $|J| \geq 2$ and $m_{i,j} \neq 0$ for $j \in J$. Since $R$ is a unique factorization domain, this contradicts the assumption that $f(\overline{x}) = \prod_{i=1}^{N} x_i$. If the $j$th column contained more than one non-zero element, then $\deg_{x_j} f \geq 2$, again contradicting $f = \prod_{j=1}^{N} x_j$. Thus there is exactly one non-zero element in each row and column and since $M\overline{1} = \overline{1}$, the non-zero element must equal one.

**Protocol 1 (Permutation Matrix).**
Common Input: Matrix commitment $a \in G_q^N$ and commitment parameters $g, g_1, \ldots, g_N \in G_q$.
Private Input: Permutation matrix $M \in \mathbb{Z}_q^{N \times N}$ and randomness $\overline{s} \in \mathbb{Z}_q^N$ such that $a = \mathcal{C}(M, \overline{s})$.

1. $\mathcal{V}$ chooses $\overline{e} \in S^N \subseteq \mathbb{Z}_q^N$ randomly and hands $\overline{e}$ to $\mathcal{P}$.
2. $\mathcal{P}$ defines $t = \langle \overline{1}, \overline{s} \rangle$ and $k = \langle \overline{s}, \overline{e} \rangle$. Then $\mathcal{V}$ outputs the result of

$$\Sigma\text{-proof}\left[ \begin{matrix} t, k \in \mathbb{Z}_q \\ \overline{e}' \in \mathbb{Z}_q^N \end{matrix} \middle| \mathcal{C}(\overline{1}, t) = a^{\overline{1}} \wedge \mathcal{C}(\overline{e}', k) = a^{\overline{e}} \wedge \prod_{i=1}^{N} e'_i = \prod_{i=1}^{N} e_i \right] .$$

We remark that $\mathcal{V}$ could instead hand a random seed to $\mathcal{P}$ and define $\overline{e}$ as the output of a PRG invoked on the seed. This reduces the amount of randomness needed by the verifier, which is important in applications where the role of $\mathcal{V}$ is played by a multiparty coin-flipping protocol. Since this trick is well known (cf. [27]) and complicates the exposition, we do not detail it here.

**Proposition 1.** *Protocol 1 is a perfectly complete, 4-message honest verifier zero-knowledge proof of knowledge of the relation $\mathscr{R}_\pi \vee \mathscr{R}_{com}$, where the relation $\mathscr{R}_\pi$ consists of pairs $((ck, a), (M, \overline{s}))$ such that $M$ is a permutation matrix and $a = \mathcal{C}_{ck}(M, \overline{s})$.*

Under the discrete logarithm assumption, it is infeasible to find a witness of the relation $\mathscr{R}_{com}$ for the commitment parameter $(g, g_1, \ldots, g_N)$. Thus, for a randomly chosen commitment parameter, the proposition may be interpreted as a proof of knowledge of the relation $\mathscr{R}_\pi$.

### 3.1 Proof of Proposition 1

The completeness and zero-knowledge properties follows from the completeness and zero-knowledge properties of the sigma protocol and the hiding property of the commitment scheme. What remains is to show that the protocol is a proof of knowledge by creating an extractor. We do this by extracting witnesses $(\overline{e}', t, k)$ from the sigma proof for $N$ linearly independent vectors $\overline{e}$ and use them to recover the matrix $M$. Finally, we show that if $M$ is not a permutation matrix, then we are able to extract a witness of the commitment relation $\mathscr{R}_{com}$.

*Three-Message Protocol.* We first make a conceptual change that allows us to view our four-round prover as a particular three-round prover of a standard sigma-protocol. Given a prover $\mathcal{P}^*$, we denote by $\mathcal{P}_+$ the interactive machine that chooses $\overline{e} \in \mathbb{Z}_q^N$ randomly itself instead of letting $\mathcal{V}$ choose it, and then simulates $\mathcal{P}^*$. We denote by $\mathcal{V}_+$ the corresponding verifier that accepts $\overline{e}$ as part of the first message in the sigma proof.

*Basic Extractor.* We augment the common input with a list of linearly independent vectors $\overline{e}_1, \ldots, \overline{e}_l \in \mathbb{Z}_q^N$ where $l < N$, let $\mathcal{P}_\perp$ be identical to $\mathcal{P}_+$, and define $\mathcal{V}_\perp$ to be identical to $\mathcal{V}_+$ except that it only accepts if $\overline{e}$ is linearly independent of these. If $\mathcal{P}^*$ convinces $\mathcal{V}$ with probability $\delta$, then $\mathcal{P}_\perp$ clearly convinces $\mathcal{V}_\perp$ with probability at least $\delta - \frac{1}{|S|}$, since the probability that $\overline{e}$ is linearly dependent of $\overline{e}_1, \ldots, \overline{e}_l \in \mathbb{Z}_q^N$ is bounded by $\frac{1}{|S|}$.

It is well known that the sigma proof has an extractor $\mathcal{E}_\perp$ running $\mathcal{P}_\perp$ as a black-box that given linearly independent $\overline{e}_1, \ldots, \overline{e}_l \in \mathbb{Z}_q^N$ extracts an $\overline{e}$ that is linearly independent of the former vectors and a corresponding witness $(\overline{e}', t, k)$ of the sigma proof. Furthermore, $\mathcal{E}_\perp$ runs in expected time $T/(\delta - \frac{1}{|S|} - \epsilon)$ for some polynomial $T(n)$ in the security parameter $n$, where $\epsilon$ is the knowledge error of the sigma proof. Denote by $\mathcal{E}_N$ the extractor that computes witnesses $(\overline{e}_l, t_l, \overline{e}'_l, k_l) = \mathcal{E}_\perp(\overline{e}_1, \ldots, \overline{e}_{l-1}, a, g, g_1, \ldots, g_N)$ for $l = 1, \ldots, N$. Then $\mathcal{E}_N$ runs in expected time $\mathcal{O}\big(NT/(\delta - \frac{1}{|S|} - \epsilon)\big)$.

*Computation of Committed Matrix.* From linear independence follows that there exists $\alpha_{l,j} \in \mathbb{Z}_q$ such that $\sum_{j=1}^N \alpha_{l,j}\overline{e}_j$ is the $l$th standard unit vector in $\mathbb{Z}_q^N$. We conclude that:

$$a_l = \prod_{j=1}^N a^{\alpha_{l,j}\overline{e}_j} = \prod_{j=1}^N \mathcal{C}\left(\overline{e}'_j, k_j\right)^{\alpha_{l,j}} = \mathcal{C}\left(\sum_{j=1}^N \alpha_{l,j}\overline{e}'_j, \sum_{j=1}^N \alpha_{l,j}k_j\right) \quad .$$

Thus, we have $a = \mathcal{C}(M, \overline{s})$, where $\sum_{j=1}^N \alpha_{l,j}\overline{e}'_j$ is the $l$th column of a matrix $M \in \mathbb{Z}_q^{N\times N}$ and $\overline{s} = \left(\sum_{j=1}^N \alpha_{1,j}k_j, \ldots, \sum_{j=1}^N \alpha_{N,j}k_j\right) \in \mathbb{Z}_q^N$ is the random vector used to commit.

*Product Inequality Extractor.* We expect that the matrix $M$ is a permutation matrix, but if this is not the case we must argue that we can find a non-trivial representation of 1 in $G_q$. We augment the original input with a non-permutation

matrix $M$ which we assume satisfy $M\overline{1} = \overline{1}$. We will see that had $M\overline{1} \neq \overline{1}$, we would have been able to extract a witness of $\mathscr{R}_{com}$. Let $\mathcal{P}_\pi$ be identical to $\mathcal{P}_+$, and define $\mathcal{V}_\pi$ to be identical to $\mathcal{V}_+$ except that it only accepts if

$$\prod_{i=1}^{N} \langle \overline{m}_i, \overline{e} \rangle \neq \prod_{i=1}^{N} e_i \ . \tag{1}$$

From Theorem 1 and Schwartz-Zippel's lemma follows that the probability that the additional requirement is not satisfied is at most $N/|S|$. Thus, if $\mathcal{P}^*$ convinces $\mathcal{V}$ with probability $\delta$, then $\mathcal{P}_\pi$ convinces $\mathcal{V}_\pi$ with probability at least $\delta - N/|S|$. Again, a standard argument implies that there exists an extractor $\mathcal{E}_\pi$ with black-box access to $\mathcal{P}^*$ running in time $\mathcal{O}\big(T'/(\delta - \frac{N}{|S|} - \epsilon)\big)$, which extracts $(\overline{e}, t, \overline{e}', k)$ such that $\mathcal{C}(\overline{e}', k) = a^{\overline{e}}$ and Equation (1) is satisfied.

*Main Extractor.* Denote by $\mathcal{E}$ the extractor that proceeds as follows:

1. It invokes $\mathcal{E}_N$ to find a matrix $M$ and randomness $\overline{s}$ such that $a = \mathcal{C}(M, \overline{s})$. If $M$ is a permutation matrix, then $\mathcal{E}$ has found the witness $(M, \overline{s})$ of relation $\mathscr{R}_\pi$.
2. If $M$ does not satisfy $M\overline{1} = \overline{1}$, then set $\overline{e}'' = M\overline{1}$ and note that

   $$\overline{e}'' \neq \overline{1} \quad \text{and} \quad \mathcal{C}\left(\overline{1}, t_1\right) = a^{\overline{1}} = \mathcal{C}\left(\overline{e}'', \langle \overline{s}, \overline{1} \rangle\right) \ .$$

   Then $\mathcal{E}$ has found the witness $(a^{\overline{1}}, \overline{1}, t_1, \overline{e}'', \langle \overline{s}, \overline{1} \rangle)$ of the commitment relation $\mathscr{R}_{com}$.
3. If $M$ satisfies $M\overline{1} = \overline{1}$, but is not a permutation matrix, then $\mathcal{E}$ invokes $\mathcal{E}_\pi$ with the additional input $M$ to find $(\overline{e}, t, \overline{e}', k)$ such that $\mathcal{C}(\overline{e}', k) = a^{\overline{e}}$ and Equation (1) holds. Define $\overline{e}'' = M\overline{e}$ and note that

   $$\overline{e}'' \neq \overline{e}' \quad \text{and} \quad \mathcal{C}(\overline{e}', k) = a^{\overline{e}} = \mathcal{C}(\overline{e}'', \langle \overline{s}, \overline{e} \rangle) \ .$$

   The former holds, since $\prod_{i=1}^{N} e'_i = \prod_{i=1}^{N} e_i \neq \prod_{i=1}^{N} e''_i$. Then $\mathcal{E}$ has found the witness $(a^{\overline{e}}, \overline{e}', k, \overline{e}'', \langle \overline{s}, \overline{e} \rangle)$ of the commitment relation $\mathscr{R}_{com}$.

Note that the the expected running time of the extractor $\mathcal{E}$ is bounded by $\mathcal{O}\big((NT + T')/(\delta - \frac{N}{|S|} - \epsilon)\big)$ as required and that it always finds a witness of either $\mathscr{R}_\pi$ or $\mathscr{R}_{com}$. □

## 4 Proof of Knowledge of Restricted Permutation Matrix

We now detail how one can restrict $\pi$ to the subset $S_F$ of permutations that satisfies $F(\overline{x}'_1, \ldots, \overline{x}'_d) = F(\overline{x}_1, \ldots, \overline{x}_d)$ for a multivariate polynomial $F(\overline{x}_1, \ldots, \overline{x}_d)$ in $\mathbb{Z}_q[\overline{x}_1, \ldots, \overline{x}_d]$ where $\overline{x}'_i = (x_{i,\pi(1)}, \ldots, x_{i,\pi(N)})$. We remark that $d = 1$ in many instances, in which case the polynomial $F$ will just depend on a single list of $N$ variables.

The protocol below is an extension of Protocol 1. Thus, to simplify the exposition we denote by $\mathsf{P}_\pi(a, t, \overline{e}, \overline{e}', k)$ the predicate used to define the sigma proof of Protocol 1, i.e., $\mathcal{C}\left(\overline{1}, t\right) = a^{\overline{1}} \wedge \mathcal{C}(\overline{e}', k) = a^{\overline{e}} \wedge \prod_{i=1}^{N} e'_i = \prod_{i=1}^{N} e_i$.

**Protocol 2 (Restricted Permutation Matrix).**
COMMON INPUT: Matrix commitment $a \in G_q^N$, commitment parameters $g, g_1, \ldots, g_N \in G_q$, and a polynomial invariant $F$.
PRIVATE INPUT: Permutation matrix $M \in \mathbb{Z}_q^{N \times N}$ for a permutation $\pi \in S_F$ and randomness $\overline{s} \in \mathbb{Z}_q^N$ such that $a = \mathcal{C}(M, \overline{s})$.

1. $\mathcal{V}$ chooses $\overline{e}_1, \ldots, \overline{e}_d \in S^N \subseteq \mathbb{Z}_q^N$ randomly and hands $\overline{e}_1, \ldots, \overline{e}_d$ to $\mathcal{P}$.
2. $\mathcal{P}$ defines $t = \langle \overline{1}, \overline{s} \rangle$ and $k_\iota = \langle \overline{s}, \overline{e}_\iota \rangle$ for $\iota = 1, \ldots, d$. Then $\mathcal{V}$ outputs the result of

$$\Sigma\text{-proof}\left[\begin{array}{c|c} \overline{e}'_1 \in \mathbb{Z}_q^N, t, k_1 \in \mathbb{Z}_q & \mathsf{P}_\pi(a, t, \overline{e}_1, \overline{e}'_1, k_1) = 1 \\ \overline{e}'_2, \ldots, \overline{e}'_d \in \mathbb{Z}_q^N & \bigwedge_{j=1}^{d} \mathcal{C}\left(\overline{e}'_j, k_j\right) = a^{\overline{e}_j} \\ k_2, \ldots, k_d \in \mathbb{Z}_q & \bigwedge F(\overline{e}'_1, \ldots, \overline{e}'_d) = F(\overline{e}_1, \ldots, \overline{e}_d) \end{array}\right].$$

**Proposition 2.** *Protocol 2 is a perfectly complete 4-message honest verifier zero-knowledge proof of knowledge of the relation $\mathscr{R}_{\mathscr{G}} \vee \mathscr{R}_{com}$, where the relation $\mathscr{R}_{\mathscr{G}}$ consists of pairs $((ck, a, F), (M, \overline{s}))$ such that $M$ is a permutation matrix of $\pi \in S_F$ and $a = \mathcal{C}_{ck}(M, \overline{s})$.*

The proof, given in the full version, is a slight modification of the proof of Proposition 1.

### 4.1 Encoding Graphs as Polynomials

So far, we have not discussed where the polynomials would come from. In this section we describe how a graph can be encoded as a polynomial which is invariant under automorphisms of the graph.

The edge set $E$ of an undirected graph $\mathscr{G}$ with $N$ vertices can be encoded by the polynomial $F_{\mathscr{G}}(\overline{x}) = \sum_{(i,j)\in E} x_i x_j$ where $\overline{x}$ is a list of $N$ independent variables. This encoding is generalized in the natural way to a hypergraph with edge set $E$ by defining $F_{\mathscr{G}}(\overline{x}) = \sum_{e\in E} \prod_{i\in e} x_i$. Both encodings allow multiple edges and self-loops.

Notice that the encoding above does not preserve information about the direction of the edges. For directed graphs, we instead introduce new variables $y_1, \ldots, y_N$ and use the polynomial $F_{\mathscr{G}}(\overline{x}, \overline{y}) = \sum_{(i,j)\in E} x_i y_j$, where $x_i$ and $y_i$ represent the origin and destination of a directed edge from and to vertex $i$ respectively. For example, the cyclic group $C_N$ of rotations of $N$ elements arise as the automorphism group of the directed cyclic graph on $N$ vertices. This graph can be encoded by the polynomial $F_{\mathscr{G}}(\overline{x}, \overline{y}) = \sum_{(i,j)\in E} x_i y_j$ so we can use a proof of a restricted shuffle to show that one list of ciphertexts is a rotation of another.

This trick of adding more variables can be generalized to encode the order of the vertices in the edges of a hypergraph.

**Theorem 2.** *Let $F_{\mathscr{G}}(\overline{x}_1, \ldots, \overline{x}_d)$ be the encoding polynomial of a (directed or undirected) graph or hypergraph $\mathscr{G}$. A permutation $\pi$ is an automorphism of $\mathscr{G}$ if and only if*

$$F_{\mathscr{G}}(\overline{x}_1, \ldots, \overline{x}_d) = F_{\mathscr{G}}(\overline{x}'_1, \ldots, \overline{x}'_d) \ ,$$

*where* $\overline{x}'_i = (x_{i,\pi(1)}, \ldots, x_{i,\pi(N)})$.

*Proof.* Recall that an automorphism is a permutation of the vertices which maps edges to edges. Since $F_{\mathscr{G}}$ is an encoding of the edge set and the edge sets are equal if and only if the permutation is an automorphism, it follows that the encoding polynomials are equal if and only if the permutation is an automorphism. □

## 5 Proofs of Restricted Shuffles

We immediately get an 8-message proof of a restricted shuffle for any shuffle-friendly map and any homomorphic cryptosystem by combining our result with the protocol in [27]. Here we give a 5-message proof of a restricted shuffle for the important special case where each element in the groups of ciphertexts and randomness have prime order $q$, e.g. El Gamal [8].

Recall the definition of shuffle-friendly maps from [27], where we use $\mathcal{C}_{pk}$ and $\mathcal{R}_{pk}$ to denote the groups of ciphertexts and randomness for a public key $pk$.

**Definition 1.** *A map $\phi_{pk}$ is shuffle-friendly for a public key $pk \in \mathcal{PK}$ of a homomorphic cryptosystem if it defines a homomorphic map $\phi_{pk} : \mathcal{C}_{pk} \times \mathcal{R}_{pk} \to \mathcal{C}_{pk}$.*

For example, the shuffle-friendly map[1] of a shuffle where the ciphertexts are re-encrypted and permuted is defined by $\phi_{pk}(c, r) = c \cdot \mathsf{E}_{pk}(1, r)$. All the known shuffles of homomorphic ciphertexts or lists of ciphertexts can be expressed similarly (see [27] for more examples).We let $\mathsf{P}_F(a, t, \{\overline{e}_\iota, k_\iota, \overline{e}'_\iota\}_{\iota=1}^d)$ denote the predicate $\mathsf{P}_\pi(a, t, \overline{e}_1, \overline{e}'_1, k_1) \wedge F(\overline{e}'_1, \ldots, \overline{e}'_d) = F(\overline{e}_1, \ldots, \overline{e}_d) \wedge \mathcal{C}(\overline{e}'_j, k_j) = a^{\overline{e}_j}$ for all $j$, i.e., the predicate that was used to define the sigma proof in Protocol 2.

**Protocol 3 (Proof of Restricted Shuffle).**
Common Input: Commitment parameters $g, g_1, \ldots, g_N \in G_q$, a polynomial $F$, public key $pk$, and ciphertexts $c_1, \ldots, c_N, c'_1, \ldots, c'_N \in \mathcal{C}_{pk}$.
Private Input: A permutation $\pi \in S_F$ and randomness $\overline{r} \in \mathcal{R}_{pk}^N$ such that $c'_i = \phi_{pk}\left(c_{\pi(i)}, r_{\pi(i)}\right)$.

1. Let $M \in \mathbb{Z}_q^{N \times N}$ be the permutation matrix representing $\pi$. $\mathcal{P}$ chooses $\overline{s} \in \mathbb{Z}_q^N$ randomly, computes $a = \mathcal{C}(M, \overline{s})$, and hands $a$ to $\mathcal{V}$.
2. $\mathcal{V}$ chooses $\overline{e}_1, \ldots, \overline{e}_d \in S^N \subseteq \mathbb{Z}_q^N$ randomly and hands $\overline{e}_1, \ldots, \overline{e}_d$ to $\mathcal{P}$.
3. $\mathcal{P}$ defines $\overline{e}'_\iota = M\overline{e}_\iota$, $t = \langle \overline{1}, \overline{s} \rangle$, $k_\iota = \langle \overline{s}, \overline{e}_\iota \rangle$ for $\iota = 1, \ldots, d$, and $u = \langle \overline{r}, \overline{e}_1 \rangle$. Then $\mathcal{V}$ outputs the result of

$$\Sigma\text{-proof}\left[ \begin{array}{c} \{\overline{e}'_\iota \in \mathbb{Z}_q^N, k_\iota \in \mathbb{Z}_q\}_{\iota=1}^d \\ t \in \mathbb{Z}_q, u \in \mathcal{R}_{pk} \end{array} \middle| \begin{array}{c} \mathsf{P}_F(a, t, \{\overline{e}_\iota, k_\iota, \overline{e}'_\iota\}_{\iota=1}^d) = 1 \\ \prod_{i=1}^N (c'_i)^{e'_{1,i}} = \phi_{pk}\left(\prod_{i=1}^N c_i^{e_{1,i}}, u\right) \end{array} \right] \ .$$

[1] We remark that nothing prevents proving a shuffle of other objects than ciphertexts, i.e., any groups $\mathcal{C}_{pk}$ and $\mathcal{R}_{pk}$ of prime order $q$, and any homomorphic map $\phi_{pk}$ defined by some public parameter $pk$ can be used.

In the full version we give a concrete instantiation of the above sigma proof for an unrestricted shuffle which has efficiency comparable to some of the most efficient proofs of a shuffle in the literature. We also explain how a shuffle of a complete binary tree can be derived with essentially no additional computational cost.

**Proposition 3.** *Protocol 3 is a perfectly complete 5-message honest verifier zero-knowledge proof of knowledge of the relation* $\mathscr{R}_{\phi_{pk}} \vee \mathscr{R}_{com}$*, where* $\mathscr{R}_{\phi_{pk}}$ *consists of pairs* $((ck, a, F, pk, \overline{c}, \overline{c}'), (M, \overline{s}, \overline{r}))$ *such that* $M$ *is a permutation matrix of* $\pi \in S_F$*,* $a = \mathcal{C}_{ck}(M, \overline{s})$ *and* $c'_i = \phi_{pk}\left(c_{\pi(i)}, r_{\pi(i)}\right)$*.*

A proof of the proposition is given in the full version.

## 6 Variations and Generalizations

There are many natural variations and generalizations of our approach. Below we briefly mention some of these.

An alternative encoding of a graph is found by switching the roles of multiplication and addition in the encoding polynomial, e.g., the encoding polynomial of an undirected graph $\mathscr{G}$ could be defined as $F_{\mathscr{G}}(x_1, \ldots, x_N) = \prod_{(i,j)\in E}(x_i + x_j)$. Direction of edges can also be represented in an alternative way using powers to distinguish the ends of each edge, e.g, given a directed graph $\mathscr{G}$ the encoding polynomial could be defined by $F_{\mathscr{G}}(x_1, \ldots, x_N) = \sum_{(i,j)\in E} x_i x_j^2$. We can combine these ideas, turning exponentiation into multiplication by a scalar, and get the encoding polynomial $F_{\mathscr{G}}(x_1, \ldots, x_N) = \prod_{(i,j)\in E}(x_i + 2x_j + 1)$ for our directed graph. The additive constant 1 is needed to fix the scalar multiple of each factor since factorization in the ring $\mathbb{Z}_q[x_1, \ldots, x_N]$ is only unique up to units. The same ideas can be extended to hypergraphs and oriented hypergraphs, i.e. hypergraphs where the edges are ordered tuples rather than sets of vertices.

It is easy to generalize the protocol to proving that $f(x_{\pi(1)}, ...x_{\pi(N)}) = g(x_1, ...x_N)$ for an arbitrary function $g$. In the body of the paper, we dealt with the important special case where $f = g$, but by choosing $g$ different from $f$, it is possible to prove that the permutation belong to a set that is not necessarily a group. For example, one can prove that the permutation is odd by choosing

$$f(x_1, ...x_N) = \prod_{i<j}(x_i - x_j) \quad \text{and} \quad g(x_1, ...x_N) = -\prod_{i<j}(x_i - x_j) \ .$$

However, it will generally not be possible to create a chain of mix-servers unless the permutation is restricted to a set that is closed under composition.

For utmost generality, one can modify the protocol to prove that $f(x_1, ...x_N, x_{\pi(1)}, ...x_{\pi(N)}) = 0$ or even $f(x_1, ...x_N, x_{\pi(1)}, ...x_{\pi(N)}) \neq 0$ for any function $f$ that can be computed verifiably. Given a commitment $y = \mathcal{C}(b, s)$, the prover can demonstrate that $b \neq 0$ by computing $t = 1/b$, $z = g^{s'} y^t$ and running a sigma proof of the form

$$\Sigma\text{-proof}\left[r, t, s' \,\middle|\, z = g^{s'} y^t \wedge z/g_1 = g^r\right] \ .$$

As an example, one can prove that a permutation is a derangement, i.e. that $\pi(i) \neq i$ for all $i$ by verifying $\prod_{i=1}^{N} \left(x_{\pi(i)} - x_i\right) \neq 0$.

In our exposition we assume for clarity that $q$ is prime, but this is not essential. Composite $q$ can be used, but this requires a more delicate analysis to handle the possibility of non-invertible elements and zero-divisors in the ring $\mathbb{Z}_q$, e.g., the random vectors are no longer vectors, but elements in a module. Even the case where $q$ is unknown can be handled using an approach similar to that of [27].

## Acknowledgements

We thank Johan Håstad for helpful discussions.

## References

1. Abe, M., Imai, H.: Flaws in some robust optimistic mix-nets. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 39–50. Springer, Heidelberg (2003)
2. Adida, B., Wikström, D.: How to shuffle in public. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 555–574. Springer, Heidelberg (2007)
3. Adida, B., Wikström, D.: Offline/online mixing. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 484–495. Springer, Heidelberg (2007)
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 136–145. IEEE Computer Society Press, Los Alamitos (2001); Full version at Cryptology ePrint Archive, Report 2000/067 (October 2001), http://eprint.iacr.org
5. Chaum, D.: Untraceable electronic mail, return addresses and digital pseudo-nyms. Communications of the ACM 24(2), 84–88 (1981)
6. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
7. de Hoogh, S., Schoenmakers, B., Skoric, B., Villegas, J.: Verifiable rotation of homomorphic encryptions. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 393–410. Springer, Heidelberg (2009)
8. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
9. Furukawa, J.: Efficient and verifiable shuffling and shuffle-decryption. IEICE Transactions 88-A(1), 172–188 (2005)
10. Furukawa, J., Miyauchi, H., Mori, K., Obana, S., Sako, K.: An implementation of a universally verifiable electronic voting scheme based on shuffling. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 16–30. Springer, Heidelberg (2003)
11. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
12. Furukawa, J., Sako, K.: An efficient publicly verifiable mix-net for long inputs. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 111–125. Springer, Heidelberg (2006)

13. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2002)
14. Groth, J.: Linear algebra with sub-linear zero-knowledge arguments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 192–208. Springer, Heidelberg (2009)
15. Groth, J., Ishai, Y.: Sub-linear zero-knowledge argument for correctness of a shuffle. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 379–396. Springer, Heidelberg (2008)
16. Jakobsson, M., Juels, A.: Mix and match: Secure function evaluation via ciphertexts. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 162–177. Springer, Heidelberg (2000)
17. Neff, A.: A verifiable secret shuffle and its application to e-voting. In: 8th ACM Conference on Computer and Communications Security (CCS), pp. 116–125. ACM Press, New York (2001)
18. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
19. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
20. Pfitzmann, B.: Breaking an efficient anonymous channel. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 332–340. Springer, Heidelberg (1995)
21. Reistad, T.I., Toft, T.: Secret sharing comparison by transformation and rotation. In: Desmedt, Y. (ed.) ICITS 2007. LNCS, vol. 4883, pp. 169–180. Springer, Heidelberg (2009)
22. Reiter, M.K., Wang, X.: Fragile mixing. In: 11th ACM Conference on Computer and Communications Security (CCS), pp. 227–235. ACM Press, New York (2004)
23. Ryan, P.Y.A., Schneider, S.A.: Prêt à voter with re-encryption mixes. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 313–326. Springer, Heidelberg (2006)
24. Sako, K., Killian, J.: Reciept-free mix-type voting scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
25. Wikström, D.: A universally composable mix-net. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 317–335. Springer, Heidelberg (2004)
26. Wikström, D.: A sender verifiable mix-net and a new proof of a shuffle. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 273–292. Springer, Heidelberg (2005)
27. Wikström, D.: A commitment-consistent proof of a shuffle. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 407–421. Springer, Heidelberg (2009)

# Batch Range Proof for Practical Small Ranges

Kun Peng and Feng Bao

Institute for Infocomm Research

**Abstract.** A batch proof and verification technique by Chida and Yamamoto is extended to work in a more general scenario. The new batch proof and verification technique is more useful and can save more cost than the original technique. An application of the new batch proof and verification technique is range proof, which proves that a secret integer is in an interval range. Like the most resent and advanced range proof protocol by Camenisch, Chaabouni and Shelat in Asiacrypt2008, the new range proof technique is especially suitable for practical small ranges, but more efficient and stronger in security than the former. The new range proof technique is very efficient and more efficient than the existing solutions in practical small ranges. Moreover, it achieves stronger security and stronger privacy (perfect honest-verifier zero knowledge) than most of the existing range proof schemes.

## 1 Introduction

In cryptographic applications, it is often needed for a party to prove that he knows a secret integer in an interval range. The party chooses an integer from an interval range $R$, encrypts it or commits to it and publishes the ciphertext or commitment. Then he has to prove that the integer encrypted in the ciphertext or committed in the commitment is in $R$. The proof cannot reveal any information about the integer except that it is in the range. This proof operation is called range proof. The following security properties must be satisfied in a range proof protocol, while high efficiency is very important as well.

- Correctness: if the integer is in the range and the prover knows the integer and strictly follows the proof protocol, he can pass the verification in the protocol.
- Soundness: if the prover passes the verification in the protocol, the integer is guaranteed with an overwhelmingly large probability to be in the range.
- Privacy: no information about the integer is revealed in the proof except that it is in the range.

The most straightforward range proof technique is ZK (zero knowledge) proof of partial knowledge [9], which proves that the committed integer may be each integer in the range one by one and then link the multiple proofs with OR logic. It has a drawback: the number of computations it needs is linear to the size of the range, which leads to very low efficiency.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 114–130, 2010.

The range proof schemes in [4,15,12] improve efficiency of range proof by discarding the cyclic group with public order in [9]. They notice that $g^x$ is a binding commitment of $x$ maintaining its non-negativity in $Z$ when the order of $g$ is unknown. So in their range proofs they employ commitment of integers in $Z$ instead of the traditional commitment of integers with a modulus. This special commitment function enables them to reduce a range proof in a range $R$ to proof of non-negativity of integers. Although this method improves efficiency, it has two drawbacks. Firstly, its soundness depends on a computational assumption: when the multiplication modulus is a composite hard to factorize multiplication operation generates a large cyclic subgroup, whose order is secret and hard to calculate. So its soundness is only computational. Secondly, it cannot achieve perfect zero knowledge or simulatability. It reveals a statistically small (and intuitively-believed negligible) amount of information about the secret integer and only achieves the co-called statistical zero knowledge.

The most recent and advanced range proof scheme is [5]. On one hand it recognizes that sacrifice of unconditional soundness and perfect zero knowledge is necessary for high efficiency. On the other hand, it shows that asymptotical efficiency is not always the dominating factor in efficiency analysis. Actually, asymptotical efficiency in range proof is only important for large ranges. As the ranges in most practical applications of range proof are not large, an asymptotically higher cost may be actually lower in practice. So, the range proof scheme in [5] ignores asymptotical efficiency but focuses on the actual cost of range proof in practical small ranges. As a result, although its asymptotical efficiency is not the highest, it achieves the highest actual efficiency in practical small ranges and is more efficient in practical applications than the other existing range proof solutions. However, [5] has its drawbacks as well. Besides conditional soundness like in [4,15,12], its has an additional limitation: its privacy depends on hardness of a special mathematical problem called ($\log_k$)-Strong Diffie Hellman assumption. Moreover, its efficiency advantage in practical small ranges is not great enough to dramatically improve efficiency of many applications.

A new range proof scheme is proposed in this paper. It is based on a new batch proof and verification technique extended from a batch proof and verification protocol in [7]. The new batch proof and verification protocol can prove and verify in a batch $n$ instances of knowledge claims, each claiming knowledge of 1-out-of-$k$ secret discrete logarithms. It is much more efficient than separately proving and verifying the $n$ instances of knowledge claims. In the new range proof protocol, the committed integer is represented in a $k$-base coding system so that range proof in a range with width $b - a$ is reduced to $n$ instances of range proof in $Z_k$ where $b - a = k^n$. Then the new batch proof and verification technique is employed to batch the $n$ instances of proof, so that efficiency of range proof in practical small ranges is greatly improved. The new range proof scheme is much more efficient than [5], not to mention the other solutions to range proof. Moreover, it achieves unconditional soundness and perfect zero knowledge (in the random oracle model) and is stronger in security and privacy (perfect honest-verifier zero knowledge) than most existing solutions to range proof including [4,15,12,5].

## 2 Background

Background knowledge about range proof and batch proof and verification is recalled in this section.

### 2.1 Range Proof and the Most Recent Development in [5]

As stated before, the simplest range proof technique [9] has a drawback: low efficiency. It is well known that it can be optimised by sealing the secret integer bit by bit and proving each commitment contains a bit. However, the optimised solution is still not efficient enough.

To improve efficiency of range proof, Boudot proposes [4]. Boudot notices that $g^x$ is a binding commitment of $x$ in $Z$ when the order of $g$ is unknown. Thus a computationally binding commitment function can be designed and any non-negative committed integer can be proved to be non-negative by showing that it is the sum of a square and a non-negative integer in a smaller range. So in his range proof [4] Boudot employs commitment of integers in $Z$ instead of the traditional commitment of integers with a modulus. This special commitment function enables him to reduce a range proof in a range $R$ to a range proof in a much smaller range. Moreover, it enlarges the smaller range and then compresses it back to its original size to asymptotically improve precision of range proof in it. Then, it implements range proof in the smaller range by publishing a monotone function of the non-negative integer in it and showing that the monotone function is in another much larger range. This method has several drawbacks. Firstly, the monotone function reveals some information of the secret integer. To limit the amount of revealed information, very large parameters (e.g. extra large integers) must be employed, so that many computations in [4] are more costly than usual. Secondly, as distribution of the monotone function cannot precisely implies the exact range of its pre-image, it is necessary to firstly enlarge the smaller range and then compress it back into its original size. The enlarging and compressing process has two drawbacks: it is only asymptotically sound and it further enlarges parameters to further increase the cost of the computations.

Later [15,12] are proposed. They inherit the setting in $Z$ and cyclic groups with secret orders but do not aim at purely high efficiency (especially in the terms of the number of exponentiations). Instead they employ more comprehensive analysis and pursue better balance and trade-off between security properties. Their soundness is absolutely precise and they do not need extra large parameters. However, they have their own drawbacks. They only need a constant number of computations as well, but employ Rabin and Shallit algorithm, which is costly and needs a lot of computations.

As [4,15,12] employs cyclic groups with secret orders and computations in $Z$, they have two drawbacks. Firstly, they depend on hardness of the factorization problem to ensure that the cyclic groups they employ have secret orders and their soundness is guaranteed. Secondly, certain computations in $Z$ (e.g. calculation of response in $Z$ as a monotone function of a secret in zero knowledge proofs) violate

perfect zero knowledge and can at most achieve statistical zero knowledge. So their soundness is only computational and their privacy is only statistical.

The most recent and advanced range proof scheme is [5]. As the newest development in range proof, [5] has its own focus. It points out that in most practical applications of range proof the ranges are not large. In [9,5], the size of a range can be as large as the order of a large cyclic group, which is usually hundreds of bits long. In [4,15,12], the size of a range has no up-bound. However, in practical applications the range size is usually not so large. Our own observation agrees with the point of view in [5]. For example, in some secure e-auction schemes [8,13,1,16] a bidder has to prove that his secret bid is in an interval range containing the biddable prices; in some secure e-voting schemes [14,12] a voter has to prove that his secret vote contains the code for one of the candidates. In these applications, the range size is much smaller than permitted. So asymptotical efficiency is not so important in range proof as emphasized in [4,15,12,5]. Instead, the actual cost, especially with a practical small range, is more important. The range proof protocol in [5] is efficient with a practical small range although it has lower asymptotical efficiency. It is convincingly illustrated in [5] that the range proof protocol in it is more efficient than any other existing range proof solutions when the range is a practical small range. Security of [5] depends on hardness of $(\log_k)$-Strong Diffie Hellman assumption.

To be compared with our new range proof technique, the concrete cost of the range proof protocol in [5] must be estimated. For fairness of the comparison, costs of both the range proof protocol in [5] and our new range proof protocol are measured in the number of multiplications. The range proof and verification protocol is presented in Fig-3 in [5] and consists of 6 steps where the range size is denoted as $u^l$. In Step 1, the verifier carries out $u + 1$ exponentiations; in Step 2, the prover carries out $l$ exponentiations and then the prover and verifier run a zero knowledge proof, which costs $12l + 3$ exponentiations; in Step 3, the prover carries out $4l$ exponentiations; in Step 6 (final verification), the verifier carries out $4l + 2$ exponentiations. In the 6 steps, there are in addition $13l + 2$ multiplications.

### 2.2 Batch Proof and Verification

Batch verification was first formally proposed by Bellare *et al* [3] to efficiently verify correctness of multiple exponentiation operations. Bellare *et al* propose three batch verification techniques: random subset test, small exponent test and bucket test. In their most popular test, small exponent test, one equation $\prod_{i=1}^{k} y_i^{t_i} = g^{\sum_{i=1}^{k} t_i x_i}$ is used to check $k$ equations: $y_i = g^{x_i}$ for $i = 1, 2, \ldots, k$. As challenges $t_i$ for $i = 1, 2, \ldots, k$ can be shorter than a full-length integer, efficiency of computation can be greatly improved. Bellare *et al* demonstrate that full length challenges are not necessary in practice and shorter $t_i$ can satisfy soundness in practical sense. Although when full length challenges are used soundness of the verification can be guaranteed with a probability $1 - 1/q$ where $q$ is a full length parameter (e.g. 256 bits long), very few practical application

needs so large a probability of soundness as $1-2^{-256}$. So Bellare [3] *et al* suggest to make a trade-off between soundness and efficiency. Bellare *et al* verify the statements in a batch to improve efficiency, while soundness of the verification is weakened to a practical level. There is an important parameter in Bellare's batch verification technique, which is denoted as $L$ in this paper. Bellare *et al* illustrate that if the batch verification succeeds with a probability larger than $2^{-L}$, it guarantees that all the statements are correct. As most of the computational cost of the batch verification is linear in $L$, Bellare *et al* suggest $L$ to be much shorter than full length such that great efficiency improvement can be achieved. According to the estimation by Bellare *et al* $2^{-L}$ is smaller than one out of one billion when $L = 30$ and the achieved soundness is strong enough for most practical applications like signature verification.

Batch proof and verification [11,2,18,17] is an extension of batch verification. It also employs the idea in [3] and sacrifice unnecessary strength of soundness to improve efficiency. Bellare's batch verification is extended into batch ZK proof and verification [11] of multiple knowledge claims, all of which are correct. In [17], batch proof and verification is extended to handle multiple possible knowledge claims, only one of which is correct. The batch proof and verification techniques in [3] and [11] only handle multiple claims linked with ALL logic, so cannot be applied to range proof, which employs OR logic.

### 2.3 Batch Proof and Verification Technique by Chida and Yamamoto [7]

A batch proof and verification technique is proposed in [7] to batch prove and verify multiple zero knowledge proofs of knowledge of 1-out-of-2 discrete logarithms. Its main idea is proposed in the so-called Protocol 3 in [7], which is recalled without any change in Figure 1.

It has been formally proved in [7] that the batch proof and verification protocol in Figure 1 is perfectly simulatable (and thus perfect zero knowledge) in the random oracle model and guarantees with an overwhelmingly large probability that the prover knows $\log_g y_{i,0}$ or $\log_g y_{i,1}$ for $i = 1, 2, \ldots, n$.

## 3 Extended Batch Proof and Verification

Our idea is to extend batch proof and verification of knowledge of 1-out-of-2 discrete logarithms to batch proof and verification of knowledge of 1-out-of-$k$ discrete logarithms where $k$ can be any integer larger than 1. Suppose there are $n$ instances of proof and in each of them a prover has to prove that he knows at least one of $k$ secret discrete logarithms. The $n$ instances of proof and the corresponding verification are batched into a single proof and verification protocol such that efficiency can be improved. The extended batch proof and verification protocol is described in Figure 2.

Security of the new batch proof and verification protocol is analysed in Definition 1, Theorem 1 and Theorem 2 where the symbol $W()$ is defined as witnesses

Common input: $(p, q, g)$, $\{(y_{i,0}, y_{i,1})\}_{i=1,\ldots,n}$ where $G$ is an instance generator such that $(p, q, g) \leftarrow G(1^\rho)$ and $p, q$ are primes such that $q|p-1$ holds.
Knowledge to prove: $b_i \in \{0,1\}, s_{i,b_i}$ s.t. $y_{i,b_i} = g^{s_{i,b_i}}$.

1. The prover selects $r, v, c_{i,\bar{b}_i} \in_R Z/qZ$ and computes

$$R_0 = g^r \prod_{\{i|b_i=1\}} y_{i,0}^{c_{i,0}}$$
$$R_1 = g^v \prod_{\{i|b_i=0\}} y_{i,1}^{c_{i,1}}$$
$$c_i = H(CI||c_{i-1}||c_{i-1,0})$$
$$c_{i,b_i} = c_i - c_{i,\bar{b}_i} \bmod q$$
$$z_0 = r - \sum_{\{i|b_i=0\}} c_{i,0} s_{i,0} \bmod q$$
$$z_1 = v - \sum_{\{i|b_i=1\}} c_{i,1} s_{i,1} \bmod q$$

   where $CI$ is a bit string comprising common inputs in a certain order, $c_0 = R_0$, $c_{0,0} = R_1$. It then sends $(z_0, z_1, c_1, c_{1,0}, \ldots, c_{n,0})$ to the verifier.
2. The verifier computes

$$c_{i,1} = c_i - c_{i,0} \bmod q$$
$$c_{i+1} = H(CI||c_i||c_{i,0})$$

   and verifies

$$c_1 = H(CI||g^{z_0} \prod_{i=1}^n y_{i,0}^{c_{i,0}}||g^{z_1} \prod_{i=1}^n y_{i,1}^{c_{i,1}})$$

**Fig. 1.** Batch Proof and Verification of knowledge of 1-out-of-2 Discrete Logarithms

of $x$: $W(x) \stackrel{def}{=} \{w|(x,w) \in R\}$ in the beginning of Section 2.1 of [7]. Definition 1, Theorem 1 and Theorem 2 in this paper are the same as Definition 1, Lemma 1 and Lemma 2 in [7] except necessary fixing of grammar mistakes and that the special case that $k = 2$ is extended to the general scenario that $k$ can be any integer larger than 1. So proof of Theorem 1 and Theorem 2 is almost the same as the proofs of Lemma 1 and Lemma 2 in [7]. Just extend $k = 2$ to $k \geq 2$ and repeat some operations $k-1$ times, and proofs of Lemma 1 and Lemma 2 in [7] become proofs of Theorem 1 and Theorem 2 in this paper respectively. The extended proofs are so straightforward and cannot be regarded as our original contribution, so are given in Appendix A for the readers who has difficulty in accessing [7].

**Definition 1.** *(R-incompatibility) Let* $L_R \stackrel{def}{=} \{y|\exists x \text{ s.t. } (y,x) \in R\}$. *For input* $(y_1, y_2, \ldots, y_k, b, x_b)$ *such that* $y_1, y_2, \ldots, y_k \in L_R$, $b \in \{1, 2, \ldots, k\}$ *and* $x_b \in W(y_b)$, *if no probabilistic polynomial-time Turing machine can output* $x_\beta \in W(y_\beta)$ *for* $\beta \in \{1, 2, \ldots, k\}$ *and* $\beta \neq b$ *except for a negligible error, we call* $(y_1, y_2, \ldots, y_k) \in L_R^k$ *R-incompatible. In particular, we call a set consisting of R-incompatible elements obtained from the same language R-incompatible set.*

Common input: $(p, q, g)$, $\{(y_{i,1}, y_{i,2}, \ldots, y_{i,k})\}_{i=1,2,\ldots,n}$.
Knowledge to prove: $b_i \in \{1, 2, \ldots, k\}$, $s_{i,b_i}$ s.t. $y_{i,b_i} = g^{s_{i,b_i}} \bmod p$ for $i = 1, 2, \ldots, n$.
Denotation: $S_i = \{1, 2, \ldots, b_i - 1, b_i + 1, \ldots, k\}$ and $L$ is a security parameter.

1. The prover randomly selects $r_1, r_2, \ldots, r_k$ from $Z_q$ and $c_{i,j}$ for $i = 1, 2, \ldots, n$ and $j \in S_i$ from $Z_{2^L}$. Then he computes

$$
\begin{gathered}
R_1 = g^{r_1} \prod_{1 \le i \le n,\ b_i = 1} \prod_{j \in S_i} y_{i,j}^{c_{i,j}} \bmod p \\
R_2 = g^{r_2} \prod_{1 \le i \le n,\ b_i = 2} \prod_{j \in S_i} y_{i,j}^{c_{i,j}} \bmod p \\
\cdots\cdots \\
\cdots\cdots \\
R_k = g^{r_k} \prod_{1 \le i \le n,\ b_i = k} \prod_{j \in S_i} y_{i,j}^{c_{i,j}} \bmod p \\
c_i = H(CI || c_{i-1} || c_{i-1,1} || c_{i-1,2} || \ldots || c_{i-1,k-1}) \text{ for } i = 1, 2, \ldots, n \\
c_{i,b_i} = c_i - \sum_{j \in S_i} c_{i,j} \bmod q \text{ for } i = 1, 2, \ldots, n \\
z_1 = r_1 - \sum_{\{i | b_i = 1\}} c_{i,1} s_{i,1} \bmod q \\
z_2 = r_2 - \sum_{\{i | b_i = 2\}} c_{i,2} s_{i,2} \bmod q \\
\cdots\cdots \\
\cdots\cdots \\
z_k = r_k - \sum_{\{i | b_i = k\}} c_{i,k} s_{i,k} \bmod q
\end{gathered}
$$

   where $CI$ is a bit string comprising common inputs in a certain order and

$$
\begin{gathered}
c_0 = R_1 \\
c_{0,1} = R_2 \\
c_{0,2} = R_3 \\
\cdots\cdots \\
\cdots\cdots \\
c_{0,k-1} = R_k.
\end{gathered}
$$

   It then sends
   $(z_1, z_2, \ldots, z_k, c_1, c_{1,1}, c_{1,2} \ldots, c_{1,k-1}, c_{2,1}, c_{2,2} \ldots, c_{2,k-1}, \cdots\cdots$
   $c_{n,1}, c_{n,2} \ldots, c_{n,k-1})$ to the verifier.

2. The verifier computes

$$
\begin{gathered}
c_{i,k} = c_i - \sum_{j=1}^{k-1} c_{i,j} \bmod 2^L \text{ for } i = 1, 2, \ldots, n \\
c_i = H(CI || c_{i-1} || c_{i-1,1} || c_{i-1,2} || \ldots || c_{i-1,k-1}) \text{ for } i = 1, 2, \ldots, n
\end{gathered}
$$

   and verifies

$$
\begin{aligned}
c_1 = H(CI || g^{z_1} \prod_{i=1}^{n} y_{i,1}^{c_{i,1}} \bmod p || g^{z_2} \prod_{i=1}^{n} y_{i,2}^{c_{i,2}} \bmod p \\
|| \ldots || g^{z_k} \prod_{i=1}^{n} y_{i,k}^{c_{i,k}} \bmod p)
\end{aligned}
$$

**Fig. 2.** Batch Proof and Verification of knowledge of 1-out-of-k Discrete Logarithms

**Theorem 1.** *(Simulatability) Let* $view_R \stackrel{def}{=} (z_1, z_2, \ldots, z_k, c_1, c_{1,1}, c_{1,2}, \ldots, c_{1,k-1}, c_{2,1}, c_{2,2} \ldots, c_{2,k-1}, \ldots\ldots, c_{n,1}, c_{n,2} \ldots, c_{n,k-1})$. *There exists a simulator that, on input* $((p,q,g), \{(y_{i,1}, y_{i,2}, \ldots, y_{i,k})\}_{i=1,2,\ldots,n})$, *outputs* $view_S$ *which is perfectly indistinguishable from* $view_R$ *in expected polynomial time in the random oracle model.*

**Theorem 2.** *(Soundness) Suppose that* $\{(y_{i,1}, y_{i,2}, \ldots, y_{i,k})\}_{i=1,2,\ldots,n}$ *in Protocol 3 is an R-incompatible set. Then, if the prover is successful in producing* $(z_1, z_2, \ldots, z_k, c_1, c_{1,1}, c_{1,2}, \ldots, c_{1,k-1}, c_{2,1}, c_{2,2} \ldots, c_{2,k-1}, \ldots\ldots, c_{n,1}, c_{n,2} \ldots, c_{n,k-1})$ *accepted by the verifier, the prover has witnesses* $b_i \in \{1, 2, \ldots, k\}$ *and* $s_{i,b_i} \in Z_q$ *s.t.* $y_{i,b_i} = g^{s_{i,b_i}}$ *for* $i = 1, 2, \ldots, n$ *with an overwhelmingly large probability in the random oracle model.*

In [7], it is illustrated that the overwhelmingly large probability to guarantee soundness of batch proof and verification is determined by the length of the challenges $c_{i,j}$. We apply the key point emphasized by Bellare [3] but ignored in [7]: the challenges $c_{i,j}$ are not necessary to be full length. Instead, they can be much smaller than $q$, while very strong soundness can still be achieved. For example, when $c_{i,j}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$ are 40 bits long,

- on one hand, the probability that a cheating prover without the claimed knowledge can pass the verification is no more than $2^{-40}$, a very small probability negligible in any practical application;
- on the other hand, the cost of an exponentiation with an exponent $c_{i,j}$ is only 40/256 of the cost of an exponentiation with a full-length exponent when $q$ is 256 bits long.

Like in [3], cost of the new batch proof and verification protocol is measured in terms of the number of full-length exponentiations. We assume that a full-length exponent like $r_j$ and $z_j$ is 256 bits long. An exponentiation with an $L$-bit exponent is regarded to be as costly as $L/256$ full length exponentiation. So in the new batch proof and verification protocol, the prover's cost is $k + (k-1)nL/256$ full exponentiations and the verifier's cost is full $k + knL/256$ exponentiations. In comparison, the standard solution to the $n$ instances of proof of knowledge of 1-out-of-k discrete logarithms, $n$ instances of zero knowledge proof of partial knowledge [9], costs the prover $2n(k-1)+n$ full length exponentiations and the verifier $2nk$ full length exponentiations. So efficiency improvement of the new batch proof and verification protocol is great.

## 4 The New Range Proof Protocol

The main idea of the new range proof scheme is to represent the secret integer in a base-$k$ system. Thus proof that an integer $x$ is in a range $\{a, a+1, \ldots, b\}$ can be reduced to $\log_k(b-a)$ instances of proof that each digit of the base-$k$ representation of $x - a$ is in $Z_k$. Then the $\log_k(b-a)$ instances of proof can be batched using the new batch proof and verification of knowledge of 1-out-of-$k$ discrete logarithms to improve efficiency.

As most range proof schemes employ the Fujisaki-Okamoto commitment function [10] or its variants to seal the secret integer, we make the same choice in commitment function. As illustrated in [10], this family of commitment functions is information-theoretically hiding and does not reveal any information. Moreover, it is only computationally binding and uniqueness of the committed integer is only guaranteed when the prover is assumed to be a polynomial party and cannot solve the DL problem. These two points will not be emphasized again in this paper. We believe that in practical applications the prover is a polynomial party and the DL problem is hard, so from now on in this paper we will use phases like "the integer committed in this commitment". That does not mean that we believe a unique integer is unconditionally committed in the commitment. Moreover, when proving and analysing soundness of our new range proof protocol we will not focus our attention on the effect of the commitment function on soundness of the range proof, as it is well known that the effect is a limitation based on hardness of the DL problem. Instead, we will focus on soundness of the proof operations themselves. When we claim soundness of the new range proof protocol without emphasizing its dependence on bindingness of the commitment function or hardness of the DL problem we do not mean to ignore this condition, but feel unnecessary to repeat a well known fact. Our claim only reflects the fact that soundness of our range proof itself does not need any computational assumption.

A secret integer $x$ is committed to $c = g^x h^r \bmod p$ where $h$ is a generator of $G$, $\log_g h$ is unknown and $r$ is a random integer in $Z_q$. $x$ is in an interval range $\{a, a+1, \ldots, b\}$ where $b-a < q$. A party with knowledge of $x$ and $r$ has to prove that the message committed in $c$ is in $\{a, a+1, \ldots, b\}$. The proof protocol and the corresponding verification are as follows.

1. $c' = c/g^a \bmod p$ and the proof that the integer committed in $c$ is in $\{a, a+1, \ldots, b\}$ is reduced to proof that the integer committed in $c'$ is in $\{0, 1, \ldots, b-a\}$.
2. The prover calculates representation of $x - a$ in the base-$k$ system $(x_1, x_2, \ldots, x_n)$ to satisfy $x - a = \sum_{i=1}^{n} x_i k^{i-1}$ where for simplicity of description[1] it is assumed $(b-a) = k^n$.
3. The prover randomly chooses $r_1, r_2, \ldots, r_n$ in $Z_q$ and calculates and publishes $e_i = g^{x_i} h^{r_i} \bmod p$ for $i = 1, 2, \ldots, n$.
4. The prover publicly proves that he knows a secret integer $r' = \sum_{i=1}^{n} r_i k^{i-1} - r \bmod q$ such that $h^{r'} c' = \prod_{i=1}^{n} e_i^{k^{i-1}} \bmod p$ using zero knowledge proof of knowledge of discrete logarithm [19].
5. The range proof is reduced to $n$ smaller-scale ranges proofs: the integer committed in $e_i$ is in $Z_k$ for for $i = 1, 2, \ldots, n$. Those $n$ instances of proof

[1] Sometimes $b-a$ is not an exponentiation, then a generalization mechanism is needed. A simple example of generalization mechanism is finding $k$ and $n$ such that $k^n$ is s little bit larger than $b-a$. Firstly, range proof is proved in the a-little-bit-larger range. Then it is proved that the commited integer is not in the extra part by showing that it is inequal to every integer in the extra part. More effective and complex generalization function can be found in [6].

can be implemented through $n$ instances of proof of knowledge of 1-out-of-k discrete logarithms

$$KN(\log_h e_i) \vee KN(\log_h e_i/g) \vee KN(\log_h e_i/g^2) \vee \ldots \\ \vee KN(\log_h e_i/g^{k-1}) \text{ for } i = 1, 2, \ldots, n \quad (1)$$

where $KN(z)$ denotes knowledge of $z$.

6. Proof of (1) can be implemented through batch proof and verification of knowledge of 1-out-of-$k$ discrete logarithms in Section 3.

The new range proof protocol is correct, sound and achieves zero knowledge privacy as illustrated in Theorem 3, Theorem 4 and Theorem 5.

**Theorem 3.** *If the message committed in $c$ is in the range $\{a, a+1, \ldots, b\}$, the prover can pass the verification in the new range proof protocol.*

*Proof.* If the message committed in $c$ is in the range $\{a, a+1, \ldots, b\}$, the message committed in $c' = c/g^a = g^{x-a}h^r$, namely $x-a$, is in the range $\{0, 1, \ldots, b-a\}$. As $n = \log_k(b-a)$ and $x - a = \sum_{i=1}^{n} x_i k^{i-1}$, $(x_1, x_2, \ldots, x_n)$ is the base-$k$ representation of $x - a$. So

- firstly,

$$h^{r'}c' = h^{\sum_{i=1}^{n} r_i k^{i-1} - r} g^x h^r / g^a = h^{\sum_{i=1}^{n} r_i k^{i-1}} g^{x-a} \\ = h^{\sum_{i=1}^{n} r_i k^{i-1}} g^{\sum_{i=1}^{n} x_i k^{i-1}} = \prod_{i=1}^{n} (g^{k^{i-1} x_i} h^{k^{i-1} r_i}) \\ = \prod_{i=1}^{n} (g^{x_i} h^{r_i})^{k^{i-1}} = \prod_{i=1}^{n} e_i^{k^{i-1}} \bmod p$$

  and thus the proof in Step 4 of the protocol can pass its verification;
- secondly, each $x_i$, the $i^{th}$ least significant digit in the base-$k$ representation of $x - a$, is in $Z_q$ and thus the proof in Step 5 of the protocol can pass its verification.

Therefore, the prover can pass the verification in the new range proof protocol. □

**Theorem 4.** *When $b-a$ is a power of $k$, if the prover passes the verification in the new range proof protocol, then it is guaranteed with an overwhelmingly large probability that the message committed in $c$ is in the range $\{a, a+1, \ldots, b\}$.*

*Proof.* As the prover passes the verification in Step 5 of the new range proof protocol, according to Theorem 2, the message committed in each $e_i$ is in $Z_k$ with an overwhelmingly large probability. So the message committed in $\prod_{i=1}^{n} e_i^{k^{i-1}}$ is in $\{0, 1, \ldots, b-a\}$ with an overwhelmingly large probability as $n = \log_k(b-a)$.

As the prover passes the verification in Step 4 of the new range proof protocol, according to the formal proof of soundness in [19] with an overwhelmingly large probability the prover knows a secret integer $r'$ such that $h^{r'}c' = \prod_{i=1}^{n} e_i^{k^{i-1}} \bmod p$. So the same integer is committed in $c'$ and $\prod_{i=1}^{n} e_i^{k^{i-1}}$ and thus the message committed in $c'$ is in $\{0, 1, \ldots, b-a\}$ with an overwhelmingly large probability.

As it is publicly verifiable that $c' = c/g^a \bmod p$, the message committed in $c$ is in $\{a, a+1, \ldots, b\}$ with an overwhelmingly large probability. □

**Theorem 5.** *The new range proof protocol is perfect zero knowledge in the random oracle model.*

*Proof.* The new range proof protocol consists of two proof primitives: zero knowledge proof of knowledge of discrete logarithm [19] and batch proof and verification of knowledge of 1-out-of-k discrete logarithms in Section 3. The former has been formally proved in [19] to be perfect zero-knowledge when an interactive challenge is generated by an honest verifier or a non-interactive challenge is generated by a hash function in the random oracle model. Theorem 1 has proved that batch proof and verification of knowledge of 1-out-of-k discrete logarithms in Section 3 is perfect zero knowledge in the random oracle model. As its both employed proof primitives are perfect zero knowledge in the random oracle model, the new range proof protocol is perfect zero knowledge in the random oracle model. □

## 5 Comparison of Efficiency and Security

The new range proof scheme is compared with the existing range proof schemes in Table 1. In analysis of communicational cost, the number of transfered bits are counted. In analysis of computational cost, both the prover's and the verifier's operations are included and the number of multiplications are counted, where each exponentiation is counted in terms of a number of multiplications according to the length of its exponent. When we say a range proof scheme is unconditionally sound, we mean soundness of the proof itself is achieved with an overwhelmingly large probability and without any assumption on hardness of mathematical problems. We emphasize that this conclusion does not take the commitment function into account. More precisely, if an unconditionally hiding and computationally binding commitment function is employed to seal the secret integer before the range proof starts, uniqueness of the committed integer depends on hardness of the mathematical problem bindingness of the commitment function is based on.

Range proof though proof of partial knowledge [9] needs a cost linear in the size of the range, so is inefficient both in the asymptotical sense and with small ranges. Although none of [4,15,12] has given a comprehensive cost analysis and range proof though bit-by-bit proof of partial knowledge is not specifically and formally proposed in any paper, it has been convincingly illustrated in [5] that they have no advantage with practical small ranges. $u$ in [5] is actually $k$ in the new scheme and $l$ in [5] is actually $n$ in the new scheme. So in efficiency analysis of both [5] and the new range proof scheme, $k$ and $n$ are used to measure the cost to achieve a clearer comparison. $|G_1|$, $|G_T|$ and $|Z_p|$ are symbols used in [5]. $|G_1|$ and $|Z_p|$ are estimated as 256 here in [5] and for fairness of the comparison, a full-length integer is assumed to be 256 bits long in our scheme as well. In the new range proof scheme, $L = 40$. As emphasized in Section 2.2, an important method to improve efficiency in batch proof and verification is employing challenges shorter than a full-length integer, as Bellare *et al* have

demonstrated that full length challenges are not necessary in practice. Bellare *et al* [3] and Peng *et al* [18] show that a 40-bit challenge is already long enough to guarantee very strong soundness (only failing with a probability $2^{-40}$) in any practical applications. An exponentiation with a full-length exponent is regarded as $2^{3-1} + 256 + 256/(3+1) = 314$ multiplications. Product of $n$ $L$-bit powers is regarded as $2^{3-1}n + 40 + n40/(3+1) = 14n + 40$ multiplications. So, in our new range proof scheme,

- in Step 3, $n$ full-length integers are transfered and $n$ full-length exponentiations and $n$ instances of $\log_2 k$-bit-exponent exponentiations are needed;
- in Step 4, ZK proof of knowledge of discrete logarithm transfers 2 full-length integers and costs 3 full-length exponentiations;
- in Step 5 and Step 6, batching of $n$ instances of ZK proof of knowledge of 1-out-of-$k$ discrete logarithm transfers $k$ full-length integers and $n(k-1)+1$ $L$-bit integers and costs $k$ instances of product of $n$ powers and $k$ instances of product of $n+1$ powers.

So, altogether the new range proof scheme cost $256(n+k+2)+40n(k-1)+40$ bits in communication and $314(n+3) + k(28n+94) + n(1.25\log_2 k + 4) = 28kn + 1.25n\log_2 k + 318n + 94k + 1256$ multiplications in computation.

Although the new range proof technique is not the most efficient in the asymptotical sense, it is the most efficient with practical small ranges. When the range $b-a$ is smaller than $10^{10}$, $nk$ is no more than 100 and thus our range proof is much more efficient than [5], not to mention costly pairing operations are needed and every independent verifier has to repeat the proof in [5].

**Table 1.** Comparison of range proof schemes

| proof range | soundness | privacy | computation | communication |
|---|---|---|---|---|
| [9] | unconditional | perfect ZK | no advantage in small ranges | no advantage in small ranges |
| [9] bit by bit | unconditional | perfect ZK | no advantage in small ranges | no advantage in small ranges |
| [4] | computational asymptotical | statistical ZK | no advantage in small ranges | no advantage in small ranges |
| [15,12] | computational | statistical ZK | no advantage in small ranges | no advantage in small ranges |
| [5] [a] | computational | computational ZK | $314k + 6597n$ $+1884$ + pairing for every verifier | $(k+2n-4)\|G_T\|+$ $+4n\|Z_p\| + n\|G_1\| =$ $942k + 3454n - 3768$ for every verifier |
| new | unconditional | perfect ZK | $28kn + 1.25n\log_2 k+$ $318n + 94k + 1256$ | $256(n+k+2)+$ $40n(k-1) + 40$ |

[a] Note that in [5], communication of $256(k+1)$ bits and computation of $314k$ multiplications can be resued later in a new range proof of another secret integer with the same verifier.

## 6 Conclusion

The new batch proof and verification technique proposed in this paper is more general and can save more cost than the batch proof and verification technique in [7]. With practical small ranges, the new range proof scheme proposed in this paper is much more efficient than [5], not to mention the other solutions. Moreover, it achieves unconditional soundness and perfect zero knowledge (in the random oracle model) and is stronger in security than most existing solutions to range proof.

## References

1. Abe, M., Suzuki, K.: M+1-st price auction using homomorphic encryption. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 115–124. Springer, Heidelberg (2002)
2. Aditya, R., Peng, K., Boyd, C., Dawson, E.: Batch verification for equality of discrete logarithms and threshold decryptions. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 494–508. Springer, Heidelberg (2004)
3. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
4. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
5. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient protocols for set membership and range proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008)
6. Chaabouni, R., Lipmaa, H., Shelat, A.: Additive combinatorics and discrete logarithm based range protocols (2009), http://eprint.iacr.org/2009/469
7. Chida, K., Yamamoto, G.: Batch processing for proofs of partial knowledge and its applications. IEICE Trans. Fundamentals E91CA(1), 150–159 (2008)
8. Chida, K., Kobayashi, K., Morita, H.: Efficient sealed-bid auctions for massive numbers of bidders with lump comparison. In: Davida, G.I., Frankel, Y. (eds.) ISC 2001. LNCS, vol. 2200, pp. 408–419. Springer, Heidelberg (2001)
9. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
10. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
11. Gennaro, R., Leigh, D., Sundaram, R., Yerazunis, W.S.: Batching schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 276–292. Springer, Heidelberg (2004)
12. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (2005)
13. Kikuchi, H.: (m+1)st-price auction. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, p. 341. Springer, Heidelberg (2002)

14. Lee, B., Kim, K.: Receipt-free electronic voting scheme with a tamper-resistant randomizer. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003)
15. Lipmaa, H.: On diophantine complexity and statistical zero-knowledge arguments. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 398–415. Springer, Heidelberg (2003)
16. Omote, K., Miyaji, A.: A second-price sealed-bid auction with the discriminant of the p-th root. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 57–71. Springer, Heidelberg (2003)
17. Peng, K., Bao, F.: Batch zk proof and verification of or logic. In: Yung, M., Liu, P., Lin, D. (eds.) Inscrypt 2008. LNCS, vol. 5487, pp. 141–156. Springer, Heidelberg (2009)
18. Peng, K., Boyd, C.: Batch zero knowledge proof and verification and its applications. ACM TISSEC 10(2), Article No. 6 (May 2007)
19. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology 4, 161–174 (1991)

## Appendix

## A Proof of Theorem 1 and Theorem 2

*Proof of Theorem 1.* A simulator performs the following procedure for input $((p,q,g),\{(y_{i,0},y_{i,2},\dots y_{i,k})\}_{i=1,\dots,n})$.

1. Select $\tilde{r}_1,\tilde{r}_2,\dots,\tilde{r}_k,\tilde{c}_{1,1},\tilde{c}_{1,2},\dots,\tilde{c}_{n,k-1},\tilde{c}_{n,k}$ from $Z_q$.
2. Compute

$$\tilde{R}_j = g^{\tilde{r}_j}\prod_{i=1}^{n} y_{i,j}^{\tilde{c}_{i,j}} \text{ for } j=1,2,\dots,k$$
$$\tilde{c}_i = \sum_{j=1} kc_{i,j} \bmod q \text{ for } i=1,2,\dots,n$$
$$\tilde{z}_j = \tilde{r}_j \bmod q \text{ for } j=1,2,\dots,k$$

3. Output $view_S \stackrel{def}{=} (\tilde{z}_1,\tilde{z}_2,\dots,\tilde{z}_k,\tilde{c}_1,\tilde{c}_{1,1},\dots,\tilde{c}_{n,k-1})$.

Assume that $H(.)$ is a random function that maps $\{0,1\}^*$ to $Z_q$, however, it returns $\tilde{c}_i$ when the string $(CI||\tilde{c}_{i-1}||\tilde{c}_{i-1,1}||\dots||\tilde{c}_{i-1,k-1}$ is input, where $\tilde{c}_1=\tilde{R}_1,\tilde{c}_{1,1}=\tilde{R}_1,\dots\tilde{c}_{1,k-1}=\tilde{R}_k$. Then it is clear $view_S$ is accepted by $V$ and $view_R$ and $view_S$ are perfectly indistinguishable. □

*Proof of Theorem 2.* To prove Theorem 2, we first consider its interactive version with a prover $P$ and an honest verifier $V$ in Figure 3.

Let $P^*$ be a possibly cheating prover against the interactive proof in Figure 3. Suppose $\pi$ represents the proof protocol hereafter. Consider the $P^*$-oracle machine, $M^{P^*}$, as described below.

1. $P^*$ and $V$ run $\pi$. If $V$ does not accept, run $\pi$ again. Otherwise, set $c'_i \leftarrow c_i$, $c'_{i,0} \leftarrow c_{i,0}$ for $i=1,2,\dots,n$ and $z'_{n,j} \leftarrow z_{n,j}$ for $j=1,2,\dots,k$ (Denote the successful protocol as $\pi_s uc$.).

2. Set $l \leftarrow n$.
3. Rewind $\pi_s uc$ to Step 2 of the proof rotocol in Figure 3 and run $\pi$ from the step, however, $V$ selects challenge data as $c'_i$ for $i = 1, 2, \ldots, l$ and $c_i \in_R Z_q$ for $i = l, l+1, \ldots, n$ (Denote the successful protocol as $\pi_l$.). If $V$ does not accept, run $\pi$ from the step again.

1. $P$ randomly selects $r_1, r_2, \ldots, r_k$ from $Z_q$ and $c_{i,j}$ for $i = 1, 2, \ldots, n$ and $j \in S_i$ from $Z_{2^L}$. and computes

$$R_1 = g^{r_1} \prod_{1 \le i \le n,\ b_i = 1} \prod_{j \in S_i} y_{i,j}^{c_{i,j}} \bmod p$$
$$R_2 = g^{r_2} \prod_{1 \le i \le n,\ b_i = 2} \prod_{j \in S_i} y_{i,j}^{c_{i,j}} \bmod p$$
$$\ldots\ldots$$
$$\ldots\ldots$$
$$R_k = g^{r_k} \prod_{1 \le i \le n,\ b_i = k} \prod_{j \in S_i} y_{i,j}^{c_{i,j}} \bmod p.$$

   It then sends $R_1, R_2, \ldots, R_k$ to $V$.
2. $V$ selects $c_1 \in Z_q$ and sends it to $P$.
3. $P$ computes

$$c_{1,b_1} = c_1 - \sum_{j \in S_1} c_{1,j}$$
$$z_{1,b_1} = r_1 - c_{1,b_1} s_{1,b_1}$$
$$z_{1,j} = r_j \text{ for } j \in S_1$$

   and sends $c_{1,1}, c_{1,2}, \ldots, c_{1,k-1}$ to $V$.
4. Repeat the following steps for $i = 2, \ldots, n$.
   (a) $V$ randomly selects $c_i$ from $Z_q$ and sends it to P.
   (b) $P$ computes

$$c_{i,b_i} = c_i - \sum_{j \in S_i} c_{i,j}$$
$$z_{i,b_i} = z_{i-1,b_i} - c_{i,b_i} s_{i,b_i}$$
$$z_{i,j} = z_{i-1,j} \text{ for } j \in S_i$$

   and sends $c_{i,1}, c_{i,2}, \ldots, c_{i,k-1}$ to $V$
5. $P$ sends $z_{n,0}, z_{n,1}, \ldots, z_{n,k}$ to $V$.
6. $V$ computes

$$c_{i,k} = c_i - \sum_{j=1}^{k-1} c_{i,j} \bmod q \text{ for } i = 1, 2, \ldots, n$$

   and verifies

$$g^{z_{n,j}} = R_j \prod_{i=1}^{n} y_{i,j}^{-c_{i,j}} \bmod p \text{ for } j = 1, 2, \ldots, k$$

   It then returns accept or reject.

**Fig. 3.** Batch Proof and Verification of Knowledge of 1-out-of-k Discrete Logarithms in its Interactive version

4. Set

$$\begin{aligned} \Delta c_i &= c_i - c'_i \bmod q \\ \Delta c_{i,0} &= c_{i,0} - c'_{i,0} \bmod q \\ \Delta c_{i,1} &= (c_i - c_{i,0}) - (c'_i - c'_{i,0}) \bmod q \\ \Delta z_j &= z_{n,j} - z'_{n,j} \bmod q \end{aligned}$$

5. Abort if $\Delta c_{l,j} = 0$.
6. Select $b \in \{1, 2, \ldots, k\}$ such that $\Delta c_{l,b} \neq 0$ and set

$$\begin{aligned} \hat{s}_{l,b} &= (\Delta z_b + \textstyle\sum_{i=l+1, \hat{b}_i = b}^{n} \Delta c_{i,b} \hat{s}_{i,b}) / - \Delta c_{l,b} \\ \hat{b}_l &= b \end{aligned} \tag{2}$$

7. Abort if $y_{l,\hat{b}_l} \neq g^{\hat{s}_{l,\hat{b}_l}}$.
8. Set $l \leftarrow l - 1$. If $l > 0$, go back to Step 3.
9. Output $(\hat{b}_1, \hat{s}_{1,\hat{b}_1}) \ldots (\hat{b}_n, \hat{s}_{n,\hat{b}_n})$.

Regarding Step 5 above, if $\Delta c_{l,j} = 0$, then $c_l \equiv c'_l (\bmod q)$ holds by the setting of Step 4. However, the probability such that $c_l \equiv c'_l (\bmod q)$ is negligible because it is assumed that $c_l$ and $c'_l$ are randomly selected by $V$. Therefore, in the above procedure, the probability such that $M^{P^*}$ aborts at Step 5 is negligible.

Next, it is shown that the probability such that $M^{P^*}$ aborts at Step 7 is negligible. This means $y_{l,\hat{b}_l} = g^{\hat{s}_{l,\hat{b}_l}}$ holds except a negligible error if $\hat{s}_{l,\hat{b}_l}$ is generated according to (2).

After performing $\pi_{suc}$ and $\pi_n$, the following equations are respectively obtained.

$$g^{z'_{n,j}} = R_j \textstyle\prod_{i=1}^{n} y_{i,j}^{-c_{i,j}} \text{ for } j = 1, 2, \ldots, k \tag{3}$$

$$g^{z_{n,j}} = R_j y_{n_j}^{-c_{n,j}} \textstyle\prod_{i=1}^{n-1} y_{i,j}^{-c_{i,j}} \text{ for } j = 1, 2, \ldots, k \tag{4}$$

From the case of $j = 1$ in (3) and (4), $g^{\Delta z_1} = y_{n,1}^{-\Delta c_{n,1}}$ holds. This is equivalent to $\Delta z_1 = -\Delta c_{n,1} s_{n,1} (\bmod q)$. Similarly, $\Delta z_j = -\Delta c_{n,j} s_{n,j} (\bmod q)$ for $j = 2, 3, \ldots, k$ are obtained from the other cases of the equations. Therefore, for input $\hat{s}_{n,b} = -\Delta z_b / \Delta c_{n,b} \bmod q$ generated according to (2), $y_{n,b} = G^{\hat{s}_{n,b}}$ holds if $\Delta c_{n,b} \neq 0$. Note that, as observed before, there exists $b \in \{1, 2, \ldots, k\}$ such that $\Delta c_{n,b} \neq 0$ with overwhelming probability. On the other hand, $\Delta c_{n,j} = 0$ for $j \in S_n$ holds because of the Rincompatibility of $(y_{n,1}, y_{n,2}, \ldots, y_{n,k})$.

We next consider the case of $l < n$. In this case, we see that

$$\Delta z_j \equiv = -\textstyle\sum_{i=l}^{n} \Delta c_{i,j} s_{i,j} (\bmod q) \tag{5}$$

holds by the same way as the case of $l = n$. Equation (5) can be transformed into

$$s_{l,b} = (\Delta z_b + \textstyle\sum_{i=l+1}^{n} \Delta c_{i,b} s_{i,b}) / - \Delta c_{l,b} \tag{6}$$

for $b \in \{1, 2, \ldots, k\}$ straightforward and (6) is equivalent to (2) if $c_{i,b} = 0$ for $b \neq \hat{b}_i$. Assume that $(\hat{b}_{l+1}, \hat{s}_{l+1,\hat{b}_{l+1}}), \ldots (\hat{b}_n, \hat{s}_{n,\hat{b}_n})$ satisfying $y_{l+1,\hat{b}_{l+1}} = g^{\hat{s}_{l+1,\hat{s}_{l+1}}}, \ldots, y_{n,\hat{b}_n} = g^{\hat{s}_{n,\hat{s}_n}}$ respectively, are already obtained by performing $\pi_{l+1} \ldots \pi_n$ and $\pi_{suc}$. Assume also that $\Delta c_{i,\hat{b}_i} \neq 0$ and $\Delta c_{i,j} = 0$ for $j \neq \hat{b}_i$ and $i = l+1, l+2, \ldots, n$ and $\Delta c_{i,\hat{b}_i}$ are already obtained. It is clear that these assumptions hold when $l = n-1$. Then, there exist $b$ in $\{1, 2, \ldots, k\}$ such that $c_{l,b} \neq 0$ and $y_{l,b} = g^{s_{l,b}}$ can be computed according (6). Then set $(\hat{b}_l, \hat{s}_{l,\hat{b}_l} \leftarrow (b, s_{l,b})$. Since the probability such that $\Delta c_{l,j} = 0$ for $j = 1, 2, \ldots, k$ is negligible as observed before, the probability such that $M^{P^*}$ aborts at Step 7 is negligible in this stage. Finally, it is shown that $\Delta c_{l,1-\hat{b}_l} = 0$ for the validity of the assumptions as stated before. If $\Delta c_{l,1-\hat{b}_l} \neq 0$, $s_{l,1-\hat{b}_l}$ such that $y_{l,1-\hat{b}_l} = g^{s_{l,1-\hat{b}_l}}$ can be computed. However, this contradicts the R-incompatibility of $y_{l,1}, y_{l,2}, \ldots, y_{l,k}$. □

# Optimistic Fair Priced Oblivious Transfer

Alfredo Rial and Bart Preneel

IBBT and Katholieke Universiteit Leuven, ESAT/COSIC
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

**Abstract.** Priced oblivious transfer (POT) is a two-party protocol between a vendor and a buyer in which the buyer purchases digital goods without the vendor learning what is bought. Although privacy properties are guaranteed, current schemes do not offer fair exchange. A malicious vendor can, e.g., prevent the buyer from retrieving the goods after receiving the payment, and a malicious buyer can also accuse an honest vendor of misbehavior without the vendor being able to prove this untrue. In order to address these problems, we define the concept of optimistic fair priced oblivious transfer and propose a generic construction that extends secure POT schemes to realize this functionality. Our construction, based on verifiably encrypted signatures, employs a neutral adjudicator that is only involved in case of dispute, and shows that disputes can be resolved without the buyer losing her privacy, i.e., the buyer does not need to disclose which digital goods she is interested in. We show that our construction can be instantiated with an existing universally composable POT scheme, and furthermore we propose a novel full-simulation secure POT scheme that is much more efficient.

**Keywords:** Priced oblivious transfer, verifiably encrypted signatures, fair exchange.

## 1 Introduction

The protection of privacy in e-commerce is necessary both from a marketing perspective, since privacy concerns discourage buyers from online purchasing [1,2], as well as from a legal perspective, since different authorities have promulgated regulations to enforce the fulfillment of privacy policies and the confidentiality of buyer's personal data [3]. Additionally, buyers, feeling themselves in an unfavorable position at the payment phase, worry that malicious vendors can, e.g., deliver inappropriate or defective goods later on, and thus e-commerce protocols are normally analyzed in order to prove their fairness [4].

Priced oblivious transfer (POT) is a two-party protocol that provides privacy in e-commerce of digital goods by hiding from the vendor which items are bought. More formally, a vendor $\mathcal{V}$ sells a set of messages $m_1, \ldots, m_N$ with prices $p_1, \ldots, p_N$ to a buyer $\mathcal{B}$. At each purchase, $\mathcal{B}$ chooses $\tau \in \{1, \ldots, N\}$, gets $m_\tau$ and pays $p_\tau$ without $\mathcal{V}$ learning any information.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 131–147, 2010.

Existing POT schemes [5,6,7] employ a prepaid mechanism where, initially, $\mathcal{B}$ makes a deposit, and, at each purchase, subtracts $p_\tau$ from the deposit with $\mathcal{V}$ learning neither $p_\tau$ nor the new value of the deposit. Therefore, $\mathcal{B}$ has to trust that $\mathcal{V}$, after receiving the payment, will deliver the requested goods. Furthermore, a malicious $\mathcal{V}$ can claim that $\mathcal{B}$ ran out of funds, and it is impossible for $\mathcal{B}$ to prove this untrue. On the other hand, a malicious $\mathcal{B}$ can claim that $\mathcal{V}$ is not fulfilling his delivery obligations or that her current funds are larger, and again $\mathcal{V}$ cannot prove the contrary. We propose the concept of optimistic fair POT (OFPOT) as a countermeasure.

*Previous Work.* Fair e-commerce is often seen as a particular case of fair exchange. Early work on fair exchange [8,9] follows the approach of dividing the items to be exchanged into small pieces and exchanging them piece by piece. As noted in [10], the resulting protocol is unfair in the exchange of the last piece.

More recent work proposes the involvement of a neutral third party [11], which can be split up into several entities to avoid dependence on the reliability of a single entity [12]. When the third party is only involved in case of dispute, the protocol is called optimistic. Some of these protocols are based on verifiably encrypted signatures [13,14,15]. In a nutshell, $\mathcal{B}$ sends $\mathcal{V}$ a verifiably encrypted signature in her request, and, after $\mathcal{V}$ fulfills his delivery obligations, reveals a valid signature. If $\mathcal{B}$ does not reveal the signature, $\mathcal{V}$ sends the verifiably encrypted signature to the third party, which returns a valid signature after $\mathcal{V}$ demonstrates that he has fulfilled his obligations.

To the best of our knowledge, there were no previous attempts to design a fair e-commerce protocol based on POT. Existing privacy-preserving fair e-commerce protocols [16] provide buyers with anonymity, i.e., they hide from vendors the identity of buyers. As noted in [5,7], anonymous purchase has some disadvantages, like hindering customer management or making the use of currently deployed online payment methods impossible.

*Our Contribution.* We define an ideal functionality for OFPOT and we propose a construction that, taking a secure POT scheme as a building block, turns it into an OFPOT scheme. Our definition and construction involve a neutral third party, an adjudicator $\mathcal{A}$, which is only active in case of dispute.

Our construction is based on the use of verifiably encrypted signatures and, to some extent, resembles non-privacy preserving fair e-commerce protocols based on them. Nevertheless, we note that, since in these protocols privacy is not protected, it is trivial for $\mathcal{V}$ to show to $\mathcal{A}$ that he has fulfilled his obligations, and for $\mathcal{A}$ to verify this fact, because $\mathcal{B}$ discloses which item she requests. One of the main contributions of our work is to show that $\mathcal{A}$ can handle complaints from $\mathcal{V}$ and $\mathcal{B}$ by learning neither the list of items offered by $\mathcal{V}$ nor which ones are requested by $\mathcal{B}$.

To be proven full-simulation secure, our construction has to be instantiated with a full-simulation secure POT scheme. We show that the universally composable scheme in [7] is suitable for such an instantiation. However, the schemes in [5,6] are only half-simulation secure (vendor's security definition follows the

ideal-world/real-world paradigm, but buyer's security definition is an indistinguishability argument [17]), and thus they are not suitable. Since the scheme in [7] is rather inefficient, we propose a novel and efficient full-simulation secure POT scheme based on the oblivious transfer scheme in [18].

*Outline of the Paper.* In Section 2 we recall the ideal functionality for POT given in [7] and we describe our novel POT scheme. In Section 3 we define an ideal functionality for OFPOT and we depict our OFPOT scheme. Finally, Section 4 draws a conclusion and discusses future work.

## 2 Efficient Priced Oblivious Transfer

### 2.1 Technical Preliminaries

A function $\nu$ is *negligible* if, for every integer $c$, there exists an integer $K$ such that for all $k > K$, $|\nu(k)| < 1/k^c$. A problem is said to be *hard* (or *intractable)* if there exists no probabilistic polynomial time (p.p.t.) algorithm that solves it with non-negligible probability (in the size of the input or the security parameter).

*Bilinear maps.* Let $\mathbb{G}$ and $\mathbb{G}_t$ be groups of prime order $p$. A map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_t$ must satisfy bilinearity, i.e., $e(g^x, g^y) = e(g, g)^{xy}$; non-degeneracy, i.e., for all generators $g \in \mathbb{G}$, $e(g, g)$ generates $\mathbb{G}_t$; and efficiency, i.e., there exists an efficient algorithm $\mathsf{BMGen}(1^\kappa)$ that outputs the pairing group setup $(p, \mathbb{G}, \mathbb{G}_t, e, g)$ and an efficient algorithm to compute $e(a, b)$ for any $a, b \in \mathbb{G}$.

**Security Assumptions.** Let $(p, \mathbb{G}, \mathbb{G}_t, e, g)$ be a pairing group setup. The Strong Diffie-Hellman assumption ($l$-SDH) [19] states that, on input $(g, g^x, \ldots, g^{x^l}) \in \mathbb{G}^{l+1}$ for $x \leftarrow \mathbb{Z}_p$, it is hard to output a pair $(c, g^{1/(x+c)})$ for $c \leftarrow \mathbb{Z}_p$. The Power Decisional Diffie-Hellman assumption ($l$-PDDH) [18] states that, on input $(g, g^x, \ldots, g^{x^l}, H) \in \mathbb{G}^{l+1} \times \mathbb{G}_t$ for random $H$ and $x \leftarrow \mathbb{Z}_p$, it is hard to distinguish between the vector $(H^x, H^{x^2}, \ldots, H^{x^l}) \in \mathbb{G}_t^l$ and a random vector $T \in \mathbb{G}_t^l$. As shown in [20], the $l$-PDDH assumption is implied by the $l$-BDHE assumption [21].

**Signature Schemes.** A signature scheme consists of the algorithms $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$. $\mathsf{Kg}(1^\kappa)$ outputs a key pair $(sk, pk)$. $\mathsf{Sign}(sk, m)$ outputs a signature $\sigma$ on message $m$. $\mathsf{Vf}(pk, \sigma, m)$ outputs $\mathsf{accept}$ if $\sigma$ is a valid signature on $m$ and $\mathsf{reject}$ otherwise. A signature scheme must be correct and unforgeable [22]. Informally speaking, correctness implies that $\mathsf{Vf}$ always accepts a valid signature. Unforgeability means that no p.p.t adversary should be able to output a message-signature pair $(\sigma, m)$ unless he has previously obtained a signature on $m$.

We employ the weakly secure signature scheme in [19], which utilizes the SDH assumption. This scheme is existentially unforgeable under a weak chosen message attack, where the adversary submits all signature queries before seeing the public key. The setup consists of the pairing group setup $(p, \mathbb{G}, \mathbb{G}_t, e, g)$.

$\mathsf{WKg}(1^\kappa)$ picks random $x \leftarrow \mathbb{Z}_p$ and outputs a key pair $(sk, pk) = (x, g^x)$. $\mathsf{WSign}(sk, m)$ computes $\sigma = g^{1/(x+m)}$. $\mathsf{WVf}(pk, \sigma, m)$ outputs $\mathsf{accept}$ if $e(\sigma, pk \cdot g^m) = e(g, g)$.

**Commitment schemes.** A non-interactive commitment scheme consists of the algorithms $(\mathsf{ComS}, \mathsf{Com}, \mathsf{Open})$. $\mathsf{ComS}(1^\kappa)$ generates the parameters of the commitment scheme $par_{com}$. $\mathsf{Com}(par_{com}, x, open)$ outputs a commitment $C$ to $x$ using auxiliary information $open$. A commitment is opened by revealing $(x, open)$ and checking whether $\mathsf{Open}(par_{com}, C, x, open)$ outputs $\mathsf{accept}$. A commitment scheme has a hiding property and a binding property. Informally speaking, the hiding property ensures that a commitment $C$ to $x$ does not reveal any information about $x$, whereas the binding property ensures that $C$ cannot be opened to another value $x'$. When it is clear from the context, we omit the commitment parameters $par_{com}$.

We employ the commitment scheme proposed by Pedersen [23]. $\mathsf{PEComS}(1^\kappa)$ picks random generators $g, h$ of a group $\mathbb{G}$ of prime order $p$ and outputs $par_{com} = (g, h)$. $\mathsf{PECom}(par_{com}, x, open_x)$ outputs $C = g^x h^{open_x}$ on input $x \in \mathbb{Z}_p$ and randomness $open_x \in \mathbb{Z}_p$. $\mathsf{PEOpen}(par_{com}, C, x', open'_x)$ outputs $\mathsf{accept}$ if $C = g^{x'} h^{open'_x}$. This scheme is information theoretically hiding and computationally binding under the discrete logarithm assumption.

**Proofs of Knowledge.** A zero-knowledge proof of knowledge [24] is a two-party protocol between a prover and a verifier. The prover proves to the verifier knowledge of some secret input that fulfills some statement without disclosing this input to the verifier. The protocol should fulfill two properties. First, it should be a proof of knowledge, i.e., a prover without knowledge of the secret input convinces the verifier with negligible probability. More technically, there exists a knowledge extractor that extracts the secret input from a successful prover with all but negligible probability. Second, it should be zero-knowledge, i.e., the verifier does not learn any information about the secret input. More technically, for all possible verifiers there exists a simulator that, without knowledge of the secret input, yields a distribution that cannot be distinguished from the interaction with a real prover.

We use several existing results to prove statements about discrete logarithms: (1) proof of knowledge of a discrete logarithm modulo a prime [25]; (2) proof of knowledge of the equality of some element in different representations [26]; (3) proof of knowledge that a value $\alpha$ lies in a given interval $[0, A)$ [27]; and (4) proof of the disjunction or conjunction of any two of the previous [28]. These results are often given in the form of $\Sigma$-protocols but they can be turned into zero-knowledge protocols using efficient zero-knowledge compilers [29,30], and they can be turned into non-interactive proofs in the random oracle model via the Fiat-Shamir heuristic [31].

When referring to the proofs above, we follow the notation introduced by Camenisch and Stadler [32] for various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. $\mathsf{NIPK}\{(\alpha, \beta, \delta) : y = g^\alpha h^\beta \wedge \tilde{y} = \tilde{g}^\alpha \tilde{h}^\delta \wedge A \leq \alpha \leq B\}$ denotes a "*non-interactive*

*zero-knowledge proof of knowledge of integers* $\alpha$, $\beta$, *and* $\delta$ *such that* $y = g^{\alpha}h^{\beta}$, $\tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\delta}$ *and* $A \leq \alpha \leq B$ *holds*", where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$ that have the same order. (Note that some elements in the representation of $y$ and $\tilde{y}$ are equal.) The convention is that letters in the parenthesis, in this example $\alpha$, $\beta$, and $\delta$, denote quantities whose knowledge is being proven, while all other values are known to the verifier.

We employ the range proof proposed in [27] to prove that a value $\alpha$ lies in an interval $[0, d^a)$. The proof has an initialization phase where the verifier provides the prover with signatures that allow the prover to write the value $\alpha$ in base-$d$. First, the verifier runs $\mathsf{InitVer}(1^{\kappa}, D_{max})$, which runs $\mathsf{WKg}(1^{\kappa})$ to get a key pair $(sk, pk)$ and computes signatures $A_i = \mathsf{WSign}(sk, i)$ on $d$-ary digits, i.e., $i \in \mathbb{Z}_d$. It also runs $\mathsf{PEComS}(1^{\kappa})$ to get $par_{com}$ and sets $par_{ran} = (pk, \{A_i\}_{i \in \mathbb{Z}_d}, par_{com})$. The verifier sends $par_{ran}$ to the prover. The prover runs $\mathsf{InitP}(par_{ran})$ to verify the signatures by running, for $i \in \mathbb{Z}_d$, $\mathsf{WVf}(pk, A_i, i)$. After this initialization phase, prover and verifier can run multiple proofs. The verifier receives as input a commitment $C$, and the prover values $(\alpha, open_{\alpha})$ such that $C = \mathsf{PEComS}(par_{com}, \alpha, open_{\alpha})$. The prover runs $\mathsf{RProve}(par_{ran}, \alpha, open_{\alpha})$, which picks $v_j \leftarrow \mathbb{Z}_p$ and computes $V_j = A_{\alpha_j}^{v_j}$ for every $j \in \mathbb{Z}_a$ such that $\alpha = \sum_{j \in \mathbb{Z}_a} \alpha_j d^j$, and a proof $pok = \mathsf{NIPK}\{(open_{\alpha}, \{\alpha_j, v_j\}_{j \in \mathbb{Z}_a}) : C = h^{open_{\alpha}} \prod_{j \in \mathbb{Z}_a} (g^{d^j})^{\alpha_j} \wedge V_j = g^{v_j/(sk+\alpha_j)}\}$. (We abbreviate it as $\mathsf{NIPK}\{(\alpha, open_{\alpha}) : 0 \leq \alpha < d^a \wedge C = \mathsf{PEComS}(par_{com}, \alpha, open_{\alpha})\}$.) The prover sends $(\{V_j\}_{j \in \mathbb{Z}_a}, pok)$ to the verifier.

**Public Key Encryption.** A public key encryption scheme consists of the algorithms $(\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$. $\mathsf{Kg}(1^{\kappa})$ outputs a key pair $(sk, pk)$. $\mathsf{Enc}(pk, m)$ outputs a ciphertext $ct$ on input $pk$ and a message $m$. $\mathsf{Dec}(sk, ct)$ outputs the message $m$.

### 2.2 Definition

We define security following the ideal-world/real-world paradigm [33]. In the real world, a set of parties interact according to the protocol description in the presence of a real adversary $\mathcal{E}$, while in the ideal world dummy parties interact with an ideal functionality that carries out the desired task in the presence of an ideal adversary $\mathcal{S}$. A protocol $\psi$ is secure if there exists no environment $\mathcal{Z}$ that can distinguish whether it is interacting with adversary $\mathcal{E}$ and parties running protocol $\psi$ or with the ideal process for carrying out the desired task, where ideal adversary $\mathcal{S}$ and dummy parties interact with an ideal functionality $\mathcal{F}_{\psi}$. More formally, we say that protocol $\psi$ emulates the ideal process when, for any adversary $\mathcal{E}$, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$, the ensembles $\mathrm{IDEAL}_{\mathcal{F}_{\psi},\mathcal{S},\mathcal{Z}}$ and $\mathrm{REAL}_{\psi,\mathcal{E},\mathcal{Z}}$ are computationally indistinguishable. We refer to [33] for a description of how these ensembles are constructed.

We recall the ideal functionality $\mathcal{F}_{\mathrm{POT}}$ for priced oblivious transfer in [7]. Every functionality and every protocol invocation should be instantiated with a unique session-ID that distinguishes it from other instantiations. For the sake of ease of notation, we omit session-IDs from our description.

**Functionality $\mathcal{F}_{\text{POT}}$**

Parameterized with the number of messages $N$, the message length $l$, the maximum price $p_{max}$, and the maximum deposit $D_{max}$, and running with a vendor $\mathcal{V}$ and a buyer $\mathcal{B}$, $\mathcal{F}_{\text{POT}}$ works as follows:

- On input a message $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$ from $\mathcal{V}$, where each $m_i \in \{0,1\}^l$ and each $p_i \in [0, p_{max}]$, it stores $(m_1, p_1, \ldots, m_N, p_N)$ and sends $(\text{init}, p_1, \ldots, p_N)$ to $\mathcal{B}$.
- On input a message $(\text{deposit}, ac)$, where $ac \in [0, D_{max})$, if a $(\text{init}, \ldots)$ message was not received before, then it does nothing. Otherwise, it stores $ac$ and sends $(\text{deposit}, ac)$ to $\mathcal{V}$.
- On input a message $(\text{request}, \tau)$ from $\mathcal{B}$, where $\tau \in \{1, \ldots, N\}$, if either messages $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$ and $(\text{deposit}, ac)$ were not received before or $ac - p_\tau < 0$, then it does nothing. Otherwise, it sends (request) to $\mathcal{V}$ and receives $(\text{response}, b)$ in response. If $b = 0$, it sends $(\text{response}, \perp)$ to $\mathcal{B}$. If $b = 1$, it updates $ac = ac - p_\tau$ and sends $(\text{response}, m_\tau)$ to $\mathcal{B}$.

### 2.3 Intuition Behind Our Construction

Our POT scheme is based on the adaptive oblivious transfer scheme in [18]. It is an assisted decryption scheme in which there is an initialization phase and several purchase phases. At the initialization phase, $\mathcal{V}$ sends $\mathcal{B}$ a collection of ciphertexts that encrypt the messages $m_1, \ldots, m_N$ to be sold, and at each purchase $\mathcal{V}$ helps $\mathcal{B}$ to decrypt one of them. As noted in [34], this design approach leads to purchase phases with constant communication and computation complexity, and ensures that $\mathcal{V}$ cannot modify the messages after the initialization phase.

Each ciphertext $C_i$ consists of a unique price[1] $p_i$, a signature on the price $A_i$ and an encryption $B_i = e(h, A_i) \cdot m_i$, where $h$ is the secret key of $\mathcal{V}$. To compute a request for $m_\tau$, $\mathcal{B}$ sends $\mathcal{V}$ a blinded value $V = A_\tau^v$ together with a zero-knowledge proof that $\mathcal{B}$ possesses a valid signature $A_\tau$ and that $V$ is correctly computed. $\mathcal{V}$ computes a blinded decryption $W = e(h, V)$ and proves that the secret key $h$ was used to compute $W$. Finally, $\mathcal{B}$ unblinds $W$ to decrypt the message $m_\tau$ via $B_\tau/(W^{1/v})$.

To allow for oblivious payments, our POT scheme follows the prepaid mechanism by [7]. At the initialization phase, $\mathcal{B}$ makes a deposit $ac_0$ and sends a commitment $D_0$ to the deposit and its opening to $\mathcal{V}$. At purchase phase $i$, $\mathcal{B}$ sends a commitment $D_i$ to the new value of the account $ac_i$ along with a zero-knowledge proof that $ac_i = ac_{i-1} - p_\tau$ and that $ac_i \in [0, D_{max})$, to prove that the account is correctly updated and that $\mathcal{B}$ has enough funds to buy $m_\tau$. For the latter, we employ the range proof recently proposed in [27].

[1] If several messages have the same price, unique prices can be obtained by adequately scaling prices and accounts [5].

We prove that our scheme realizes $\mathcal{F}_{\mathrm{POT}}$ under the SDH and PDDH assumptions (used in [18]), and under the assumption the binding and hiding properties of the commitment scheme hold. In the description of the scheme given below, we compute non-interactive proofs via Fiat-Shamir heuristic [31] and thus the construction is secure in the random oracle model [35]. This is convenient when using the scheme as a building block of the OFPOT scheme depicted in Section 3, because it allows $\mathcal{A}$ to verify requests and responses non-interactively. Nevertheless, for other uses of the scheme interactive proofs of knowledge can be employed, yielding a construction in the standard model.

### 2.4 Description of the Scheme

We begin with a high level description of the POT scheme. Details on the algorithms can be found below.

**POT**

**Initialization phase.** On input $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$, in which each price is unique, $\mathcal{V}$ runs $\mathsf{POTInitV}(1^\kappa, m_1, p_1, \ldots, m_N, p_N, D_{max})$ in order to obtain a database commitment $T$ and a key pair $(pk_{\mathcal{V}}, sk_{\mathcal{V}})$, and sends $(pk_{\mathcal{V}}, T)$ to $\mathcal{B}$. On input $(\text{deposit}, ac_0)$, $\mathcal{B}$ runs $(P, D_0^{'}) \leftarrow \mathsf{POTInitB}(1^\kappa, pk_{\mathcal{V}}, T, ac_0)$ and aborts if the output is reject. Otherwise, $\mathcal{B}$ sends the payment message $(P)$ to $\mathcal{V}$ and pays an amount of $ac_0$ through an arbitrary payment channel. $\mathcal{V}$ runs $(D_0, ac_0) \leftarrow \mathsf{POTGetDep}(sk_{\mathcal{V}}, P, D_{max})$ and checks that $ac_0$ corresponds to the amount of money received. $\mathcal{V}$ stores state information $V_0 = (T, sk_{\mathcal{V}}, pk_{\mathcal{V}}, D_0)$ and outputs $(\text{deposit}, ac_0)$, and $\mathcal{B}$ stores state information $B_0 = (pk_{\mathcal{V}}, T, D_0^{'})$.

**Transfer phase.** On input $(\text{request}, \tau_i)$, $\mathcal{B}$ runs $\mathsf{POTReq}(pk_{\mathcal{V}}, T, D_{i-1}^{'}, \tau_i)$ to get a request $Q$ and private state $(Q^{'}, D_i^{'})$. $\mathcal{B}$ sends $(Q)$ and stores $(Q^{'}, D_i^{'})$. On input $(\text{response}, b)$, if $b = 0$ $\mathcal{V}$ sends $(\perp)$ to $\mathcal{B}$. If $b = 1$, $\mathcal{V}$ runs $\mathsf{POTVerReq}(pk_{\mathcal{V}}, D_{i-1}, Q)$ and ignores the request if the output is reject. Otherwise $\mathcal{V}$ runs $\mathsf{POTResp}(pk_{\mathcal{V}}, sk_{\mathcal{V}}, Q)$ to obtain a response $R$ and state $D_i$, and sends $(R)$ to $\mathcal{B}$. $\mathcal{B}$ runs $\mathsf{POTVerResp}(pk_{\mathcal{V}}, R)$ and outputs $(\text{response}, \perp)$ it the output is reject. Otherwise $\mathcal{B}$ runs $\mathsf{POTComplete}(T, R, Q^{'})$ to obtain $m_{\tau_i}$. $\mathcal{V}$ stores state information $V_i = (sk_{\mathcal{V}}, T, pk_{\mathcal{V}}, D_i)$, and $\mathcal{B}$ stores state information $B_i = (T, pk_{\mathcal{V}}, D_i^{'})$ and outputs $(\text{response}, m_{\tau_i})$.

$\mathsf{POTInitV}(1^\kappa, m_1, p_1, \ldots, m_N, p_N, D_{max})$ computes a pairing group setup $\Phi = (p, \mathbb{G}, \mathbb{G}_t, e, g)$, picks a random generator $h \in \mathbb{G}$ and sets $H = e(g, h)$. It runs $\mathsf{InitVer}(1^\kappa, D_{max})$ to obtain $par_{ran}$ (which includes parameters of the

commitment scheme $par_{com}$) and $\mathsf{Kg}(1^\kappa)$ to obtain $(sk_{enc}, pk_{enc})$.[2] It also runs $\mathsf{WKg}(1^\kappa)$ to obtain a key pair $(sk, pk)$ for the signature scheme, and, for $i = 1$ to $N$, it computes $A_i = \mathsf{WSign}(sk, p_i)$ and $B_i = e(h, A_i) \cdot m_i$. It sets $C_i = (A_i, B_i, p_i)$. It computes a non-interactive proof $\pi_1 = \mathsf{NIPK}\{(h) : H = e(g, h)\}$. It outputs $sk_{\mathcal{V}} = (sk_{enc}, h)$, $pk_{\mathcal{V}} = (\Phi, H, par_{ran}, pk, pk_{enc}, \pi_1)$ and $T = (C_1, \ldots, C_N)$.

$\mathsf{POTInitB}(1^\kappa, pk_{\mathcal{V}}, T, ac_0)$ parses, for $i = 1$ to $N$, $C_i$ as $(A_i, B_i, p_i)$, and checks whether $\mathsf{WVf}(pk, A_i, p_i)$ outputs accept. It also verifies $\pi_1$ and checks $par_{ran}$ via $\mathsf{InitP}(par_{ran})$. Then it runs $\mathsf{Enc}(pk_{enc}, ac_0)$ to obtain an encryption $ct$ and computes a commitment $D_0 = \mathsf{PECom}(par_{com}, ac_0, open_{ac_0})$ for random $open_{ac_0}$.[3] It sets $P = (ct, open_{ac_0})$ and $D_0^{'} = (ac_0, open_{ac_0}, D_0)$. It outputs $(P, D_0^{'})$.

$\mathsf{POTGetDep}(sk_{\mathcal{V}}, P, D_{max})$ runs $\mathsf{Dec}(sk_{enc}, ct)$ to obtain $ac_0$ and checks that $ac_0 \in [0, D_{max})$. It computes $D_0 = \mathsf{PECom}(par_{com}, ac_0, open_{ac_0})$ and outputs $(D_0, ac_0)$.

$\mathsf{POTReq}(pk_{\mathcal{V}}, T, D_{i-1}^{'}, \tau)$ calculates $ac_i = ac_{i-1} - p_\tau$, picks random $(open_p, open_{ac_i}) \leftarrow \mathbb{Z}_p$, and computes $D_p = \mathsf{PECom}(par_{com}, p_\tau, open_p)$ and $D_i = \mathsf{PECom}(par_{com}, ac_i, open_{ac_i})$. It picks random $v \in \mathbb{Z}_p$ and computes $V = A_\tau^v$. It computes a non-interactive proof[4] $\pi_2$:

$$
\begin{aligned}
&\mathsf{NIPK}\{(p_\tau, open_p, ac_i, open_{ac_i}, v, \alpha) : \\
&\quad e(V, pk) = e(V, g)^{-p_\tau} e(g, g)^v \wedge && (1)\\
&\quad D_p = \mathsf{PECom}(par_{com}, p_\tau, open_p) \wedge && (2)\\
&\quad D_i = \mathsf{PECom}(par_{com}, ac_i, open_{ac_i}) \wedge && (3)\\
&\quad D_{i-1}/(D_i D_p) = \tilde{h}^\alpha \wedge && (4)\\
&\quad 0 \leq ac_i < D_{max}\}. && (5)
\end{aligned}
$$

Equation 1 proves that $V = A_\tau^v$ and that $A_\tau$ is a signature computed by $\mathcal{V}$. Equations 2 and 3 prove knowledge of the committed price $p_\tau$ and $ac_i$ respectively, where $p_\tau$ is the value signed in $A_\tau$. Equation 4 proves that $ac_i = ac_{i-1} - p_\tau$, and Equation 5 proves that $ac_i$ is non-negative. The algorithm outputs $Q = (V, D_p, D_i, \pi_2)$, $Q^{'} = (v, \tau)$ and $D_i^{'} = (ac_i, open_{ac_i}, D_i)$.

$\mathsf{POTVerReq}(pk_{\mathcal{V}}, D_{i-1}, Q)$ verifies $\pi_2$ and outputs either accept or reject.

$\mathsf{POTResp}(pk_{\mathcal{V}}, sk_{\mathcal{V}}, Q)$ parses $Q$ as $(V, D_p, D_i, \pi_2)$ and computes $W = e(h, V)$ and a proof $\pi_3 = \mathsf{NIPK}\{(h) : H = e(g, h) \wedge W = e(V, h)\}$ that $W$ is computed correctly by using the request $V$ and the secret key $h$. It outputs $R = (W, \pi_3)$ and $D_i$.

$\mathsf{POTVerResp}(pk_{\mathcal{V}}, R)$ verifies $\pi_3$ and outputs either accept or reject.

$\mathsf{POTComplete}(T, R, Q^{'})$ parses $Q^{'}$ as $(v, \tau)$, $R$ as $(W, \pi_3)$ and $C_\tau$ as $(A_\tau, B_\tau, p_\tau)$. It outputs $m_\tau = B_\tau/(W^{1/v})$.

[2] We can employ any IND-CPA secure encryption scheme.

[3] Although it would be possible to use a fixed value $open_{ac_0}$, this allows $\mathcal{B}$ to hide her deposit from the adjudicator in the OFPOT scheme.

[4] $\tilde{h}$ is included in the parameters of the commitment scheme $par_{com} = (\tilde{g}, \tilde{h})$.

**Theorem 1.** *This POT scheme securely realizes* $\mathcal{F}_{\text{POT}}$.

We prove this theorem in the full version [36]. The proof follows the proof of the oblivious transfer scheme in [18] and of the priced oblivious transfer scheme in [7]. Buyer security holds under the assumption that the commitment scheme is hiding and under the extractability and zero-knowledge properties of proofs of knowledge. Vendor security holds under the $(N+1, d+1)$-SDH assumption, under the $(N+1)$-PDDH assumption and under the binding property of the commitment scheme. Security is proven under static corruptions.

As for efficiency, we note that the communication and computation cost of our protocol is roughly that of the OT scheme in [18], plus the overhead introduced by the range proof in [27]. We refer to [34] for a comparison of the efficiency of several OT schemes, and to [27] for efficiency measurements of the range proof.

## 3 An Optimistic Fair POT Scheme

### 3.1 Technical Preliminaries

**Verifiably Encrypted Signatures.** A verifiably encrypted signature (VES) scheme consists of algorithms $(\mathsf{Kg}, \mathsf{AdjKg}, \mathsf{Sign}, \mathsf{Vf}, \mathsf{Create}, \mathsf{VesVf}, \mathsf{Adj})$. Algorithm $\mathsf{Kg}(1^\kappa)$ outputs a key pair $(sk, pk)$. $\mathsf{AdjKg}(1^\kappa)$ outputs a key pair $(ask, apk)$ for the adjudicator. $\mathsf{Sign}(sk, m)$ computes a signature $\sigma$ under $sk$ on a message $m$. $\mathsf{Vf}(pk, \sigma, m)$ outputs $\mathsf{accept}$ if $\sigma$ is a valid signature on $m$ under $pk$. $\mathsf{Create}(sk, apk, m)$ outputs a verifiably encrypted signature $\omega$ on input the secret key $sk$, the adjudicator's public key $apk$ and the message $m$. $\mathsf{VesVf}(pk, apk, \omega, m)$ outputs $\mathsf{accept}$ if $\omega$ is a valid verifiably encrypted signature on input the public key $pk$, the adjudicator's public key $apk$ and the message $m$. $\mathsf{Adj}(pk, ask, apk, \omega, m)$ outputs an ordinary signature $\sigma$ on $m$ if $\omega$ is valid.

A VES scheme is complete if, for all key pairs $(sk, pk)$ computed by $\mathsf{Kg}(1^\kappa)$ and for all key pairs $(ask, apk)$ computed by $\mathsf{AdjKg}(1^\kappa)$, it holds that $\mathsf{VesVf}(pk, apk, \mathsf{Create}(sk, apk, m), m)$ outputs $\mathsf{accept}$ and algorithm $\mathsf{Vf}(pk, \mathsf{Adj}(pk, ask, apk, \mathsf{Create}(sk, apk, m), m), m)$ outputs $\mathsf{accept}$.

Security for verifiably encrypted signatures [13,15] is defined via four properties: unforgeability, opacity, extractability and abuse-freeness. Roughly speaking, unforgeability ensures that it is intractable to compute a verifiably encrypted signature on behalf of another user. Opacity ensures that nobody but the adjudicator and the signer can obtain an ordinary signature from a verifiably encrypted signature. Extractability means that the adjudicator is able to extract an ordinary signature from a valid verifiably encrypted signature with all but negligible probability. Abuse-freeness prevents an adversary that colludes with the adjudicator from forging verifiably encrypted signatures.

### 3.2 Definition

Previous work on defining optimistic fair exchange in the ideal-world/real-world paradigm only covers the case where buyer's privacy is not protected [37]. We

define an ideal functionality $\mathcal{F}_{\text{OFPOT}}$ for OFPOT that extends $\mathcal{F}_{\text{POT}}$. It describes the functionality provided by our construction when the adjudicator $\mathcal{A}$ is neutral, and we prove security under this restriction. (We explain more intuitively the functionality provided by $\mathcal{F}_{\text{OFPOT}}$ in Section 3.3.) We also recall the description of a functionality $\mathcal{F}_{\text{CRS}}$ given in [38], which is used when the POT scheme used to instantiate the OFPOT scheme operates in the $\mathcal{F}_{\text{CRS}}$-hybrid model, where parties have access to an honestly-generated common reference string.

**Functionality $\mathcal{F}_{\text{CRS}}$**

Parameterized with a distribution $D$ and a set of participants $\mathcal{P}$, on input (crs) from party $P$, if $P \notin \mathcal{P}$ it aborts. Otherwise, if there is no value $r$ recorded, it picks $r \leftarrow D$ and records $r$. It sends $(\text{crs}, r)$ to $P$.

**Functionality $\mathcal{F}_{\text{OFPOT}}$**

Parameterized with $(N, l, p_{max}, D_{max})$, and running with a vendor $\mathcal{V}$, a buyer $\mathcal{B}$ and an adjudicator $\mathcal{A}$, $\mathcal{F}_{\text{OFPOT}}$ works as follows:

- On input a message $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$ from $\mathcal{V}$, it behaves as $\mathcal{F}_{\text{POT}}$.
- On input a message $(\text{deposit}, ac_0)$ from $\mathcal{B}$, it behaves as $\mathcal{F}_{\text{POT}}$.
- On input a message $(\text{request}, i, \tau)$ from $\mathcal{B}$, where $\tau \in \{1, \ldots, N\}$, if messages $(\text{init}, \ldots)$ or $(\text{deposit}, \ldots)$ were not received before, $i$ does not equal the number of the transfer $t$, $ac_i - p_\tau < 0$, or $dispute_i = 1$, then it does nothing. Otherwise, it sends $(\text{request}, i)$ to $\mathcal{V}$ and receives $(\text{response}, i, b)$ in response. It stores $(i, \tau, b)$. If $b = 0$, it sends $(\text{response}, i, \perp)$ to $\mathcal{B}$. If $b = 1$, it sends $(\text{response}, i, m_\tau)$ to $\mathcal{B}$. $\mathcal{B}$ returns a message $(\text{complete}, i, c)$, which is handed to $\mathcal{V}$. If $c = 1$, it sets $ac_i = ac_{i-1} - p_\tau$ and updates the number of transfers $t = t + 1$. If $c = 0$, it sets $dispute_i = 1$.
- On input a message $(\text{complain}_{\mathcal{V}}, i)$ from $\mathcal{V}$, if $dispute_i = 0$ or $t \neq i$, then it sends $(\text{compInv}_{\mathcal{V}})$ to $\mathcal{V}$ and $(\text{complain}_{\mathcal{V}}, i, \text{inv})$ to $\mathcal{A}$. Otherwise it sets $ac_i = ac_{i-1} - p_\tau$, updates the number of transfers $t = t + 1$ and sends $(\text{compSolved}_{\mathcal{V}})$ to $\mathcal{V}$, $(\text{compResp}_{\mathcal{V}}, i, m_\tau)$ to $\mathcal{B}$ and $(\text{complain}_{\mathcal{V}}, i, \text{solved})$ to $\mathcal{A}$.
- On input a message $(\text{complain}_{\mathcal{B}}, i, \tau)$ from $\mathcal{B}$, if $t \neq i$ or if there exists a tuple $(i, \tau, b)$ and $\tau \neq \tau'$, then it sends $(\text{compInv}_{\mathcal{B}})$ to $\mathcal{B}$ and $(\text{complain}_{\mathcal{B}}, i, \text{inv})$ to $\mathcal{A}$. Otherwise it sends $(\text{complain}_{\mathcal{B}}, i)$ to $\mathcal{V}$. Upon receiving $(\text{collaborate}, i, b)$ from $\mathcal{V}$, if $b = 0$ it sends $(\text{guilty}_{\mathcal{V}})$ to $\mathcal{V}$ and $\mathcal{B}$ and $(\text{complain}_{\mathcal{B}}, i, \text{guilty}_{\mathcal{V}})$ to $\mathcal{A}$. If $b = 1$ it sets $ac_i = ac_{i-1} - p_\tau$, updates the number of transfers $t = t + 1$, sends $(\text{compResp}_{\mathcal{B}}, i, m_\tau)$ to $\mathcal{B}$, $(\text{compSolved}_{\mathcal{B}})$ to $\mathcal{V}$ and $(\text{complain}_{\mathcal{B}}, i, \text{solved})$ to $\mathcal{A}$.

### 3.3 Intuition Behind Our Construction

Our OFPOT scheme extends a full-simulation secure POT scheme with a fair exchange protocol based on verifiably encrypted signatures. In a nutshell, at a purchase phase, $\mathcal{B}$ computes her request following the POT scheme and a verifiably encrypted signature that signs the request. Then $\mathcal{V}$ computes a response following the POT scheme. If $\mathcal{B}$ is satisfied, $\mathcal{B}$ reveals a valid signature on her request, and otherwise $\mathcal{B}$ complains. $\mathcal{V}$ also complains when he computed his response honestly and $\mathcal{B}$ did not reveal a valid signature.

More concretely, the initialization phase follows that of the POT scheme. Additionally, $\mathcal{B}$ signs the payment message she sends to $\mathcal{V}$, $\mathcal{A}$ hands its public key to $\mathcal{V}$ and $\mathcal{B}$, and $\mathcal{A}$ receives the public keys of $\mathcal{V}$ and $\mathcal{B}$. At purchase phase $i$, $\mathcal{B}$ computes a request $Q_i$ following algorithm $\mathsf{POTReq}$ of the POT scheme, and computes a verifiably encrypted signature $\omega_i$ on $H(i, Q_i)$, where $H$ is a collision-resistant hash function. $\mathcal{V}$ rejects the request if $i$ is not the valid purchase index, i.e., if $\mathcal{V}$ did not receive a signature $\sigma_i$ on $H(i-1, Q_{i-1})$ or if $\mathcal{V}$ already received a signature on $H(i, Q_i')$. Otherwise $\mathcal{V}$ computes a response $R$ following algorithm $\mathsf{POTResp}$ of the POT scheme. Finally, $\mathcal{B}$ obtains the message $m_\tau$ by running algorithm $\mathsf{POTComplete}$ and reveals to $\mathcal{V}$ a signature $\sigma_i$ on $H(i, Q_i)$.[5]

When $\mathcal{B}$ does not reveal $\sigma_i$, $\mathcal{V}$ complains by sending to $\mathcal{A}$ the request $(i, Q_i, \omega_i)$ and the response $R$. $\mathcal{A}$ verifies both request and response, sends $R$ to $\mathcal{B}$, extracts $\sigma_i$ from $\omega_i$ and sends $\sigma_i$ to $\mathcal{V}$. Verification by $\mathcal{A}$ can be done without knowing the choice $\tau$ of $\mathcal{B}$. We note that a malicious $\mathcal{V}$ cannot produce fake requests on behalf of $\mathcal{B}$ thanks to the unforgeability property of the VES scheme. Moreover, if $\mathcal{B}$ did not reveal $\sigma_i$ because the response $R'$ from $\mathcal{V}$ was not correct, after $\mathcal{V}$ complains $\mathcal{A}$ ensures that $\mathcal{B}$ receives a valid response.

When the response sent by $\mathcal{V}$ is not correct, $\mathcal{B}$ complains by sending to $\mathcal{A}$ a request $(i, Q_i, \omega_i)$.[6] $\mathcal{A}$ tells $\mathcal{V}$ to send $(j, Q_j, \sigma_j)$ for $j = i - 1$ in order to check if the complaint is valid. (When $i = 1$, $\mathcal{V}$ should send the signature on the payment message.) If $\mathcal{V}$ sends it for $j \geq i$, then $\mathcal{A}$ states that the complaint is invalid because this means that $\mathcal{B}$ has already shown conformity for purchase $i$, and if $j < i - 1$, then it asks $\mathcal{B}$ to send a request with index $j$.[7] If $\mathcal{V}$ does not reveal any signature, $\mathcal{A}$ finds $\mathcal{V}$ guilty. (We note that $\mathcal{V}$ cannot produce signatures on behalf of $\mathcal{B}$ thanks to the opacity property of the VES scheme.) Otherwise, $\mathcal{A}$ verifies the request from $\mathcal{B}$ and sends it to $\mathcal{V}$. If $\mathcal{V}$ returns a different tuple $(i, Q_i', \omega_i')$, then $\mathcal{A}$ states that the complaint is invalid because this may mean that a malicious $\mathcal{B}$ already obtained the message requested in $Q_i'$, and is trying to obtain a different message. Otherwise, if $\mathcal{V}$ returns a valid response $R$, $\mathcal{A}$ forwards $R$ to $\mathcal{B}$ and reveals $\sigma_i$ to $\mathcal{V}$. If not, $\mathcal{A}$ finds $\mathcal{V}$ guilty.

[5] In practical implementations, this message can be sent together with the next request in order to save communication rounds.

[6] We note that $\mathcal{B}$ can complain without previously interacting with $\mathcal{V}$, but this does not give a malicious $\mathcal{B}$ any advantage over $\mathcal{V}$.

[7] We note that $\mathcal{V}$ should always send the larger index $j$ for which he possesses a signature, because otherwise he would be claiming that some purchases did not happen, thus increasing the account of $\mathcal{B}$.

We prove that our scheme realizes $\mathcal{F}_{\text{OFPOT}}$ if it is instantiated with a full-simulation secure POT scheme (which realizes $\mathcal{F}_{\text{POT}}$) and under the assumptions that the VES scheme is unforgeable, opaque and extractable, and that $\mathcal{A}$ is neutral. The abuse-freeness property of the VES scheme gives $\mathcal{B}$ some protection when $\mathcal{V}$ and $\mathcal{A}$ collude: they cannot produce fake requests. However, once $\mathcal{B}$ sends a request, a malicious $\mathcal{A}$ can reveal $\sigma_i$ to $\mathcal{V}$ without sending $R$ to $\mathcal{B}$. (Nevertheless, privacy of $\mathcal{B}$ is still preserved.) Similarly, when $\mathcal{B}$ and $\mathcal{A}$ collude, $\mathcal{V}$ is not protected.

When our construction is instantiated with the universally composable POT scheme in [7] (which we summarize in the full version [36]), we obtain a UC secure construction in the common reference string model. (In the description given below, we include the common reference string as input of the algorithms, but it should be removed when unnecessary.) When instantiated with the POT scheme proposed in Section 2, we obtain a full-simulation secure and more efficient scheme.

Finally, we point out that our scheme does not ensure that all the buyers receive the same messages, or that the messages that $\mathcal{V}$ sells fulfill the expectations of $\mathcal{B}$. Nevertheless, there already exist countermeasures which employ a third party against these problems [39], and $\mathcal{A}$ can apply them. Basically, these countermeasures ensure that all the buyers obtain the same messages from $\mathcal{V}$. We can achieve this by ensuring that they obtain the same database commitment $T$.

### 3.4 Description of the Scheme

We begin with a high level description of the OFPOT scheme. Details on the algorithms can be found below. We recall that the scheme is parameterized with $(N, l, p_{max}, D_{max})$.

**OFPOT**

**Initialization phase.** On input $(\text{init}, m_1, p_1, \ldots, m_N, p_N)$, $\mathcal{V}$ queries $\mathcal{F}_{\text{CRS}}$ with (crs), which runs $\mathsf{POTGenCRS}(1^\kappa, p_{max}, D_{max})$ and returns (crs, $crs$). $\mathcal{B}$ and $\mathcal{A}$ also query $\mathcal{F}_{\text{CRS}}$ with (crs), which returns (crs, $crs$). $\mathcal{V}$ runs $\mathsf{POTInitV}(crs, m_1, p_1, \ldots, m_N, p_N, D_{max})$ to get a database commitment $T$ and a key pair $(sk_{\mathcal{V}}, pk_{\mathcal{V}})$, and sends $(pk_{\mathcal{V}}, T)$ to $\mathcal{B}$. On input $(\text{deposit}, ac_0)$, $\mathcal{B}$ computes $(P, D_0^{'}) \leftarrow \mathsf{POTInitB}(crs, pk_{\mathcal{V}}, T, ac_0)$ to obtain a payment message $P$ and the opening of the commitment to the account $D_0^{'}$, and aborts if the output is reject. Otherwise, $\mathcal{B}$ runs $(sk_{\mathcal{B}}, pk_{\mathcal{B}}, \sigma_0) \leftarrow \mathsf{OFInitB}(crs, D_0^{'})$ to obtain a key pair $(sk_{\mathcal{B}}, pk_{\mathcal{B}})$ and a signature $\sigma_0$ on $D_0$, sends $(P, pk_{\mathcal{B}}, \sigma_0)$ and pays $ac_0$ to $\mathcal{V}$ through an arbitrary payment channel. $\mathcal{V}$ runs $(D_0, ac_0) \leftarrow \mathsf{POTGetDep}(crs, sk_{\mathcal{V}}, P, D_{max})$ to obtain the commitment $D_0$ and checks that $ac_0$ equals the amount of money paid. $\mathcal{V}$ runs $\mathsf{OFInitV}(pk_{\mathcal{B}}, \sigma_0, D_0)$

to check $\sigma_0$ and aborts if the output is reject. $\mathcal{A}$ runs $(ask, apk) \leftarrow \mathsf{OFInitA}(crs)$ and sends $apk$ to $\mathcal{V}$ and $\mathcal{B}$. $\mathcal{V}$ and $\mathcal{B}$ retrieve $apk$. $\mathcal{V}$ ($\mathcal{B}$) sends $(pk_{\mathcal{V}}, pk_{\mathcal{B}})$ $((pk'_{\mathcal{V}}, pk'_{\mathcal{B}}))$ to $\mathcal{A}$, $\mathcal{A}$ checks equality of the public keys and stores $(pk_{\mathcal{V}}, pk_{\mathcal{B}}, ask)$. (Alternatively, $\mathcal{A}$, $\mathcal{V}$ and $\mathcal{B}$ can use a PKI.) $\mathcal{V}$ stores state information $V_0 = (sk_{\mathcal{V}}, T, pk_{\mathcal{V}}, pk_{\mathcal{B}}, apk, D_0, \sigma_0)$ and outputs $(\text{deposit}, ac_0)$, and $\mathcal{B}$ stores state information $B_0 = (sk_{\mathcal{B}}, T, pk_{\mathcal{B}}, pk_{\mathcal{V}}, apk, D_0^{'})$ and outputs $(\text{init}, p_1, \ldots, p_N)$.

**Transfer phase.** On input $(\text{request}, i, \tau)$, $\mathcal{B}$ does nothing if $\mathcal{B}$ has previously received $(\text{request}, i, \ldots)$. Otherwise $\mathcal{B}$ runs $(Q_i, Q_i^{'}, D_i^{'}) \leftarrow \mathsf{POTReq}(crs, pk_{\mathcal{V}}, T, D_{i-1}^{'}, \tau)$ to compute a request message $Q_i$, trapdoor information $Q_i^{'}$ and opening $D_i^{'}$, and $\omega_i \leftarrow \mathsf{OFReq}(sk_{\mathcal{B}}, apk, i, Q_i)$ to obtain a verifiably encrypted signature $\omega_i$ on $Q_i$. $\mathcal{B}$ sends the request $(i, Q_i, \omega_i)$ and stores $(Q_i^{'}, D_i^{'})$. Upon receiving $(i, Q_i, \omega_i)$, if $i$ does not equal the number of the transfer $t$, $\mathcal{V}$ does nothing. Otherwise, on input $(\text{response}, b)$, if $b = 0$ $\mathcal{V}$ sends $(\perp)$ to $\mathcal{B}$. If $b = 1$ $\mathcal{V}$ runs $\mathsf{POTVerReq}(crs, pk_{\mathcal{V}}, D_{i-1}, Q_i)$ and $\mathsf{OFVerReq}(pk_{\mathcal{B}}, apk, \omega_i, i, Q_i)$. If any of the two algorithms outputs reject, $\mathcal{V}$ rejects the request. Otherwise $\mathcal{V}$ runs $(R) \leftarrow \mathsf{POTResp}(crs, pk_{\mathcal{V}}, sk_{\mathcal{V}}, Q_i)$, sends the response $(R)$ and keeps state $(i, Q_i, \omega_i)$. If the output of $\mathsf{POTVerResp}(crs, pk_{\mathcal{V}}, R)$ is reject, $\mathcal{B}$ outputs $(\text{response}, i, \perp)$ and proceeds with the complaint phase. Otherwise, $\mathcal{B}$ runs $(m_\tau) \leftarrow \mathsf{POTComplete}(crs, T, R, Q_i^{'})$ and outputs $(\text{response}, i, m_\tau)$. On input $(\text{complete}, c, i)$, if $c = 0$, $\mathcal{B}$ sends $(\perp)$, and, if $c = 1$, $\mathcal{B}$ runs $\sigma_i \leftarrow \mathsf{OFComplete}(sk_{\mathcal{B}}, i, Q_i)$ and sends $(\sigma_i)$. $\mathcal{B}$ stores state information $B_i = (sk_{\mathcal{B}}, T, pk_{\mathcal{B}}, pk_{\mathcal{V}}, apk, D_i^{'})$. $\mathcal{V}$ runs $\mathsf{OFVerComp}(pk_{\mathcal{B}}, \sigma_i, i, Q_i)$ and, if the output is reject, $\mathcal{V}$ outputs $(\text{complete}, i, 0)$ and proceeds with the complaint phase. Otherwise $\mathcal{V}$ stores state information $V_i = (sk_{\mathcal{V}}, T, pk_{\mathcal{V}}, pk_{\mathcal{B}}, apk, Q_i, \sigma_i)$, increments the number of the transfer $t = t + 1$ and outputs $(\text{complete}, i, 1)$.

**Vendor complaint.** On input the tuple $(\text{complain}_{\mathcal{V}}, i)$, if no valid request $(i, Q_i, \omega_i)$ was previously received, $\mathcal{V}$ does nothing. Otherwise $\mathcal{V}$ runs $\mathsf{POTResp}(crs, pk_{\mathcal{V}}, sk_{\mathcal{V}}, Q_i)$ and sends the request-response pair $((i, Q_i, \omega_i), (R))$ to $\mathcal{A}$, along with the signature of the previous transfer $(i-1, Q_{i-1}, \sigma_{i-1})$ $((0, D_0, \sigma_0)$ when $i = 1)$. $\mathcal{A}$ parses $Q_{i-1}$ to obtain $D_{i-1}$ and runs $\mathsf{POTVerReq}(crs, pk_{\mathcal{V}}, D_{i-1}, Q_i)$, $\mathsf{OFVerReq}(pk_{\mathcal{B}}, apk, \omega_i, i, Q_i)$, $\mathsf{OFVerComp}(pk_{\mathcal{B}}, \sigma_{i-1}, i-1, Q_{i-1})$ $(\mathsf{OFInitV}(pk_{\mathcal{B}}, \sigma_0, D_0)$ when $i = 1)$ and $\mathsf{POTVerResp}(crs, pk_{\mathcal{V}}, R)$ and, if any of them outputs reject, sets $\sigma_i = \perp$ and $R = \perp$ and outputs $(\text{complain}_{\mathcal{V}}, i, \text{inv})$. Otherwise $\mathcal{A}$ runs $\sigma_i \leftarrow \mathsf{OFAdj}(pk_{\mathcal{B}}, ask, apk, \omega_i, i, Q_i)$. $\mathcal{A}$ sends $\sigma_i$ to $\mathcal{V}$ and $(i, R)$ to $\mathcal{B}$ and outputs $(\text{complain}_{\mathcal{V}}, i, \text{solved})$. $\mathcal{V}$ runs $\mathsf{OFVerComp}(pk_{\mathcal{B}}, \sigma_i, i, Q_i)$ and outputs $(\text{compSolved}_{\mathcal{V}})$ if the output is accept. $\mathcal{B}$ runs $(m_\tau) \leftarrow \mathsf{POTComplete}(crs, T, R, Q_i^{'})$, where $Q_i^{'}$ corresponds to the request for transfer $i$, and outputs $(\text{compResp}_{\mathcal{V}}, i, m_\tau)$.

**Buyer complaint.** On input $(\text{complain}_{\mathcal{B}}, i, \tau)$, if a request $(i, Q_i, \omega_i)$ was previously computed $\mathcal{B}$ sends it to $\mathcal{A}$. Otherwise, $\mathcal{B}$ sends to $\mathcal{A}$ a new re-

quest computed by running POTReq and OFReq. $\mathcal{A}$ sends $(i)$ to $\mathcal{V}$, which returns $(j, Q_j, \sigma_j)$ ($(0, D_0, \sigma_0)$ if $i = 1$). Then $\mathcal{A}$ proceeds as follows:

- If $\sigma_j$ is invalid, $\mathcal{A}$ sends $(\text{guilty}_{\mathcal{V}})$ to $\mathcal{V}$ and $\mathcal{B}$ and outputs $(\text{complain}_{\mathcal{B}}, i, \text{guilty}_{\mathcal{V}})$.
- If $\sigma_j$ is valid but $j \geq i$, $\mathcal{A}$ sends $(\text{compInv}_{\mathcal{B}})$ to $\mathcal{B}$ and outputs $(\text{complain}_{\mathcal{B}}, i, \text{inv})$.
- If $\sigma_j$ is valid and $j = i - 1$, $\mathcal{A}$ parses $Q_{i-1}$ to obtain $D_{i-1}$, runs $\mathsf{POTVerReq}(crs, pk_{\mathcal{V}}, D_{i-1}, Q_i)$ and, if the output is reject, sends $(\text{compInv}_{\mathcal{B}})$ to $\mathcal{B}$ and outputs $(\text{complain}_{\mathcal{B}}, i, \text{inv})$. $\mathcal{V}$ (possibly) sends another request $(i, Q_i', \omega_i')$. $\mathcal{A}$ verifies it and, if it is correct, but $(i, Q_i', \omega_i') \neq (i, Q_i, \omega_i)$, $\mathcal{A}$ sends $(\text{compInv}_{\mathcal{B}})$ to $\mathcal{B}$ and outputs $(\text{complain}_{\mathcal{B}}, i, \text{inv})$. Otherwise $\mathcal{A}$ sends $(i, Q_i, \omega_i)$ to $\mathcal{V}$. $\mathcal{V}$ verifies the request, runs $(R) \leftarrow \mathsf{POTResp}(crs, T, sk_{\mathcal{V}}, Q_i)$ and sends $(R)$ to $\mathcal{A}$. $\mathcal{A}$ verifies the response via $\mathsf{POTVerResp}(crs, pk_{\mathcal{V}}, R)$ and, if it is not valid, $\mathcal{A}$ sends $(\text{guilty}_{\mathcal{V}})$ to $\mathcal{B}$ and $\mathcal{V}$ and outputs $(\text{complain}_{\mathcal{B}}, i, \text{guilty}_{\mathcal{V}})$. If it is valid, $\mathcal{A}$ sends $R$ to $\mathcal{B}$ and $\sigma_i \leftarrow \mathsf{OFAdj}(pk_{\mathcal{B}}, ask, apk, \omega_i, i, Q_i)$ to $\mathcal{V}$, and outputs $(\text{complain}_{\mathcal{B}}, i, \text{solved})$. $\mathcal{B}$ retrieves $m_\tau$ via POTComplete and outputs $(\text{compResp}_{\mathcal{B}}, i, m_\tau)$, while $\mathcal{V}$ outputs $\text{compSolved}_{\mathcal{B}}$.

Algorithms POT* correspond to those of the POT schemes (see Section 2.4 and the full version [36]). Algorithms OF* are defined as follows:

$\mathsf{OFInitB}(crs, D_0')$ chooses a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and runs $(sk, pk) \leftarrow \mathsf{Kg}(1^\kappa)$ of the VES scheme for the pairing group setup contained in $crs$. Then it parses $D_0'$ to obtain the commitment to the deposit $D_0$ and signs $\sigma_0 \leftarrow \mathsf{Sign}(sk, H(0, D_0))$. It outputs $sk_{\mathcal{B}} = sk$, $pk_{\mathcal{B}} = (H, pk)$ and $\sigma_0$.

$\mathsf{OFInitV}(pk_{\mathcal{B}}, \sigma_0, D_0)$ outputs the result of $\mathsf{Vf}(pk, \sigma_0, H(0, D_0))$.

$\mathsf{OFInitA}(crs)$ runs $\mathsf{AdjKg}(1^\kappa)$ and outputs a key pair $(ask, apk)$ for the pairing group setup contained in $crs$.

$\mathsf{OFReq}(sk_{\mathcal{B}}, apk, i, Q_i)$ outputs a verifiably encrypted signature $\omega_i \leftarrow \mathsf{Create}(sk_{\mathcal{B}}, apk, H(i, Q_i))$.

$\mathsf{OFVerReq}(pk_{\mathcal{B}}, apk, \omega_i, i, Q_i)$ outputs the result of $\mathsf{VesVf}(pk, apk, \omega_i, H(i, Q_i))$.

$\mathsf{OFComplete}(sk_{\mathcal{B}}, i, Q_i)$ outputs $\sigma_i \leftarrow \mathsf{Sign}(sk_{\mathcal{B}}, H(i, Q_i))$.

$\mathsf{OFVerComp}(pk_{\mathcal{B}}, \sigma_i, i, Q_i)$ outputs $\mathsf{Vf}(pk, \sigma_i, H(i, Q_i))$.

$\mathsf{OFAdj}(pk_{\mathcal{B}}, ask, apk, \omega_i, i, Q_i)$ outputs $\mathsf{Adj}(pk, ask, apk, \omega, H(i, Q_i))$.

**Theorem 2.** *This OFPOT scheme securely realizes* $\mathcal{F}_{\text{OFPOT}}$.

We prove this theorem in the full version [36]. We prove security under static corruptions and under the assumption that $\mathcal{A}$ is neutral. If the $\mathcal{A}$ colludes either with $\mathcal{V}$ or with $\mathcal{B}$, the privacy properties of the underlying POT scheme still hold, but fairness does not hold anymore. Buyer security holds under the assumption

that buyer's security in the POT scheme holds, under the unforgeability and opacity properties of the VES scheme, and under the assumption that $H$ is collision-resistant. Vendor security holds under the assumption that vendor's security in the POT scheme holds and under the extractability of the VES scheme. We note that the proof that buyer's privacy also holds with respect to $\mathcal{A}$ follows the proof given with respect to $\mathcal{V}$, because the view of $\mathcal{A}$ and $\mathcal{V}$ with respect to $\mathcal{B}$ is equivalent.

As for efficiency, we note that the overhead in terms of computation and communication cost introduced by the VES scheme are small compared to the cost of the POT scheme. Therefore, a secure POT can efficiently be extended to provide optimistic fair exchange.

## 4 Conclusion

Our contribution is twofold. First, we have designed a full-simulation secure POT scheme that is more efficient than previous work. Second, we have proposed a generic construction that provides any secure POT scheme with optimistic fair exchange.

We leave open the definition of a more general ideal functionality for fair privacy-preserving e-commerce protocols. Another interesting open problem is the design of a fair POT scheme that is abuse-free in the sense of [40]. Further research also needs to be conducted to show how to integrate e-commerce protocols based on POT with digital rights management systems, and to analyze the compliance of such e-commerce protocols with e-commerce legislation.

## Acknowledgements

This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State. Alfredo Rial is funded by the Research Foundation - Flanders (FWO).

## References

1. Ackerman, M.S., Cranor, L.F., Reagle, J.: Privacy in e-commerce: examining user scenarios and privacy preferences. In: ACM Conference on Electronic Commerce, pp. 1–8 (1999)
2. Tsai, J., Egelman, S., Cranor, L., Acquisti, R.: The effect of online privacy information on purchasing behavior: An experimental study (June 2007) (working paper)
3. Enforcing privacy promises: Section 5 of the ftc act. Federal Trade Commission Act, http://www.ftc.gov/privacy/privacyinitiatives/promises.html
4. Kremer, S.: Formal analysis of optimistic fair exchange protocols (2004)
5. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)

6. Tobias, C.: Practical oblivious transfer protocols. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 415–426. Springer, Heidelberg (2003)
7. Rial, A., Kohlweiss, M., Preneel, B.: Universally composable adaptive priced oblivious transfer. In: [42], pp. 231–247
8. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM 28(6), 637–647 (1985)
9. Goldreich, O.: A simple protocol for signing contracts. In: CRYPTO, pp. 133–136 (1983)
10. Bao, F., Deng, R.H., Mao, W.: Efficient and practical fair exchange protocols with off-line ttp. In: IEEE Symposium on Security and Privacy, pp. 77–85. IEEE Computer Society, Los Alamitos (1998)
11. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures (extended abstract). In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 591–606. Springer, Heidelberg (1998)
12. Avoine, G., Vaudenay, S.: Optimistic fair exchange based on publicly verifiable secret sharing. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 74–85. Springer, Heidelberg (2004)
13. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
14. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)
15. Rückert, M., Schröder, D.: Security of verifiably encrypted signatures and a construction without random oracles. In: [42], pp. 17–34
16. Ray, I., Ray, I.: An anomymous fair exchange e-commerce protocol. In: IPDPS, p. 172. IEEE Computer Society, Los Alamitos (2001)
17. Naor, M., Pinkas, B.: Computationally secure oblivious transfer. J. Cryptology 18(1), 1–35 (2005)
18. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
19. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
20. Camenisch, J., Dubovitskaya, M., Neven, G.: Oblivious transfer with access control. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM Conference on Computer and Communications Security, pp. 131–140. ACM, New York (2009)
21. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
22. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
23. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
24. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
25. Schnorr, C.P.: Efficient signature generation for smart cards. Journal of Cryptology 4(3), 239–252 (1991)

26. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
27. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient protocols for set membership and range proofs. In: [41], pp. 234–252
28. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
29. Damgård, I.: Concurrent zero-knowledge is easy in practice. Available online at Theory of Cryptography Library (June 1999)
30. Damgård, I.: On $\sigma$-protocols (2002), http://www.daimi.au.dk/~ivan/Sigma.ps
31. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
32. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich (March 1997)
33. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 136–145 (2001)
34. Green, M., Hohenberger, S.: Universally composable adaptive oblivious transfer. In: [41], pp. 179–197
35. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: First ACM Conference on Computer and Communication Security, Association for Computing Machinery, pp. 62–73 (1993)
36. Rial, A., Preneel, B.: Optimistic fair priced oblivious transfer (2009), http://www.cosic.esat.kuleuven.be/publications/article-1428.pdf
37. Okada, Y., Manabe, Y., Okamoto, T.: An optimistic fair exchange protocol and its security in the universal composability framework. IJACT 1(1), 70–77 (2008)
38. Canetti, R.: Obtaining universally compoable security: Towards the bare bones of trust. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 88–112. Springer, Heidelberg (2007)
39. Herranz, J.: Restricted adaptive oblivious transfer. Cryptology ePrint Archive, Report 2008/182 (2008), http://eprint.iacr.org/
40. Garay, J., Jakobsson, M., MacKenzie, P.: Abuse-free optimistic contract signing. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 449–466. Springer, Heidelberg (1999)
41. Pieprzyk, J. (ed.): ASIACRYPT 2008. LNCS, vol. 5350. Springer, Heidelberg (2008)
42. Shacham, H., Waters, B. (eds.): Pairing 2009. LNCS, vol. 5671. Springer, Heidelberg (2009)

# Information-Theoretically Secure Key-Insulated Multireceiver Authentication Codes

Takenobu Seito, Tadashi Aikawa, Junji Shikata, and Tsutomu Matsumoto

Graduate School of Environment and Information Sciences,
Yokohama National University, Japan
seito@mlab.jks.ynu.ac.jp, {shikata,tsutomu}@ynu.ac.jp

**Abstract.** Exposing a secret-key is one of the most disastrous threats in cryptographic protocols. The key-insulated security is proposed with the aim of realizing the protection against such key-exposure problems. In this paper, we study key-insulated authentication schemes with information-theoretic security. More specifically, we focus on one of information-theoretically secure authentication, called multireceiver authentication codes, and we newly define a model and security notions of information-theoretically secure key-insulated multireceiver authentication codes (KI-MRA for short) based on the ideas of both computationally secure key-insulated signature schemes and multi-receiver authentication-codes with information-theoretic setting. In addition, we show lower bounds of sizes of entities' secret-keys. We also provide two kinds of constructions of KI-MRA: direct and generic constructions which are provably secure in our security definitions. It is shown that the direct construction meets the lower bounds of key-sizes with equality. Therefore, it turns out that our lower bounds are tight, and that the direct construction is optimal.

**Keywords:** information-theoretic security, key-insulated security, multireceiver authentication-code, unconditional security.

## 1 Introduction

### 1.1 Background

The security of most of present cryptographic techniques is based on the assumption of difficulty of computationally hard problems such as the integer factoring or discrete logarithm problems. However, taking into account recent rapid development of algorithms and computer technologies, such a scheme based on the assumption of difficulty of computationally hard problems might not maintain sufficient long-term security. In fact, it is known that quantum computers can solve the factoring and discrete logarithm problems in polynomial time [19]. From these aspects, it is necessary and interesting to consider cryptographic techniques whose security does not depend on any computationally hard problems.

One of the most serious threats in cryptographic protocols is exposure of secret-keys. For example, digital signature schemes require use of the secret-keys

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 148–165, 2010.

to sign a message, and if a secret-key is compromised, this implies an attacker gaining ability to generate a signature of all messages. Hence, it leads to a total break of the systems. However, we would want to minimize the risk of damage even if such an unfortunate situation as exposure of secret-keys does occur. For this purpose, there have been several approaches proposed to minimize the risk of key-exposure: *Key-splitting* is a method which divides portions of a single secret-key among multiple entities so that no entity will have the ability to reconstruct a whole secret-key [3] [5] [20]; Another approach is to consider key-evolution to realize *forward-secure schemes* [1] [2] [4] where lifetime of systems is divided into discrete periods and a secret-key is updated at each new period. It ensures that the security of past periods remains uncompromised even if the current secret-key is exposed. The security notion that is one solution to the key-exposure problems is proposed by Dodis et al., and it is called *Key-Insulated Security* [7] [8], which is based on the ideas of combining key-splitting and key-evolution. This method extracts benefits from both approaches: having the information manageable in case of loss of key and leaving authentication as a stand-alone user operation at the same time.

In the model of key-insulated signature schemes, a signer has two kinds of devises: a trusted device (e.g. a smart card, USB flash memory) in which a master-key is stored; and an insecure device in which a signer's secret-key is stored. The actual secret-key updating is performed in the insecure device. When the signer wants to get a signing-key at a period $j$, the following process is performed in the beginning of the period $j$: first, the trusted device generates key-updating information by using the master-key and sends it to the signer; and then, the signer updates his secret-key by using the key-updating information, and he deletes the previous secret-key. In key-insulated signature schemes, if the trusted device is not compromised, then signer's secret-keys of at most $\gamma$ periods can be exposed without losing security, where $\gamma$ is a predefined number. In addition, even if the trusted device is exposed, the system will not be violated if no signer's secret-key is exposed. This property is called *strong key-insulation* [7] [8]. Hence the system having strong key-insulation guarantees the security against two different types of attacks: (i) the attack to steal a secret-key stored in an insecure device via a network; and (ii) the attack to steal a master-key stored in a (physically-protected) secure device directly. We consider that the property of strong key-insulation is important and useful when using a system in the real world.

As mentioned earlier, the first constructions of key-insulated schemes are proposed in [7] [8]. Since then, many papers on this subject have been reported to give theoretical and practical key-insulated schemes. Also, Itkis et al. proposed an extended version of key-insulated signature schemes, *Intrusion-resilient signatures* [13]. The security of most of key-insulated schemes described so far is based on the assumption of difficulty of computationally hard problem such as the integer factoring or discrete logarithm problems. In this paper, we study key-insulated schemes in the setting of the *information-theoretic security* (a.k.a. *unconditional security*) rather than the computational security.

### 1.2 Our Contribution

Confidentiality (secrecy) and authenticity (integrity) are currently fundamental cryptographic functions, and encryption and authentication/signature schemes are usually used for providing confidentiality and authenticity, respectively. As key-insulated schemes with information-theoretic security, Hanaoka et al. first proposed the information-theoretically secure key-insulated encryption schemes [11]. However, information-theoretically secure authentication/signature schemes with key-insulated security have not proposed so far. Therefore, we study authentication schemes which have both information-theoretic and key-insulated security, and the purpose of this paper is to consider a simple model of information-theoretically secure key-insulated authentication schemes.

We note that information-theoretically secure signature schemes were proposed by Shikata et al. in [12] [18]. Thus, one may consider the information-theoretically secure key-insulated signature schemes. However, the model of those schemes in [12] [18] is not so simple, since it requires complicated security notions. On the other hand, we note that the model of *multireceiver authentication codes* (*MRA-codes* for short) was proposed by Desmedt et al. in [6] and later generalized by Safavi-Naini et al. [17] and Johansson [14]. The MRA-code is one of the information-theoretically secure authentication schemes which allows a single honest sender to transmit an authenticated message to a group of receivers via a broadcast channel. And each receiver can individually verify the authenticated message. Note that the model of MRA-codes is simpler than that of information-theoretically secure signature schemes.

In this paper, we study the model of *key-insulated MRA-codes* (*KI-MRA* for short), which are information-theoretically secure authentication schemes with key-insulated security, rather than information-theoretically secure key-insulated signature schemes, since the model of MRA-codes is simpler than that of information-theoretically secure signature schemes. More specifically, we newly introduce the model and security definitions, and show lower bounds and constructions of KI-MRA. We begin by formalizing the model and security notions of KI-MRA based on those of MRA-codes and computationally secure key-insulated signature schemes. In particular, the notion of strong key-insulation is formalized along with our model. In addition, we show lower bounds of sizes of entities' secret-keys. We also provide two kinds of constructions of KI-MRA, direct and generic constructions which are provably secure in our security definitions: we propose the direct construction by using polynomials over finite fields; and we provide the generic construction of KI-MRA starting from cover-free-families(CFF) [9] and MRA-codes. Furthermore, it is shown that the direct construction meets the lower bounds of key-sizes with equality. Therefore, it turns out that our lower bounds are tight, and that the direct construction is optimal.

The rest of this paper is organized as follows. In Section 2, we introduce the model of KI-MRA based on the ideas according to [7], [8], [17], and formalize the security notions of KI-MRA. We also show lower bounds of memory-sizes of

secret-keys of entities. In Section 3, we propose two kinds of constructions: direct and generic constructions which are provably secure in our security notions. Finally, in Section 4, we give concluding remarks of the paper.

## 2 The Model and Security Definitions

In this section, we introduce the model and security notions of KI-MRA, based on those of key-insulated signatures with computational security and those of MRA-codes with information-theoretic security.

### 2.1 The Model

We show the model of KI-MRA, which is the model of MRA-codes with key-insulated security. As in the models of many schemes with information-theoretic security (e.g. [16] [17] [18]), we assume that there is a trusted authority whose role is to generate and to distribute secret-keys of entities. We call this model the *trusted initializer model* as in [16]. In KI-MRA, there are $n+3$ entities, a sender $S$, a secure device $H$, $n$ receivers $R_1, R_2, \ldots, R_n$ and a trusted initializer TI. We assume that the sender is honest in the model. In our model, the notion of a *secure device* implies that it is usually isolated from a network (e.g. the Internet or LAN) and that the attacker can neither wiretap nor substitute information stored in the device via the network. For example, a smart card or USB flash memory seems to be a candidate of such devices. In addition, in KI-MRA, we assume that lifetime of the system is divided into $N$ periods. For simplicity, we consider a *one-time model* of KI-MRA, in which the sender is allowed to generate and broadcast an authenticated message at most only once per period[1].

Informally, KI-MRA is executed as follows. In the initial phase, TI generates secret-keys on behalf of $S$, $H$ and $R_i$ $(1 \leq I \leq n)$. After distributing these secret-keys via a secure channel, TI deletes them in his memory. For updating the sender's secret-key for the period $j$, $S$ receives key-updating information from $H$ by connecting with $H$, and $S$ computes a secret-key at the period $j$ by using the secret-key of the previous period and the key-updating information. $S$ then deletes the secret-key of the previous period and the key-updating information. On the other hand, each receiver's secret-key used to check the validity of an authenticated messages will not be updated at each period as key-insulated signature schemes with computational security. If $S$ generates and broadcasts an authenticated message $\alpha$ at a period $j$, each receiver, $R_i$, can check the validity of $\alpha$ by using his secret-key. Formally, we give the definition as follows.

**Definition 1 (KI-MRA).** A *key-insulated multireceiver authentication codes* (*KI-MRA* for short) $\Pi$ involves $n+3$ entities, TI, $S$, $H$ and $R_1, R_2, \ldots, R_n$,

[1] We can consider a more general setting: the number up to which the sender is allowed to generate authenticated messages is more than one per period. However, the one-time model of KI-MRA is simpler than this model in terms of the number of generating authenticated messages. Therefore, we focus on the model since our purpose is to study a simple model as mentioned in Section 1.2.

and consists of a five-tuple of algorithms (*KGen*, *KUpd*$^*$, *KUpd*, *KAuth*, *KVer*) with eight spaces, $\mathcal{M}$, $\mathcal{A}$, $\mathcal{I}$, $\mathcal{T}$, $\tilde{\mathcal{T}}$, $\mathcal{MK}$, $\mathcal{E}_S$ and $\mathcal{E}_R$, where all of the above algorithms except *KGen* are deterministic and all of the above spaces are finite. In addition, $\Pi$ is executed with four phases as follows.

– **Notation:**
  - TI is a trusted initializer, $S$ is a sender, $H$ is a secure device and $R_i$ $(i = 1, 2, \ldots, n)$ is a receiver. Let $R := \{R_1, R_2, \ldots, R_n\}$ be a set of receivers.
  - $\mathcal{M}$ is a set of possible messages with a probability distribution $\mathrm{Pr}_M$.
  - $\mathcal{A}$ is a set of possible authenticated messages.
  - $\mathcal{I}$ is a set of possible key-updating information.
  - $\mathcal{T} := \{1, 2, \ldots, N\}$ is a set of time periods. Let $\tilde{\mathcal{T}} := \mathcal{T} \cup \{0\}$.
  - $\mathcal{MK}$ is a set of possible master-keys for the device $H$ with a probability distribution $\mathrm{Pr}_{MK}$.
  - $\mathcal{E}_S^{(j)}$ is a set of possible secret-keys at the period $j$ for the sender with a probability distribution $\mathrm{Pr}_{E_S^{(j)}}$. Let $\mathcal{E}_S := \mathcal{E}_S^{(0)} \cup \mathcal{E}_S^{(1)} \cup \ldots \cup \mathcal{E}_S^{(N)}$.
  - $\mathcal{E}_i$ is a set of secret-keys for a receiver $R_i$ with a probability distribution $\mathrm{Pr}_{E_i}$. Let $\mathcal{E}_R := \mathcal{E}_1 \cup \mathcal{E}_2 \cup \ldots \cup \mathcal{E}_n$.
  - *KGen* is a key generation algorithm which on input a security parameter $1^k$, outputs a master-key, a sender's secret-key and each receiver's secret-key.
  - *KUpd*$^*$: $\mathcal{MK} \times \tilde{\mathcal{T}} \times \mathcal{T} \to \mathcal{I}$ is a key-updating algorithm for the device $H$.
  - *KUpd*: $\mathcal{E}_S \times \mathcal{I} \to \mathcal{E}_S$ is a key-updating algorithm for the sender $S$.
  - *KAuth*: $\mathcal{E}_S \times \mathcal{M} \to \mathcal{A}$ is an authentication algorithm for producing authenticated messages.
  - *KVer*: $\mathcal{E}_R \times \mathcal{A} \times \mathcal{T} \to \{true, false\}$ is a verification algorithm.

1. **Key Generation and Distribution by TI.** In the initial phase, TI generates the following keys by using *KGen*: a master-key $mk \in \mathcal{MK}$; an initial secret-key $e_S^{(0)} \in \mathcal{E}_S^{(0)}$ (i.e., a secret-key at the period 0) for the sender; and the receiver $R_i$'s secret-key $e_i \in \mathcal{E}_i$ $(i = 1, 2, \ldots, n)$. These keys are distributed to corresponding entities via secure channels. After distributing these keys, TI deletes these keys from his memory. And, $H$, $S$, and $R_i$ keep their keys secret, respectively.
2. **Updating Sender's Secret-keys.** For updating a sender's secret-key for a period $j$ from a period $h$, $H$ generates key-updating information $mk^{(h,j)} = KUpd^*(mk, h, j) \in \mathcal{I}$ by using the master-key $mk$, the information on periods $h \in \tilde{\mathcal{T}}, j \in \mathcal{T}$ and *KUpd*$^*$, and then sends it to $S$ via a secure channel. After that, $S$ computes a secret-key $e_S^{(j)} = KUpd(e_S^{(h)}, mk^{(h,j)}) \in \mathcal{E}_S^{(j)}$ at the period $j$. Then, $S$ deletes $e_S^{(h)}$ and $mk^{(h,j)}$ from his memory.
3. **Authentication.** During a period $j$, for a message $m \in \mathcal{M}$, $S$ generates an authenticated message $\alpha = KAuth(e_S^{(j)}, m) \in \mathcal{A}$ by using a secret-key $e_S^{(j)}$ at the period $j$. Then, $S$ sends the authenticated message with information on $j$, namely $(\alpha, j)$, to all receivers via a broadcast channel.

4. **Verification.** After receiving $(\alpha, j)$ from $S$, each receiver $R_i$ can verify the validity of it by using secret-key $e_i$ and $KVer$. If $KVer(e_i, \alpha, j) = \text{true}$, then $R_i$ accepts $(\alpha, j)$ as valid, and rejects it otherwise.

In the model of KI-MRA, we require the following equation holds: for all possible $j \in \mathcal{T}$, $m \in \mathcal{M}$, $e_S^{(j)} \in \mathcal{E}_S^{(j)}$, and $e_i \in \mathcal{E}_i$, we have

$$KVer(e_i, KAuth(e_S^{(j)}, m), j) = true$$

The above requirement implies that any legal authenticated message can be accepted without error if entities correctly follows the specification of KI-MRA.

In addition, we define several notation as follows. Let $\omega$ be the number of possible colluders, and let $\gamma$ be the number of possible periods at which sender's secret-keys may be exposed. And, for any set $\mathcal{Z}$ and any nonnegative integer $z$, let $\mathcal{P}(\mathcal{Z}, z) := \{Z \subset \mathcal{Z} || Z| \leq z\}$ be the family of all subsets of $\mathcal{Z}$ whose cardinality is less than or equal to $z$. And also, let $W := \{R_{i_1}, R_{i_2}, \ldots, R_{i_\omega}\} \in \mathcal{P}(R, \omega)$ be a set of possible colluders and $\mathcal{E}_W := \mathcal{E}_{i_1} \times \mathcal{E}_{i_2} \ldots \times \mathcal{E}_{i_\omega}$ be a set of possible secret-keys held by $W$. Furthermore, let $\Gamma := \{j_1, j_2, \ldots, j_\gamma\} \subset \mathcal{P}(\mathcal{T}, \gamma)$ be a set of periods at which sender's secret-keys are exposed, and $\mathcal{E}_\Gamma := \mathcal{E}_S^{(j_1)} \times \mathcal{E}_S^{(j_2)} \times \ldots \times \mathcal{E}_S^{(j_\gamma)}$ be a set of sender's secret-keys exposed. With these notation, we will discuss KI-MRA in the following sections.

### 2.2 Security Notions and Their Formalization

We now provide security notions and their formalization of KI-MRA in the one-time model based on key-insulated signature schemes with computational security [7] [8] and MRA-codes [17]. In MRA-codes, there are two kinds of attacks: *impersonation attack* and *substitution attacks* (see Appendix A for the detail of those attacks). Therefore, we consider similar attacks in KI-MRA. In the model of KI-MRA, we assume that the adversary can corrupt at most $\omega$ dishonest receivers among $R$, and we do not think about any attack by the sender, since he is assumed to be honest in the model as in that of MRA-codes. In addition, we need to consider the following two types of exposure as in [7] [8]:

- Type A: *Sender's secret-key exposure*, which models compromise of sender's secret-keys from the insecure device.
- Type B: *Master-key exposure*, which models compromise (robbery) of the secure device by physical means.

Therefore, by combining two kinds of attacks (i.e., impersonation and substitution attacks) in MRA-codes and two types of key-exposure (i.e., Types A and B) above, we consider four kinds of security notions of KI-MRA as follows.

**Definition 2 (Security of KI-MRA).** Let $\Pi$ be a KI-MRA. The scheme $\Pi$ is said to be an $(n, \omega; N, \gamma; \epsilon_A, \epsilon_B)$-*one-time secure* if $P_{\Pi,A} \leq \epsilon_A$ and $P_{\Pi,B} \leq \epsilon_B$, where $P_{\Pi,A}$ and $P_{\Pi,B}$ are defined as follows.

A-1) **Impersonation attack in the exposure type A.** The adversary who corrupts at most $\omega$ receivers tries to generate a fraudulent authenticated message at a period $t$, $(\alpha, t)$, that has not been legally generated by the sender $S$ but will be accepted by a receiver $R_i$. Here, we assume that $R_i$ is not included in the colluders, and the adversary can gather information by pooling secret-keys of corrupted receivers, at most $\gamma$ sender's secret-keys exposed (but, the adversary cannot obtain the secret-key at the target period $t$). The success probability of this attack denoted by $P_{\Pi,I_A}$ is defined as follows: For any set of colluders $W \in \mathcal{P}(R, \omega)$, any set of key-exposure periods $\Gamma \in \mathcal{P}(\mathcal{T}, \gamma)$, any targeted honest receiver $R_i \notin W$ and target period $t \notin \Gamma$, we define $P_{\Pi,I_A}(R_i, W, \Gamma, t)$ as

$$P_{\Pi,I_A}(R_i, W, \Gamma, t) := \max_{e_W} \max_{e_\Gamma} \max_{(\alpha,t)} \Pr(KVer(e_i, \alpha, t) = true | e_W, e_\Gamma),$$

where the probability is taken over random choice of KGen, and the maximum is taken over: all possible sets of the colluders' secret-keys $e_W \in \mathcal{E}_W$; all possible sets of sender's secret-keys $e_\Gamma \in \mathcal{E}_\Gamma$ exposed such that $e_S^{(t)} \notin e_\Gamma$; and all possible authenticated messages $(\alpha, t) \in \mathcal{A} \times \mathcal{T}$. Then, the probability $P_{\Pi,I_A}$ is defined as $P_{\Pi,I_A} := \max_{R_i,W,\Gamma,t} P_{\Pi,I_A}(R_i, W, \Gamma, t)$.

A-2) **Substitution attack in the exposure type A.** The adversary who corrupts at most $\omega$ receivers tries to generate a fraudulent authenticated message at a period $t$, $(\alpha, t)$, that has not been legally generated by the sender $S$ but will be accepted by a receiver $R_i$, after observing a valid authenticated message at the same period, $(\alpha', t)$. Here, we assume that $R_i$ is not included in the colluders, and the adversary can gather information by pooling secret-keys of corrupted receivers, at most $\gamma$ sender's secret-keys exposed (but, the adversary cannot obtain the secret-key at the target period $t$). The success probability of this attack denoted by $P_{\Pi,S_A}$ is defined as follows: For any set of colluders $W \in \mathcal{P}(R, \omega)$, any set of key-exposure periods $\Gamma \in \mathcal{P}(\mathcal{T}, \gamma)$, any targeted honest receiver $R_i \notin W$ and target period $t \notin \Gamma$, we define $P_{\Pi,S_A}(R_i, W, \Gamma, t)$ as

$$P_{\Pi,S_A}(R_i, W, \Gamma, t) := \max_{e_W} \max_{e_\Gamma} \max_{(\alpha',t)} \max_{(\alpha,t) \neq (\alpha',t)} \Pr(KVer(e_i, \alpha, t) = true | e_W, e_\Gamma, (\alpha', t)),$$

where the probability is taken over random choice of KGen, and the maximum is taken over: all possible sets of the colluders' secret-keys $e_W \in \mathcal{E}_W$; all possible sets of sender's secret-keys exposed $e_\Gamma \in \mathcal{E}_\Gamma$ such that $e_S^{(t)} \notin e_\Gamma$; and all possible authenticated messages $(\alpha', t), (\alpha, t) \in \mathcal{A} \times \mathcal{T}$ such that $\alpha \neq \alpha'$. Then, the probability $P_{\Pi,S_A}$ is defined as $P_{\Pi,S_A} := \max_{R_i,W,\Gamma,t} P_{\Pi,S_A}(R_i, W, \Gamma, t)$. And, we define $P_{\Pi,A} := \max(P_{\Pi,I_A}, P_{\Pi,S_A})$.

B-1) **Impersonation attack in the exposure type B.** The adversary who corrupts at most $\omega$ receivers tries to generate a fraudulent authenticated message $(\alpha, t)$ that has not been legally generated by the sender $S$ but

will be accepted by a receiver $R_i$. Here, we assume that $R_i$ is not included in the colluders, and the adversary can gather information by pooling secret-keys of corrupted receivers and a master-key exposed. The success probability of this attack denoted by $P_{\Pi,I_B}$ is defined as follows: For any set of colluders $W \in \mathcal{P}(R,\omega)$, any targeted honest receiver $R_i \notin W$ and target period $t$, we define $P_{\Pi,I_B}(R_i,W,t)$ as

$$P_{\Pi,I_B}(R_i,W,t) := \max_{e_W} \max_{mk} \max_{(\alpha,t)} \Pr(KVer(e_i,\alpha,t) = true | e_W, mk),$$

where the probability is taken over random choice of KGen, and the maximum is taken over: all possible sets of the colluders' secret-keys $e_W \in \mathcal{E}_W$; all possible master-keys exposed $mk \in \mathcal{MK}$; and all possible authenticated messages $(\alpha,t) \in \mathcal{A} \times \mathcal{T}$. Then, the probability $P_{\Pi,I_B}$ is defined as $P_{\Pi,I_B} := \max_{R_i,W,t} P_{\Pi,I_B}(R_i,W,t)$.

B-2) **Substitution attack in the exposure type B.** The adversary who corrupts at most $\omega$ receivers tries to generate a fraudulent authenticated message at a period $t$, $(\alpha,t)$, that has not been legally generated by the sender $S$ but will be accepted by a receiver $R_i$, after observing a valid authenticated message at the same period, $(\alpha',t)$. Here, we assume that $R_i$ is not included in the colluders, and the adversary can gather information by pooling secret-keys of corrupted receivers and a master-key exposed. The success probability of this attack denoted by $P_{\Pi,S_B}$ is defined as follows: For any set of colluders $W \in \mathcal{P}(R,\omega)$, any targeted honest receiver $R_i \notin W$ and target period $t$, we define $P_{\Pi,S_B}(R_i,W,t)$ as

$$P_{\Pi,S_B}(R_i,W,t) := \max_{e_W} \max_{mk} \max_{(\alpha',t)} \max_{(\alpha,t)\neq(\alpha',t)} \Pr(KVer(e_i,\alpha,t) = true | e_W, mk, (\alpha',t)),$$

where the probability is taken over random choice of KGen, and the maximum is taken over: all possible sets of the colluders' secret-keys $e_W \in \mathcal{E}_W$; all possible master-keys exposed $mk \in \mathcal{MK}$; and all possible authenticated messages $(\alpha',t),(\alpha,t) \in \mathcal{A} \times \mathcal{T}$ such that $\alpha \neq \alpha'$. Then, the probability $P_{\Pi,S_B}$ is defined as $P_{\Pi,S_B} := \max_{R_i,W,t} P_{\Pi,S_B}(R_i,W,t)$. And, we define $P_{\Pi,B} := \max(P_{\Pi,I_B}, P_{\Pi,S_B})$.

Instead of Definition 2, one can consider a more general attacking model as follows: in addition to information in Definition 2, the adversary may observe authenticated messages in all periods which are generated by the honest sender, since he can broadcast at most one authenticated message per period. However, this model is not so simple. Furthermore, even if we consider the general attacking model, we will obtain the very similar results (i.e., lower bounds and constructions) shown in Sections 2.3 and 3. Therefore, we consider the attacking model in Definition 2, since the purpose of this paper is to consider a simple and essential model of multireceiver authentication systems with key-insulated security.

### 2.3 Lower Bounds

In this section, we derive lower bounds of success probabilities of attacks and memory-sizes required for an $(n, \omega; N, \gamma; \epsilon_A, \epsilon_B)$-one-time secure KI-MRA. Let $\mathcal{A}^{(t)} := \{\alpha \in \mathcal{A} | KAuth(e_S^{(t)}, m) = \alpha \text{ for some } e_S^{(t)} \in \mathcal{E}_S^{(t)}, m \in \mathcal{M}\}$ be a set of possible authenticated messages which can be generated at a period $t$ by the sender. Also, let $\mathcal{I}^{(h,j)} \subset \mathcal{I}$ be a finite set of possible key-updating information which is used for key-updating process from a period $h$ to a period $j$. And, let $I^{(h,j)}$, $E_W$ and $E_\Gamma$ be random variables which take values on $\mathcal{I}^{(h,j)}$, $\mathcal{E}_W$ and $\mathcal{E}_\Gamma$, respectively. And also, let $(A^{(t)}, \tilde{A}^{(t)})$ be a joint random variable which takes values on the set $\mathcal{A}^{(t)} \times \mathcal{A}^{(t)}$ such that $A^{(t)} \neq \tilde{A}^{(t)}$.

We assume that there exist the following mappings in the model of KI-MRA, $\pi^{(j)} : \mathcal{E}_S^{(j)} \to \mathcal{E}_1^{(j)} \times \ldots \times \mathcal{E}_n^{(j)}$ and $f_i : \mathcal{E}_i \to \mathcal{E}_i^{(1)} \times \ldots \times \mathcal{E}_i^{(N)}$, where $\mathcal{E}_i^{(j)}$ is a set of possible $R_i$'s keys which are actually used at the period $j$[2]. Note that the assumption is natural and not so strange, since we will actually see these mappings in our constructions in Section 3. In the following, let $E_i^{(j)}$ be a random variable which takes values on $\mathcal{E}_i^{(j)}$.

Then, we can derive lower bounds of success probabilities of attacks as follows. The proof of sketch is given in Appendix C.

**Theorem 1.** *For any $i \in \{1, 2, \ldots, n\}$, any colluding group $W$ with $R_i \notin W$, any $t \in \mathcal{T}$, and any set of key-exposed time periods $\Gamma$ with $t \notin \Gamma$, we have the following inequalities:*

1. $P_{\Pi,I_A}(R_i, W, \Gamma, t) \geq 2^{-I(A^{(t)}; E_i^{(t)} | E_W, E_\Gamma)}$,
2. $P_{\Pi,S_A}(R_i, W, \Gamma, t) \geq 2^{-I(\tilde{A}^{(t)}; E_i^{(t)} | E_W, E_\Gamma, A^{(t)})}$,
3. $P_{\Pi,I_B}(R_i, W, t) \geq 2^{-I(A^{(t)}; E_i^{(t)} | E_W, MK)}$,
4. $P_{\Pi,S_B}(R_i, W, t) \geq 2^{-I(\tilde{A}^{(t)}; E_i^{(t)} | E_W, MK, A^{(t)})}$,

*where $I(X; Y|Z)$ means the conditional mutual information of random variables $X$ and $Y$ given $Z$.*

We next show lower bounds of memory-sizes of entities in KI-MRA. From Theorem 1, we obtain the following lower bounds of memory-sizes.

**Theorem 2.** *Let $\Pi$ be an $(n, \omega; N, \gamma; \epsilon, \epsilon)$-one-time secure KI-MRA. Let $q := \epsilon^{-1}$. Then, for any $i \in \{1, 2, \ldots, n\}$, $j \in \mathcal{T}$ and $h \in \tilde{\mathcal{T}}$, we have*: (i) $|\mathcal{E}_S^{(j)}| \geq q^{2(\omega+1)}$; (ii) $|\mathcal{E}_i| \geq q^{2(\gamma+1)}$; (iii) $|\mathcal{MK}| \geq q^{2\gamma(\omega+1)}$; (iv) $|\mathcal{I}^{(h,j)}| \geq q^{2(\omega+1)}$; and (v) $|\mathcal{A}^{(j)}| \geq 2^{H(M)} q^{\omega+1}$. *In particular, if $\Pr_M$ is the uniform distribution, $|\mathcal{A}^{(j)}| \geq q^{\omega+1} |\mathcal{M}|$.*

*Proof. The proof is given in Appendix B.*

As we will see in the next section, the above lower bounds are tight. Therefore, we define optimality of constructions of KI-MRA as follows.

[2] Note that, $e_i^{(j)}$, a $R_i$'s key actually used for a period $j$, may be a part of an entire key $e_i$ or may be equal to the entire key itself.

**Definition 3.** A construction of $(n, \omega; N, \gamma; \epsilon, \epsilon)$-one-time secure KI-MRA is said to be *optimal* if it meets all the inequalities (i)-(v) in Theorem 2 with equalities.

## 3 Constructions

In this section, we propose two kinds of constructions of KI-MRA, direct and generic constructions.

### 3.1 Direct Construction

We provide a direct construction which is one-time secure KI-MRA in our model by using polynomials over finite fields. In addition, it is shown that the direct construction meets the lower bounds of key-sizes with equalities. Therefore, it turns out that the direct construction is optimal. The construction is given as follows.

1. **Key Generation Algorithm *KGen*.** For a security parameter $1^k$, the algorithm *KGen* outputs matching secret-keys $e_S^{(0)}, mk, e_i (1 \leq i \leq n)$ for $S$, $H$, $R_i (1 \leq i \leq n)$ as follows. *KGen* picks a $k$-bit prime power $q$, where $q > \max(n, N)$, and constructs the finite field $F_q$ with $q$ elements. We assume that the identity of each receiver $R_i$ is also denoted by $R_i$ and that $R_i \subset F_q \setminus \{0\}$. Also, we assume $M \subset F_q \setminus \{0\}$, $\mathcal{T} = \{1, 2, \ldots, N\} \subset F_q \setminus \{0\}$ and $\tilde{\mathcal{T}} := \mathcal{T} \cup \{0\} \subset F_q$ by using appropriate encoding. And, *KGen* takes two random polynomials over $F_q$:

$$F(x, z) = \sum_{i=0}^{\omega} \sum_{k=0}^{1} a_{i,0,k} x^i z^k$$

$$mk(x, y, z) = \sum_{i=0}^{\omega} \sum_{j=1}^{\gamma} \sum_{k=0}^{1} a_{i,j,k} x^i y^j z^k,$$

   where each coefficient $a_{i,j,k}$ is chosen uniformly at random from $F_q$, and we define $x^0 = z^0 = 1$. *KGen* also computes $n + 1$ polynomials $e_S^{(0)}(x, z) := F(x, z)$ and $e_i(y, z) := F(x, z)|_{x=R_i} + mk(x, y, z)|_{x=R_i}$ $(1 \leq i \leq n)$. Then, the algorithm *KGen* outputs secret-keys $e_S^{(0)} := e_S^{(0)}(x, z)$, $mk := mk(x, y, z)$ and $e_i := e_i(y, z)$ $(1 \leq i \leq n)$ for $S$, $H$ and $R_i$, respectively.
2. **Device Key-Updating Algorithm *KUpd** and Sender's Key-Updating Algorithm *KUpd*.** For two periods $h \in \tilde{\mathcal{T}}$, $j \in \mathcal{T}$ and $mk = mk(x, y, z)$, the algorithm *KUpd** generates a polynomial $mk^{(h,j)}(x, z) := mk(x, y, z)|_{y=j} - mk(x, y, z)|_{y=h}$. Then, *KUpd** outputs key-updating information $mk^{(h,j)} := mk^{(h,j)}(x, z)$. For $mk^{(h,j)}$ and $e_S^{(h)} := e_S^{(h)}(x, z)$, the algorithm *KUpd* generates the polynomial $e_S^{(j)}(x, z) := e_S^{(h)}(x, z) + mk^{(h,j)}(x, z)$. Then, *KUpd* outputs the secret-key at the period $j$, $e_S^{(j)} := e_S^{(j)}(x, z)$.

3. **Authentication Algorithm *KAuth*.** For a message $m \in F_q$ and $e_S^{(j)} = e_S^{(j)}(x,z)$, the algorithm *KAuth* generates the polynomial $\alpha(x) := e_S^{(j)}(x,z)|_{z=m}$, and outputs the authenticated message at the period $j$, $\alpha := (m, \alpha(x))$.
4. **Verification Algorithm *KVer*.** For an authenticated message at a period $t$, $(\alpha, t)$ $(\alpha = \alpha(x))$, and $e_i = e_i(y,z)$, the algorithm *KVer* outputs *true* if $\alpha(x)|_{x=R_i} = e_i(y,z)|_{y=t,z=m}$ holds, and otherwise outputs *false*.

We can show security of the above construction as follows. The proof of sketch is given in Appendix C.

**Theorem 3.** *The resulting KI-MRA by the above construction is* $(n, w; N, \gamma; \frac{1}{q}, \frac{1}{q})$*-one-time secure. Furthermore, the required memory-sizes of secret-keys and the size of authenticated messages are given as follows.*

$$|\mathcal{E}_S^{(j)}| = q^{2(\omega+1)}, \quad |\mathcal{E}_i| = q^{2(\gamma+1)}, \quad |\mathcal{MK}| = q^{2\gamma(\omega+1)}$$
$$|\mathcal{I}^{(h,j)}| = q^{2(\omega+1)}, \quad |A^{(j)}| = q^{\omega+1}|M|.$$

*Therefore, the above construction is optimal.*

### 3.2 Generic Construction

We propose a generic construction (i.e., a black box construction) of one-time secure KI-MRA by using CFFs (cover free families) and MRA-codes. In general, the merit to consider a generic construction lies in that it is possible to take general settings of parameters in the construction starting from underlying primitives, though it is often the case that generic constructions are inefficient compared to direct constructions. From this point of view, we propose a generic construction of KI-MRA by using both CFFs and MRA-codes. As we will see, our construction is simple and better CFFs and MRA-codes lead to better KI-MRAs. We start with describing the definition of CFF and the model of MRA-codes as follows.

**Definition 4 (CFF [9]).** Let $\mathcal{L} := \{l_1, l_2, \ldots, l_d\}$ be a universal set and $\mathcal{F} := \{F_1, F_2, \ldots, F_N\}$ be a family of subsets of $\mathcal{L}$. Then, we call it $(d, N, \gamma)$*-CFF (Cover Free Family)* if $F_{i_0} \not\subset F_{i_1} \cup F_{i_2} \cup \ldots \cup F_{i_\gamma}$ for all $F_{i_0}, F_{i_1} \ldots, F_{i_\gamma} \in \mathcal{F}$ where $F_{i_j} \neq F_{i_k}$ if $j \neq k$.

A trivial CFF is the family consisting of single-elements subsets, in which we have $N = d$ i.e., $\mathcal{L} = \{1, 2, \ldots, d\}$ and $\mathcal{F} = \{\{1\}, \{2\}, \ldots, \{d\}\}$. We note that there exist non-trivial constructions of CFFs. The constructions of CFFs are studied in various areas in mathematics such as finite geometry, design theory and probability theory. We also note that concrete methods for generating CFFs are given in [9] [15]. Next, we describe a model of MRA-codes in [17].

**MRA-codes.** We consider the scenario where there are $n+1$ entities, a sender $\tilde{S}$ and $n$ receivers $\tilde{R}_1, \ldots, \tilde{R}_n$. The MRA-code $\tilde{\Pi}$ consists of a three-tuple of algorithms (*MGen*, *MAuth*, *MVer*) with four spaces, $\tilde{\mathcal{M}}$, $\mathcal{D}$, $\mathcal{U}$ and $\mathcal{V}$, where $\tilde{\mathcal{M}}$

is a finite set of possible messages, $\mathcal{D}$ is a finite set of possible authenticated messages, $\mathcal{U}$ and $\mathcal{V}$ are finite sets of possible secret-keys for the sender and receivers, respectively. *MGen* is a key generation algorithm, which takes security parameter on input and outputs matching keys $u \in \mathcal{U}$ and $v_i \in \mathcal{V}$ $(i = 1, 2, \ldots, n)$, where $u$ and $v_i$ are secret-keys for $\tilde{S}$ and $\tilde{R}_i$, respectively. *MAuth* is an algorithm for generating an authenticated message, and it is used when the sender wants to broadcast the authenticated message to all verifiers via an insecure broadcast channel. *MAuth* takes a secret-key $u \in \mathcal{U}$ and message $m \in \tilde{\mathcal{M}}$ on input and outputs an authenticated message $\delta \in \mathcal{D}$, and we write $\delta = MAuth(u, m)$ for it. On receiving $\delta$, the receiver $R_i$ can check the validity of it by using *MVer*. *MVer* takes a secret-key $v_i \in \mathcal{V}$ and an authenticated message $\delta \in \mathcal{D}$ on input, and outputs *true* or *false*, where *true* is output if and only if $\delta$ is valid, and we write $true = MVer(v_i, \delta)$ or $false = MVer(v_i, \delta)$ for it. In MRA-codes, there are two kinds of attacks: the *impersonation attack* and *substitution attack*. The formal definitions of those attacks are given in Appendix A.

Our generic construction of KI-MRA is given as follows.

1. **Key Generation Algorithm *KGen*.** For a security parameter $1^k$, the algorithm *KGen* outputs matching secret-keys for $S$, $H$ $R_1, \ldots, R_n$ as follows. *KGen* generates $(d, N, \gamma)$-$CFF$ $\mathcal{L} := \{l_1, \ldots, l_d\}$ and $\mathcal{F} := \{F_1, \ldots, F_N\}$, and makes them public to all entities. And *KGen* calls *MGen* $N$ times with taking on input the security parameter $1^k$. Let $(u_0^{(j)}, v_{1,0}^{(j)}, \ldots, v_{n,0}^{(j)})$ be the $j$-th output from *MGen* $(1 \leq j \leq N)$. Similarly, *KGen* calls *MGen* $d$ times, and let $(u_1^{(l_j)}, v_{1,1}^{(l_j)}, \ldots, v_{n,1}^{(l_j)})$ be the $j$-th output $(1 \leq j \leq d)$[3]. Also, it sets $U^{(0)} := \emptyset$. Then, the algorithm *KGen* outputs secret-keys $e_S^{(0)} := (u_0^{(1)}, u_0^{(2)}, \ldots, u_0^{(N)}, U^{(0)})$, $mk := (u_1^{(l_1)}, u_1^{(l_2)}, \ldots, u_1^{(l_d)})$ and $e_i := (v_{i,0}^{(1)}, v_{i,0}^{(2)}, \ldots, v_{i,0}^{(N)}, v_{i,1}^{(l_1)}, v_{i,1}^{(l_2)}, \ldots, v_{i,1}^{(l_d)})$ for $S$, $H$ and $R_i$, respectively.
2. **Device Key-Updating Algorithm *KUpd** and Sender's Key-Updating Algorithm *KUpd*.** For two periods $h \in \tilde{\mathcal{T}}$, $j \in \mathcal{T}$ and $mk = (u_1^{(l_1)}, \ldots, u_1^{(l_d)})$, *KUpd** generates $U^{(j)} := \{u_1^{(l)} | l \in F_j\}$. Then, *KUpd** outputs the key-updating information $mk^{(h,j)} := U^{(j)}$.

   For $mk^{(h,j)}$ and $e_S^{(h)} = (u_0^{(1)}, \ldots, u_0^{(N)}, U^{(h)})$, *KUpd* generates the secret-key at the period $j$, $e_S^{(j)} := (u_0^{(1)}, \ldots, u_0^{(N)}, U^{(j)})$, and outputs it.
3. **Authentication Algorithm *KAuth*.** For a message $m \in M$ and $e_S^{(j)} = (u_0^{(1)}, \ldots, u_0^{(N)}, U^{(j)})$, *KAuth* generates the authenticated message at the period $j$, $\alpha := (m, \delta_0^{(j)}, \delta_{i_1}^{(j)}, \ldots, \delta_{i_{|F_j|}}^{(j)})$, where $\delta_0^{(j)} := MAuth(u_0^{(j)}, m)$ and $\delta_{i_g}^{(j)} := MAuth(u_1^{(i_g)}, m)$ for all $i_g \in F_j$. *KAuth* then outputs $\alpha$.
4. **Verification Algorithm *KVer*.** For an authenticated message at a period $j$, $(\alpha, j)$, where $\alpha = (m, \delta_0^{(j)}, \delta_{l_1}^{(j)}, \ldots, \delta_{l_{|F_j|}}^{(j)})$ and $e_i = (v_{i,0}^{(1)}, \ldots, v_{i,0}^{(N)}, v_{i,1}^{(l_1)}, \ldots, v_{i,1}^{(l_d)})$, *KVer* outputs *true* if $MVer(v_{i,0}^{(j)}, \delta_0^{(j)}) = true$ and $MVer(v_{i,1}^{(l_g)}, \delta_{l_g}^{(j)}) = true$ for all $l_g \in F_j$, and otherwise outputs *false*.

[3] We note that $(u_1^{(l_j)}, v_{1,1}^{(l_j)}, \ldots, v_{n,1}^{(l_j)})$ is corresponding to $l_j \in \mathcal{L}$.

The security of the above generic construction is shown as follows. The proof of sketch is given in Appendix C.

**Theorem 4.** *Given a $(d, N, \gamma)$-CFF $\mathcal{F}$ and an $(n, \omega, \epsilon)$-secure MRA-code $\tilde{\Pi}$, then the KI-MRA $\Pi$ formed by the above construction based on the CFF and $\tilde{\Pi}$ is $(n, \omega; N, \gamma; \epsilon_A, \epsilon_B)$-one-time secure, where $\epsilon_A \leq \epsilon^{\phi}$ and $\epsilon_B \leq \epsilon$. Here, $\phi := \min(|F_{i_0} - \{F_{i_1} \cup \ldots \cup F_{i_\gamma}\}|)$, where the minimum is taken over all $F_{i_0}$, $F_{i_1}$, $\ldots$, $F_{i_\gamma} \in \mathcal{F}$. Furthermore, required memory-sizes of authenticated messages and secret-keys are given as follows*:

$$|\mathcal{E}_S^{(j)}| = (N + |F_j|)|\mathcal{U}|, \quad |\mathcal{E}_i| = (N + d)|\mathcal{V}|, \quad |\mathcal{MK}| = d|\mathcal{U}|$$
$$|\mathcal{I}^{(h,j)}| = |F_j||\mathcal{U}|, \quad |A^{(j)}| = (|F_j| + 1)|\mathcal{D}|.$$

**Remark 1.** In [17], the generic construction of MRA-codes by combining CFFs and A-codes [10] [21] is proposed. Therefore, by combining our generic construction and the one in [17], KI-MRA can be constructed by using two kinds of simple primitives, CFFs and A-codes.

## 4 Concluding Remarks

In this paper, we studied information-theoretically secure authentication schemes with key-insulated security. Specifically, we introduced a model of information-theoretically secure multireceiver authentication codes (KI-MRA), and proposed security notions and their formalizations in our model. In addition, we derived tight lower bounds of memory-sizes required for the KI-MRA. Furthermore, we provided two kinds of constructions: direct and generic constructions which were provably secure in our security definition. In particular, It was shown that the direct construction was optimal.

In this paper, for simplicity, we discussed the one-time model of KI-MRA. However, it would be interesting to extend our one-time model to the multi-use model of KI-MRA. Also, it would be a future research to study information-theoretically secure key-insulated signature schemes which have security notions stronger than those of KI-MRAs.

## Acknowledgements

We would like to thank anonymous referees for helpful and valuable comments.

## References

1. Anderson, R.: Two remarks on public key cryptology. In: ACM CCCS (1997) (invited Lecture), http://www.cl.cam.ac.uk/users/rja14/
2. Bellare, M., Miner, S.K.: A Forward-Secure Digital Signature Scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)

3. Canetti, R., Goldwasser, S.: An efficient threshold public-key cryptosystem secure against adaptive chosen-ciphertext attack. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 90–106. Springer, Heidelberg (1999)
4. Canneti, R., Halevi, S., Katz, J.: A forward secure public key encryption scheme. J. Cryptology 20(3), 265–294 (2007)
5. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
6. Desmedt, Y., Frankel, Y., Yung, M.: Multi-receiver/Multi-sender network security: efficient authenticated multicast/feedback. In: Proc. of IEEE Inforcom 1992, pp. 2045–2054 (1992)
7. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-Insulated Public-Key Cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
8. Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong Key-Insulated Signature Schemes. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 130–144. Springer, Heidelberg (2002)
9. Erdös, P., Frankl, P., Furedi, Z.: Families of finite sets in which no sets is covered by the union of $r$ others. Israel Journal of Mathematics 51, 79–89 (1985)
10. Gilbert, E.N., MacWilliams, F.J., Sloane, N.J.A.: Codes which detect deception. Bell System Technical Journal 53, 405–425 (1974)
11. Hanaoka, Y., Hanaoka, G., Shikata, J., Imai, H.: Information-Theoretically Secure Key Insulated Encryption: Models, Bounds and Constructions. IEICE Trans. Fundamentals E.87-A(10), 2521–2532 (2004)
12. Hanaoka, G., Shikata, J., Zheng, Y., Imai, H.: Unconditionally secure digital signature schemes admitting transferability. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 130–142. Springer, Heidelberg (2000)
13. Itkis, G., Reyzin, L.: SiBIR: Signer-Base Intrusion-Resilient Signatures. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 499–514. Springer, Heidelberg (2002)
14. Johansson, T.: Further results on asymmetric authentication schemes. Information and Computation 151, 100–133 (1999)
15. Kumar, R., Rajagopalan, S., Sahai, A.: Coding constructions for blacklisting problems without computational assumptions. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 609–623. Springer, Heidelberg (1999)
16. Rivest, R.: Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer (1999) (manuscript), http://people.csail.mit.edu/rivest/Rivest-commitment.pdf
17. Safavi-Naini, R., Wang, H.: Multireceiver authentication codes: model, bounds, constructions and extensions. Information and Computation 151, 148–172 (1999)
18. Shikata, J., Hanaoka, G., Zheng, Y., Imai, H.: Security Notions for Unconditionally Secure Signature Schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 434–449. Springer, Heidelberg (2002)
19. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Compt. 26(5), 1484–1509 (1997)
20. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen-ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)
21. Simmons, G.J.: Authentication theory/coding theory. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 411–431. Springer, Heidelberg (1985)

## Appendix A: Security Notions of MRA-Codes

We describe the security notions of MRA-codes shown in [6], [17], [14]. In MRA-codes, two kinds of attacks are considered: *impersonation attack* and *substitution attack*. We describe the formal definitions of those security notions as follows.

**Definition 5.** [17] *Let $\tilde{\Pi}$ be a MRA-codes. The scheme $\tilde{\Pi}$ said to be $(\tilde{n}, \tilde{\omega}, \tilde{\epsilon})$-secure if* $\max(P_{\tilde{\Pi},I}, P_{\tilde{\Pi},S}) \leq \tilde{\epsilon}$ *where $P_{\tilde{\Pi},I}$ and $P_{\tilde{\Pi},S}$ are defined as follows.*

1) ***Impersonation Attack:*** *The adversary who corrupts at most $\tilde{\omega}$ receivers tries to generate a fraudulent authenticated message $\delta$ that has not been legally generated by the sender $\tilde{S}$ but will be accepted by a receiver $\tilde{R}_i$. Here, we assume that $\tilde{R}_i$ is not included in the colluders, and the adversary can obtain information by pooling secret-keys of corrupted receivers. Success probability of this attack denoted by $P_{\tilde{\Pi},I}$ is defined as follows. For any $\tilde{W} \in \mathcal{P}(\tilde{R}, \tilde{\omega})$ and $\tilde{R}_i \notin \tilde{W}$, we define $P_{\tilde{\Pi},I}(\tilde{R}_i, \tilde{W})$ as follows.*

$$P_{\tilde{\Pi},I}(\tilde{R}_i, \tilde{W}) := \max_{e_{\tilde{W}}} \max_{\delta} \Pr(MVer(v_i, \delta) = true | e_{\tilde{W}})$$

*where the probability is taken over random choice of MGen, and the maximum is taken over: all possible sets of the colluders' secret-keys $e_{\tilde{W}} \in \mathcal{E}_{\tilde{W}}$; and all possible authenticated messages $\delta \in \mathcal{D}$. Then, the probability $P_{\tilde{\Pi},I}$ is defined as $P_{\tilde{\Pi},I} := \max_{\tilde{R}_i, \tilde{W}}(\tilde{R}_i, \tilde{W})$.*

2) ***Substitution Attack:*** *The adversary who corrupts at most $\tilde{\omega}$ receivers tries to generate a fraudulent authenticated message $\delta$ that has not been legally generated by the sender $\tilde{S}$ but will be accepted by a receiver $\tilde{R}_i$, after observing the transmitted authenticated message $\delta'$. Here, we assume that $\tilde{R}_i$ is not included in the colluders, and the adversary can obtain information by pooling secret-keys of corrupted receivers. Success probability of this attack denoted by $P_{\tilde{\Pi},S}$ is defined as follows. For any $\tilde{W} \in \mathcal{P}(\tilde{R}, \tilde{\omega})$ and $\tilde{R}_i \notin \tilde{W}$, we define $P_{\tilde{\Pi},S}(\tilde{R}_i, \tilde{W})$ as follows.*

$$P_{\tilde{\Pi},S}(\tilde{R}_i, \tilde{W}) := \max_{e_{\tilde{W}}} \max_{\delta'} \max_{\delta \neq \delta'} \Pr(MVer(v_i, \delta) = true | e_{\tilde{W}}, \delta')$$

*where the probability is taken over random choice of MGen, and the maximum is taken over: all possible sets of the colluders' secret-keys $e_{\tilde{W}} \in \mathcal{E}_{\tilde{W}}$; and all possible authenticated messages $\delta', \delta \in \mathcal{D}$ such that $\delta \neq \delta'$. Then, the probability $P_{\tilde{\Pi},S}$ is defined as $P_{\tilde{\Pi},S} := \max_{\tilde{R}_i, \tilde{W}}(\tilde{R}_i, \tilde{W})$.*

## Appendix B: Proof of Theorem 2

The proof of Theorem 2 follows from Lemmas 2-6 below.

**Lemma 1.** Let $R_i$ be a receiver and $t$ be a period. For any $W \in \mathcal{P}(\mathcal{R}, \omega)$ and any $\Gamma \in \mathcal{P}(\mathcal{T}, \gamma)$ such that $R_i \notin W$ and $t \notin \Gamma$, we have $P_{\Pi,S_A}(R_i, W, \Gamma, t) \geq 2^{-H(E_i^{(t)}|E_W, E_\Gamma, A^{(t)})}$.

*Proof.* The lemma follows from Theorem 1 straightforwardly, since $I(\tilde{A}^{(t)}; E_i^{(t)} \mid E_W,\ E_\Gamma,\ A^{(t)}) = H(E_i^{(t)}|E_W,\ E_\Gamma, A^{(t)}) - H(E_i^{(t)} \mid E_W,\ E_\Gamma,\ \tilde{A}^{(t)},\ A^{(t)}) \leq H(E_i^{(t)}|E_W,\ E_\Gamma, A^{(t)})$. □

**Lemma 2.** $|\mathcal{E}_S^{(j)}| \geq q^{2(\omega+1)}$ for any $j \in \tilde{\mathcal{T}}$.

*Proof.* We first prove that $H(E_S^{(j)}|E_\Gamma) \geq 2(\omega+1)\log q$ for any $\Gamma \in \mathcal{P}(\mathcal{T}, \gamma)$ and $j \in \mathcal{T}$ with $j \notin \Gamma$. Let $W_i := \{R_1, \ldots, R_{i-1}, R_{i+1}, \ldots, R_{\omega+1}\}$. Then, for any $\Gamma \in \mathcal{P}(\mathcal{T}, \gamma)$ and $j \in \mathcal{T}$ with $j \notin \Gamma$, we have

$$\prod_{i=1}^{\omega+1} P_{\Pi,I_A}(R_i, W_i, \Gamma, j) P_{\Pi,S_A}(R_i, W_i, \Gamma, j) \geq 2^{-\sum_{i=1}^{\omega+1} H(E_i^{(j)}|E_1^{(j)}, \ldots, E_{i-1}^{(j)}, E_\Gamma)} \quad (1)$$

$$= 2^{-H(E_1^{(j)}, \ldots, E_{\omega+1}^{(j)}|E_\Gamma)}$$

$$\geq 2^{-H(E_S^{(j)}|E_\Gamma)}, \quad (2)$$

where (1) follows from Theorem 1 and Lemma 1, and (2) is shown by considering the mapping $\pi^{(j)} : \mathcal{E}_S^{(j)} \to \mathcal{E}_1^{(j)} \times \ldots \times \mathcal{E}_n^{(j)}$.

Since $\prod_{i=1}^{\omega+1} P_{\Pi,I_A}(R_i, W_i, \Gamma, j) P_{\Pi,S_A}(R_i, W_i, \Gamma, j) \leq \prod_{i=1}^{\omega+1} (\frac{1}{q})^2 = (\frac{1}{q})^{2(\omega+1)}$, we obtain $2^{-H(E_S^{(j)}|E_\Gamma)} \leq (\frac{1}{q})^{2(\omega+1)}$ and hence $H(E_S^{(j)}|E_\Gamma) \geq 2(\omega+1)\log q$. Therefore, we have $|\mathcal{E}_S^{(j)}| \geq q^{2(\omega+1)}$, since $H(E_S^{(j)}|E_\Gamma) \leq \log |\mathcal{E}_S^{(j)}|$. □

**Lemma 3.** $|\mathcal{E}_i| \geq q^{2(\gamma+1)}$ for any $i \in \{1, 2, \ldots, n\}$.

*Proof.* Let $\Gamma_j := \{1, \ldots, j-1, j+1, \ldots, \gamma+1\}$. Then, for $W = \emptyset$, we have

$$\prod_{j=1}^{\gamma+1} P_{\Pi,I_A}(R_i, W, \Gamma_j, j) P_{\Pi,S_A}(R_i, W, \Gamma_j, j) \geq \prod_{j=1}^{\gamma+1} 2^{-H(E_i^{(j)}|E_S^{(1)}, \ldots, E_S^{(j-1)})} \quad (3)$$

$$\geq \prod_{j=1}^{\gamma+1} 2^{-H(E_i^{(j)}|E_i^{(1)}, \ldots, E_i^{(j-1)})} \quad (4)$$

$$= 2^{-H(E_i^{(1)}, \ldots, E_i^{(\gamma+1)})}$$

$$\geq 2^{-H(E_i)}. \quad (5)$$

In the above expressions, (3) follows from Theorem 1 and Lemma 1, (4) is shown by considering the mapping $p_i^{(k)} \circ \pi^{(k)} : \mathcal{E}_S^{(k)} \to \mathcal{E}_i^{(k)}$ for $1 \leq k \leq j-1$, where $p_i^{(k)} : \mathcal{E}_1^{(k)} \times \ldots \times \mathcal{E}_n^{(k)} \to \mathcal{E}_i^{(k)}$ is the $i$-th projection, and (5) is shown by considering the mapping $f_i : \mathcal{E}_i \to \mathcal{E}_i^{(1)} \times \ldots \times \mathcal{E}_i^{(N)}$.

Since $\prod_{i=1}^{\gamma+1} P_{\Pi,I_A}(R_i, W, \Gamma_j, j) P_{\Pi,S_A}(R_i, W, \Gamma_j, j) \leq \prod_{i=1}^{\gamma+1} (\frac{1}{q})^2 = (\frac{1}{q})^{2(\gamma+1)}$, we obtain $|\mathcal{E}_i| \geq 2^{H(E_i)} \geq q^{2(\gamma+1)}$. □

**Lemma 4.** $|\mathcal{MK}| \geq q^{2\gamma(\omega+1)}$.

*Proof.* For any $\Gamma = \{k_1, k_2, \ldots, k_\gamma\} \in \mathcal{P}(\mathcal{T}, \gamma)$ with $|\Gamma| = \gamma$, and $j \in \mathcal{T}$ such that $j \notin \Gamma$, we have

$$\begin{aligned} H(MK) &\geq H(MK|E_S^{(j)}) \\ &\geq I(E_S^{(k_1)}, \ldots, E_S^{(k_\gamma)}; MK|E_S^{(j)}) \\ &= H(E_S^{(k_1)}, \ldots, E_S^{(k_\gamma)}|E_S^{(j)}) \\ &= \sum_{i=1}^{\gamma} H(E_S^{(k_i)}|E_S^{(j)}, E_S^{(k_1)}, \ldots, E_S^{(k_{i-1})}) \\ &\geq 2\gamma(\omega+1)\log q, \end{aligned} \tag{6}$$

where (6) follows from the proof of Lemma 2. Therefore, we have $|\mathcal{MK}| \geq 2^{H(MK)} \geq q^{2\gamma(\omega+1)}$. □

**Lemma 5.** $|\mathcal{I}^{(h,j)}| \geq q^{2(\omega+1)}$ for any $h \in \tilde{\mathcal{T}}$ and $j \in \mathcal{T}$.

*Proof.* From the deterministic algorithm (i.e., mapping) $KUpd$: $\mathcal{E}_S \times \mathcal{I} \to \mathcal{E}_S$, it follows that $H(I^{(h,j)}|E_S^{(h)}) \geq H(E_S^{(j)}|E_S^{(h)})$. Thus, we have

$$H(I^{(h,j)}) \geq H(I^{(h,j)}|E_S^{(h)}) \geq H(E_S^{(j)}|E_S^{(h)}) \geq 2(\omega+1)\log q,$$

where the last inequality follows from the proof of Lemma 2. Therefore, we have $|\mathcal{I}^{(h,j)}| \geq 2^{H(I^{(h,j)})} \geq q^{2(\omega+1)}$. □

**Lemma 6.** $|\mathcal{A}^{(j)}| \geq 2^{H(M)} q^{\omega+1}$ *for any* $j \in \mathcal{T}$. In particular, if $\Pr_M$ is the uniform distribution, we have $|\mathcal{A}^{(j)}| \geq q^{\omega+1}|M|$ *for any* $j \in \mathcal{T}$.

*Proof.* For any $j \in \mathcal{T}$ and any $\Gamma \in \mathcal{P}(\mathcal{T}, \gamma)$ such that $j \notin \Gamma$, we have $I(A^{(j)}; E_S|E_\Gamma) \geq I(A^{(j)}; E_1, \ldots, E_{\omega+1}|E_\Gamma)$ by considering the mapping $\pi : \mathcal{E}_S \to \mathcal{E}_1 \times \cdots \times \mathcal{E}_n$. Let $W_i := \{R_1, R_2, \ldots, R_{i-1}\}$. Then, we have

$$\begin{aligned} 2^{-I(A^{(j)};E_S|E_\Gamma)} &\leq 2^{-I(A^{(j)};E_1,\ldots,E_{\omega+1}|E_\Gamma)} \\ &= 2^{-\sum_{i=1}^{\omega+1} I(A^{(j)};E_i|E_1,\ldots,E_{i-1},E_\Gamma)} \\ &\leq \prod_{i=1}^{\omega+1} 2^{-I(A^{(j)};E_i^{(j)}|E_1,\ldots,E_{i-1},E_\Gamma)} \qquad (7) \\ &\leq \prod_{i=1}^{\omega+1} P_{\Pi,I_A}(R_i, W_i, \Gamma, j) \qquad (8) \\ &\leq (P_{\Pi,A})^{\omega+1} \leq \left(\frac{1}{q}\right)^{\omega+1}, \end{aligned}$$

where (7) is shown by considering the mapping $f_i : \mathcal{E}_i \to \mathcal{E}_i^{(1)} \times \ldots \times \mathcal{E}_i^{(N)}$, and (8) follows from Theorem 1. In addition, for $\Gamma = \emptyset$, we have

$$2^{-I(A^{(j)};E_S|E_\Gamma)} = 2^{-H(A^{(j)})+H(A^{(j)}|E_S)} \geq \frac{2^{H(M)}}{|\mathcal{A}^{(j)}|},$$

where the last inequality follows from the deterministic algorithm *KAuth*: $\mathcal{E}_S \times \mathcal{M} \to \mathcal{A}$. Therefore, we obtain $|\mathcal{A}^{(j)}| \geq 2^{H(M)} q^{\omega+1}$. □

## Appendix C: Sketch Proofs

*Sketch Proof of Theorem 1.* The proof can be shown in a way similar to the proof of Theorem 3.2 in [17]. Here, we only show a sketch proof of the first inequality. We define a characteristic function $\mathcal{X}_{I_A}$ as follows.

$$\mathcal{X}_{I_A}((\alpha,t), e_i^{(t)}, e_W, e_\Gamma) = \begin{cases} 1 \text{ if } KVer(e_i, \alpha, t) = true \wedge \Pr((\alpha,t), e_i^{(t)}, e_W, e_\Gamma) \neq 0 \\ 0 \text{ otherwise.} \end{cases}$$

Then, from Definition 2, we can express $P_{\Pi,I_A}(R_i, W, \Gamma, t)$ as

$$P_{\Pi,I_A}(R_i, W, \Gamma, t) = \max_{e_W} \max_{e_\Gamma} \max_{(\alpha,t)} \sum_{e_i^{(t)}} \mathcal{X}_{I_A}((\alpha,t), e_i^{(t)}, e_W, e_\Gamma) \Pr(e_i^{(t)} | e_W, e_\Gamma).$$

By a way similar to the proof of Theorem 3.2 in [17], we have $P_{\Pi,I_A}(R_i, W, \Gamma, t) \geq 2^{-I(A^{(t)}; E_i^{(t)} | E_W, E_\Gamma)}$. Similarly, other inequalities can also be proved. □

*Sketch Proof of Theorem 3.* The proof can be directly shown as in the proofs of constructions of MRA (For example, see [17] [14]). Here, we only describe the outline of the proof of $P_{\Pi,S} \leq \frac{1}{q}$, since other ones can be shown by a similar idea. To succeed in the substitution attack in the exposure type A, the adversary will generate a fraudulent authenticated message at a period $t$ $(\alpha, t)$, where $\alpha = (m, \alpha(x))$, under the following conditions: the adversary can obtain $\gamma$ exposed secret-keys for the sender, $\omega$ secret-keys for corrupted receivers and a valid authenticated message at the same period $(\alpha', t)$, where $\alpha' \neq \alpha$. However, the degrees of $F(x,z) + mk(x,y,z)$ with respect to $x$, $y$ and $z$ are at most $\omega$, $\gamma$ and 1, respectively. Thus, for the message $m$, the adversary cannot guess the polynomial $\alpha(x) = F(x,m) + mk(x,t,m)$ with probability more than $1/q$. Therefore, we have $P_{\Pi,S_A} \leq \frac{1}{q}$. In a manner similar to this, we can prove that $P_{\Pi,I_A} \leq \frac{1}{q}$. Thus, we have $P_{\Pi,A} = \max(P_{\Pi,I_A}, P_{\Pi,S_A}) \leq \frac{1}{q}$. Similarly, we can also prove that $P_{\Pi,B} = \max(P_{\Pi,I_B}, P_{\Pi,S_B}) \leq \frac{1}{q}$. Furthermore, it is straightforward to evaluate memory-sizes required in the construction. □

*Sketch Proof of Theorem 4.* The proof of security can be directly shown by the definition of CFF and security of MRA. Here, we only describe the outline of the proof of $P_{\Pi,S_A} \leq \epsilon^\phi$, since other ones can be shown by a similar idea. The adversary can know $\gamma$ sets $U^{(1)}, U^{(2)}, \ldots, U^{(\gamma)}$ from $\gamma$ exposed secret-keys for the sender. However, from the definition of CFF, the adversary cannot obtain at least $\phi$ elements of the set $U^{(t)} = \{u_1^{(l)} | l \in F_t\}$. Therefore, the adversary needs to forge at least $\phi$ authenticated messages of the underlying MRA-code. Thus, we have $P_{\Pi,S_A} \leq \epsilon^\phi$. In a manner similar to this, we can prove that $P_{\Pi,I_A} \leq \epsilon^\phi$. Thus, we have $P_{\Pi,A} = \max(P_{\Pi,I_A}, P_{\Pi,S_A}) \leq \epsilon^\phi$. Similarly, we can also prove that $P_{\Pi,B} = \max(P_{\Pi,I_B}, P_{\Pi,S_B}) \leq \epsilon$. Furthermore, it is straightforward to evaluate memory-sizes required in the construction. □

# Simple and Communication Complexity Efficient Almost Secure and Perfectly Secure Message Transmission Schemes

Yvo Desmedt[1,2,⋆], Stelios Erotokritou[1,⋆⋆], and Reihaneh Safavi-Naini[3,⋆⋆⋆]

[1] Department of Computer Science, University College London, UK
{y.desmedt,s.erotokritou}@cs.ucl.ac.uk
[2] Research Center for Information Security (RCIS), AIST, Japan
[3] Department of Computer Science, University of Calgary, Canada
rei@ucalgary.ca

**Abstract.** Recently Kurosawa and Suzuki considered almost secure (1-phase $n$-channel) message transmission when $n = (2t + 1)$. The authors gave a lower bound on the communication complexity and presented an exponential time algorithm achieving this bound. In this paper we present a polynomial time protocol achieving the same security properties for the same network conditions.

Additionally, we introduce and formalize new security parameters to message transmission protocols which we feel are missing and necessary in the literature.

We also focus on 2-phase protocols. We present a protocol achieving perfectly secure message transmission of a single message with $O(n^2)$ communication complexity in polynomial time. This is an improvement on previous protocols which achieve perfectly secure message transmission of a single message with a communication complexity of $O(n^3)$.

**Keywords:** Information-theoretic cryptography, Perfectly secure message transmission, Almost secure message transmission, Cryptographic protocols, Privacy, Reliability, Coding theory, Secret sharing.

## 1 Introduction

Perfectly secure message transmission (PSMT) schemes were introduced by Dolev et al. in [2]. Such protocols aim to provide perfect privacy and perfect reliability of message transmission between two parties (a sender and a receiver) who do not share a key, in the presence and influence of an infinitely powerful

⋆ This work was done while funded by EPSRC EP/C538285/1 and by BT, as BT Chair of Information Security.
⋆⋆ The author would like to thank EPSRC and BT for their funding of this research.
⋆⋆⋆ This research was supported in part by iCORE (Informatics Circle of Research Excellence) in the Province of Alberta, as well as NSERC (Natural Sciences and Engineering Research Council) in Canada.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 166–183, 2010.

active adversary. These protocols are important as they are fundamental cryptographic protocols with important applications in research [3,5]. In the ($r$-phase, $n$-channel) model, a sender wishes to securely send a secret to a receiver. This transmission should occur in $r$ phases with $n$ channels connecting the sender and receiver. The adversary is able to corrupt up to $t$ network nodes.

In [2] it was shown that if $r = 2$, $n \geq (2t + 1)$ is a necessary and sufficient condition for PSMT. In this paper we present a protocol which is the most efficient in the literature for the transmission of a single message. Our protocol achieves this in polynomial time and with $O(n^2)$ communication complexity. This improves on the $O(n^3)$ communication complexity of previous protocols. Optimum results for two phase protocols with regards to transmission rate have only recently been achieved. One of these results was that of [1] at CRYPTO 2006. The paper presented a protocol with $O(n)$ transmission rate but exponential computational complexity. These results were improved at EUROCRYPT 2008 in [7] where the authors presented a polynomial 2-phase PSMT protocol able to achieve $O(n)$ transmission rate. The communication complexity for this protocol is however $O(n^3)$, while the one we present is $O(n^2)$. Despite the work of [7] which achieves linear transmission rate the protocol they present is only optimal for communication with a large number of messages. For applications which require the perfectly secure transmission of a single message, the protocol we present is the most efficient in the literature.

Different variations on the security of message transmission schemes exist in the literature [2,6,9,13]. In [2] the authors showed that a necessary and sufficient condition for PSMT to occur when $r = 1$ is that $n \geq 3t+1$. For these conditions the authors also provided a protocol achieving perfect security with optimum communication complexity. In [4] the authors considered perfect security with probabilistic reliability. In such protocols, reliability of message transmission may fail with a bounded probability. The work was important as it presented the sizable gap between the connectivity required to achieve perfect as opposed to probabilistic security. In [8] Kurosawa and Suzuki considered almost secure (1-phase, $n$-channel) message transmission when $n = 2t+1$. The authors showed that almost secure message transmission can be achieved provided we are willing to accept a small probability that the protocol will fail. A lower bound in communication complexity for this type of transmission was given and an exponential time protocol achieving this bound was presented. Srinathan et al. [12] improved on this by presenting a polynomial time protocol achieving similar security. We also present a polynomial time almost secure message transmission protocol. Contrary to the protocol presented in [12] which can transmit $O(n)$ messages, the protocol we present is for the transmission of a single message only. The protocol we present is computationally more efficient requiring in the order of $n$ less computation than the protocol of [12]. For applications where the almost secure transmission of a single message is required, the protocol we present is the most efficient and appropriate to use.

Both protocols we present are of interest to both theoretical and practical purposes. Whereas previous protocols have sought to optimize transmission rate

only [7,8,12], we have concentrated on making protocols more efficient with regards to communication and computation complexities. The protocols we present are theoretically the most efficient regarding these complexities. They are also important for practical purposes. In real world practical applications, computational cryptography is generally used. When these are not available or are broken, in order to initialize these, new keys will have to be exchanged between communicating parties. In such circumstances it makes no sense to send multiple messages.

In this paper we also look at different security definitions for message transmission schemes which exist in the literature. These seem to vary from paper to paper and different aspects to security of message transmission are termed in a different way by various authors. We aim to resolve this confusion by introducing and formalizing new security parameters to message transmission protocols which we feel are missing and necessary in the literature.

## 2 Background

### 2.1 Environment of Message Transmission Protocols

In a message transmission protocol the sender starts with a message $M^A$ drawn from a message space $\mathcal{M}$ with respect to a certain probability distribution. We assume that the message space $\mathcal{M}$ is a subset of a finite field $\mathbb{F}$. At the end of the protocol, the receiver outputs a message $M^B$. We consider a synchronous setting in which messages between sender and receiver are sent in phases. A phase is thus a transmission from sender to receiver or vice-versa.

We will assume an unconditionally secure setting where the adversary is computationally unbounded. We consider an active adversary which can take control of $t$ internal nodes in a network. The adversary is assumed to know the complete protocol specification, message space $\mathcal{M}$ and the complete structure of the network graph. We will only consider static adversaries which must choose the nodes to corrupt before the protocol begins. The adversary is able to view all the behavior (randomness, computation, messages received) and can take full control of any node which is compromised.

The communication network is modeled as a directed graph $G = G(V, E)$ whose nodes are the parties and whose edges are point-to-point reliable and private communication channels. The network model which will be considered consists of bidirectional channels between a sender and a receiver. Disjoint paths between a sender and a receiver are referred to as wires and will be termed so from now on. As the adversary can take control of $t$ internal nodes in a network, the adversary has the capability to control $t$ wires.

### 2.2 Security Definition

Here we present the current security definitions for message transmission protocols which exist in the literature. We argue why we think these definitions are incomplete and formalize new security parameters we believe should be included.

**Current Security Definition - Probabilistic Reliability.** The following definition is taken from Franklin and Wright [4] and defines $(\epsilon, \delta)$-security.

1. Let $\delta < \frac{1}{2}$. A message transmission protocol is $\delta$-reliable if, with probability at least 1 - $\delta$, B terminates with $M^B = M^A$. The probability is over the choices of $M^A$ and the coin flips of all nodes.
2. A message transmission protocol is perfectly reliable if it is 0-reliable.
3. $\epsilon$ refers to the privacy that is achieved. As we will be considering perfect privacy we refer the reader to [4] for the definition of $\epsilon$-privacy.
4. A message transmission protocol is $(\epsilon, \delta)$-secure if it is $\epsilon$-private and $\delta$-reliable.

**Current Security Definition - Probabilistic Failure.** Almost secure message transmission was considered in [8] with the following security definition:

**Definition 1.** *We say that a (1-phase, n-channel) message transmission scheme is $(t, \delta)$-secure if the following conditions are satisfied for any adversary* ***A*** *who can corrupt at most t out of n channels.*

***Privacy.*** ***A*** *learns no information on $M^A$. More precisely when $R$ denotes the random variable,*

$$Pr(R = M^A | X_{i_1} = x_{i_1}, \cdots, X_{i_t} = x_{i_t}) = Pr(R = M^A)$$

*for any $M^A \in \mathcal{M}$ and any possible $x_{i_1}, \cdots, x_{i_t}$ messages observed by* ***A*** *on adversary controlled wires.*

***General Reliability.*** *The receiver outputs $M^B = M^A$ or $\perp$ (failure). The receiver thus never outputs a wrong secret.*
***Failure.*** *$Pr(\text{Receiver outputs } \perp) < \delta$.*

**Comparison of the Two Security Definitions.** The above security definitions refer to two very different security properties. The definition of reliability from [4] only considers executions that terminate and always output a message. The protocol outputs the correct message with a bounded probability. However, the reliability definition of [8] identifies two types of executions. The first of these are executions that terminate but do not produce an output (i.e. output $\perp$). The second of these are executions that terminate and always produce the correct result. Reliability in these definitions refers to both types of executions.

We propose a new security definition that reconciles the above definitions. This is done by introducing new security parameters which capture the *authenticity* of the received message and the *availability* of the message transmission protocol. Executions that terminate but do not produce a correct output can be seen as an attack on authenticity of the message. Executions that output failure can be seen as an attack on the availability of the transmission protocol. This leads us to propose the following definition.

**New Security Definition.** The first security parameter we call the availability of a transmission protocol and is defined as follows:

– Let $\gamma \leq 1$. A message transmission protocol achieves $\gamma$-availability if, with probability at least 1 - $\gamma$, B accepts a message, i.e. B accepts $\perp$ with probability $\gamma$.

Availability considers executions that were captured by the $\delta$ parameter in the security definition of [8].

Authenticity is the second security parameter we introduce. Assuming the sender transmits $M^A$ and the receiver accepts $M^B \in \{\mathcal{M}, \perp\}$, $\delta$-authenticity is achieved by the following conditional probability.

$$\delta = P(M^A \neq M^B | ReceiverAccepts).$$

The above covers both the reliability definition of [4] and the executions of [8] which terminate and always produce the correct result.

$\epsilon$-privacy remains the same as defined in [4]. This type of security from now on will be referred to as $(\epsilon, \delta, \gamma)$-security. It should be pointed out that perfectly secure message transmission protocols (i.e. the work of [2,6,10]) achieve $(0, 0, 0)$-security under this new definition. In our work we will present a polynomial 1-phase $(0, 0, \gamma)$-secure protocol in Section 3 and in Section 5 we consider 2-phase $(0, 0, 0)$-secure protocols.

### 2.3 Secret Sharing Schemes

An $m$-out-of-$n$ threshold secret sharing scheme allows for a secret message $M^A$ to be distributed as a selection of $n$ shares $\{s_1,\ldots,s_n\}$ so that the following properties are achieved:

1. Any collection of $m$ shares is able to reconstruct the secret message $M^A$.

2. Any subset of $(m-1)$ or less shares reveals no information about $M^A$.

**Variant of Shamir Secret Sharing Scheme.** Shamir secret sharing [11] allows for the reconstruction of a secret using polynomial interpolation. When secret sharing is used in the literature it is assumed that the $x$-coordinates of points to be used are available in a public database and shares sent to participants (or in our case across wires) are the $y$-coordinates of the points. When using Shamir secret sharing in message transmission protocols, denoting as $p$ the polynomial from which the shares are constructed, it is usually the norm to transmit share $p(i)$ across wire $w_i$ - where $1 \leq i \leq n$ and $n$ denotes the number of wires connecting sender and receiver.

In this paper we will use a variant of the above for the protocol of Section 3. The only thing that will vary is that the $x$-coordinates of points to be used will be private and shares sent will be the $(x, y)$-coordinates of the points.

## 3 Polynomial 1-Phase Almost Secure Message Transmission

We now describe our 1-phase almost secure message transmission protocol for $n = (2t + 1)$. As our protocol is relatively simple we first present the main idea

of our protocol. We then describe the main techniques used in the protocol. We then formally present the security and complexity proof - showing that it is a polynomial algorithm regarding computation and communication complexities.

### 3.1 Main Idea

As our protocol is a 1-phase protocol, communication can only occur one way from sender to receiver. Denoting as $M^A$ the secret message of the transmission, the sender will construct a $(t+1)$-out-of-$n$ secret sharing code of $M^A$. The sender thus has $n$ shares $(s_1, \ldots, s_n)$ of $M^A$. For each one of the $n$ shares the sender will then carry out a $(t+1)$-out-of-$n$ secret sharing (these new shares are termed as sub-shares for clarity) and will transmit these sub-shares to the receiver - the way this is done is outlined in Section 3.2. The receiver will then check the correctness of the shares of $M^A$ and considering only the correct shares will proceed to carry out error detection. This is also outlined in Section 3.2. If no error is detected the receiver accepts the message interpolated by the shares. If at least one error is detected, the receiver accepts $\perp$. When the receiver accepts a value the protocol terminates. As in some cases the receiver accepts $\perp$ and in all other cases the receiver accepts the correct message with perfect secrecy and authenticity the protocol achieves $(0, 0, \gamma)$-security.

### 3.2 Main Protocol Techniques

In this section we outline three main techniques used in the protocol. The first is the encoding and transmission of the secret message executed by the sender. The other two are carried out by the receiver and are the identification of faulty wires and error detection schemes.

**Message Encoding and Transmission.** Denoting as $M^A$ the secret message of the transmission, the sender will carry out a $(t+1)$-out-of-$n$ secret sharing of $M^A$ - obtaining shares $(s_1, \ldots, s_n)$. The sender does this by choosing a random polynomial $p$ of degree at most $t$ over $GF(q)$ such that $p(x) = M^A + a_1x^1 + \cdots + a_tx^t$ - where $a_1, \ldots, a_t$ are uniformly random elements of $GF(q)$ and $q \gg n$ denotes the size of the finite field. The $n$ shares $(s_1, \ldots, s_n)$ are obtained by evaluating $(p(1), \ldots, p(n))$.

For $1 \leq i, j \leq n$ the sender proceeds to construct a $(t+1)$-out-of-$n$ secret sharing scheme of share $s_i$ and transmit the constructed shares in the following manner:

1. The sender chooses a random polynomial $p_i$ of degree at most $t$ over $GF(q)$ such that $p_i(x) = s_i + a_{i1}x^1 + \cdots + a_{it}x^t$ where $a_{i1}, \ldots, a_{it}$ are uniformly random elements of $GF(q)$.
2. $n$ different uniformly random elements $(r_{i1}, \ldots, r_{in})$ over the finite field are then selected.
3. The $n$ sub-shares of $s_i$ are computed by evaluating $p_i$ at $(r_{i1}, \ldots, r_{in})$ to obtain $(s_{i1} = p_i(r_{i1}), \ldots, s_{in} = p_i(r_{in}))$.
4. The random elements with the corresponding sub-shares are coupled together to obtain $((r_{i1}, s_{i1}), \ldots, (r_{in}, s_{in}))$.

5. The definition of polynomial $p_i$ is transmitted over wire $w_i$.
6. The sender transmits the pair $(r_{ij}, s_{ij})$ over wire $w_j$.

**Faulty Wire Detection.** This technique is carried out by the receiver at the end of the protocol phase. Identification of faulty wires is done in the following way:

- Initialize set $FAULTY := \emptyset$, $REPEAT_{FLAG} := TRUE$.
- Do the following while $(REPEAT_{FLAG} = TRUE)$:
  **a** $REPEAT_{FLAG} := FALSE$.
  **b** For $i := 1, \ldots, n$
    1. **IF** wire $w_i \in FAULTY$ GOTO Step 6.
    2. Denote as $p_i$ the polynomial definition received from wire $w_i$.
    3. Considering only the $i^{th}$ pair of values - $(r_{ji}, s_{ji})$, received from wires $w_j \notin FAULTY$.
    4. **IF** $p_i(r_{ji}) = s_{ji}$ for at least $(t+1)$ pair of values received from different wires, then do nothing.
    5. **ELSE** wire $w_i$ is identified as a faulty wire. Add $w_i$ to $FAULTY$. Set $REPEAT_{FLAG} := TRUE$.
    6. End of loop.

The faulty wire detection scheme described above identifies faulty wires. However, it cannot guarantee that wires identified by the scheme as non-faulty are not controlled by the adversary and that changes were not carried out. As will be outlined later, adversary controlled wires can still pass the test of this scheme - even if changes were carried out. In short, adversary controlled wires can achieve this if any alterations carried out on polynomial definitions still result in the algorithm finding at least $(t+1)$ pair of values passing the test of Step 4.

It is easy to see that the computational complexity of the above scheme is polynomial. The while loop can be repeated at most $t$ times (as overall at most $t$ faulty wires can be identified in different instances of the while loop).

**Error Detection.** Error detection is carried out in the following manner. Considering only wires $w_i \notin FAULTY$, evaluate $p_i(0)$ for the polynomial received from wire $w_i$ to obtain share $s_i$ and denote as $m$ the number of shares obtained. Carry out error detection on the $m$ shares in the following manner. Select $t+1$ random shares from the $m$ shares to obtain polynomial $p$. If any of the remaining $m-(t+1)$ shares do not lie on $p$ then an error has been detected and the receiver outputs $\perp$. Otherwise the receiver accepts $p(0)$ of the obtained polynomial as the message of the transmission.

It is clear to see that the computational requirements of this error detection process is polynomial.

### 3.3 Security and Efficiency

**Theorem 1.** *The above protocol achieves* $(0, 0, \gamma)$ *security for appropriately large* $q$.

*Proof.* We first prove the perfect privacy of the protocol. The secret message $M^A$ is secret shared using a $(t+1)$-out-of-$n$ secret sharing scheme. The adversary can only learn $t$ of these shares - those whose polynomial definitions are sent on adversary controlled wires. For the secret sharing of the remaining shares the adversary only learns $t$ sub-shares of each and thus cannot learn any of these shares. As the adversary learns $t$ shares, no information is obtained about the secret message. The protocol thus achieves perfect privacy.

Perfect authenticity is achieved as the receiver accepts a message only when no errors are detected. This is proven by the following lemma.

**Lemma 1.** *The faulty wire detection scheme will always identify as correct wires the set of $t+1$ non-faulty wires.*

*Proof.* The faulty wire detection scheme identifies as a correct wire those whose polynomial definitions correspond to at least $(t+1)$ received sub-shares. As honest wires do not alter any information and as there are at least $(t+1)$ honest wires, honest wires will always be identified as correct wires. □

Due to the above lemma, at least $(t+1)$ original shares of $M^A$ will be considered by the sender in the error detection scheme. Because of this, no matter what changes the adversary carries out the receiver will never accept a wrong message. This is because the degree of the polynomial used in the secret sharing of $M^A$ is at most $t$. This means that any alterations the adversary carries out cannot result in a different $t$-degree polynomial which includes all the honest $(t+1)$ shares (corresponding to honest wires) of $M^A$ . Therefore no matter what alterations the adversary carries out to share values, at least one error will always be detected. The receiver will thus never accept a message different to the message sent by the sender. Perfect authenticity is therefore achieved.

We now calculate the availability. Failure of message transmission (i.e. when the receiver outputs $\perp$) occurs when an error is detected in the error detection scheme of Section 3.2. An error is detected by this technique only when the adversary successfully alters at least one share of the secret message. This can only be achieved when a polynomial definition transmitted over a faulty wire is altered and after the execution of the faulty wire detection technique, the specific wire does not belong in the set $FAULTY$. For this to occur the adversary must ensure that at least $(t+1)$ sub-shares received over all wires lie on the altered polynomial. Assuming that the adversary controls $t$ wires (all of which do not belong in the set $FAULTY$) - and in turn controls $t$ sub-shares, any strategy followed by the adversary must ensure that *at least one sub-share* not controlled by the adversary lies on the polynomial definition to be altered by the adversary.

The reader is reminded that shares in the scheme are a pair $(r_x, s_y)$ where $r_x$ is a random field element representing the $x$-value of a point on a 2-dimensional plane and $s_y$ is the evaluation of $r_x$ for a particular polynomial. As the adversary knows the polynomial definition transmitted on adversary controlled wires, the adversary knows the value of all possible sub-shares ($(r_x, s_y)$ pairs) that could be constructed using that polynomial. What the adversary does not know is which specific $(t+1)$ sub-shares were transmitted over the honest wires. As the values

$r_x$ were uniformly randomly chosen by the sender, in order for the adversary to succeed in an attack against the protocol availability, the best strategy the adversary could follow is to guess these $r_x$ values. (Note that from the original polynomial and a correct $r_x$ value, the adversary knows the corresponding $s_y$.) Given that the adversary controls $t$ wires the adversary is able to rule out $t$ of these $r_x$ possible values over the finite field - leaving $(q-t)$ possible values for $r_x$ sent over the $(t+1)$ honest wires.

As mentioned earlier, it is sufficient that *at least one sub-share* not controlled by the adversary lies on the altered polynomial. The adversary can try to guess up to $t$ of these sub-shares[1]. The probability that at least one is successful, is one minus the probability that all $t$ guesses are wrong. All being wrong means, that all $t$ were not chosen by the sender, which obviously means they came from the $q-(2t+1)$ remaining ones. So, the probability of success is then given by:

$$1 - \frac{\binom{q-2t-1}{t}}{\binom{q-t}{t}}$$

The above analysis is for the case the adversary decides to alter a single polynomial. If the adversary were to change the polynomial definition of two wires and the attack of one wire failed, this would mean that for the other wire attack to succeed, it would require for two sub-shares transmitted over honest wires to be guessed correctly. This makes the full analysis more complex and due to space reasons will appear in the full version of the text. □

We now analyze the complexity of the protocol. Denoting as $|\mathbb{F}|$ the bit length of the field elements, the communication complexity of the protocol is $O(n^2|\mathbb{F}|)$. The computational complexities of both sender and receiver are polynomial.

We have thus presented a polynomial almost secure polynomial protocol improving on the exponential time protocol presented in [8].

## 4 Comparison of Protocol to Previous Work

In this section we compare the protocol presented in the previous section to previous work and argue as to why it is a valuable addition to the knowledge.

**Comparison of Protocol to Suzuki and Kurusawa Protocol.** The work presented by Suzuki and Kurusawa in ICITS 2007 [8] was important as it proved the lower bound of communication complexity required for almost secure message transmission. Despite the authors presenting a protocol achieving this bound,

[1] As the degree of the polynomial is at most $t$ the adversary cannot guess more that $t$ of the remaining sub-shares. This is because $(t+1)$ correct sub-shares interpolate the original polynomial. If the adversary guesses more than $t$ sub-shares and more than $t$ are correct this means that the original polynomial will be interpolated and in effect the adversary carries out no changes.

the computational complexity of the protocol was exponential which makes the protocol inefficient with regard to time for large values of $n$.

The protocol presented in the previous section, despite having a greater communication complexity of $O(n^2)$ - as opposed to the optimal $O(n)$, is of polynomial computation time making it a more appropriate protocol for use.

**Comparison of Protocol to Srinathan et al. Protocol.** The work presented by Srinathan et al. in PODC 2008 [12] was important as it presented an almost secure message polynomial time transmission protocol which achieves the optimal transmission rate of $O(n)$ with a communication complexity of $O(n^2)$. For the protocol to achieve this transmission rate $O(n)$ messages are transmitted.

The protocol presented in the previous section also has a communication complexity of $O(n^2)$, but only a single message is transmitted.

When these two protocols are compared it may seem that the Srinathan et al. protocol is the best protocol to use as more messages can be sent with the same communication complexity.

Even though this is true, if the two protocols were to be compared with respect to their computational complexities, it will be found that the protocol presented in the previous section requires much less computation to complete than the Srinathan et al. protocol.

The protocol of Srinathan et al. decides on whether to accept a message or $\perp$ using error detection. For the Srinathan et al. protocol to accept a message a total of $n$ error detections need to be carried out. For $\perp$ to be accepted between one and $n$ error detections need to be carried out - the actual number depending on the actions of the adversary. Contrary to this, the protocol presented in the previous section accepts a message or $\perp$ within only a single error detection. Because of this, the protocol presented in this paper requires in the order of $n$ less computation than the protocol of Srinathan et al.

This makes the presented protocol very useful for situations where the almost secure transmission of a single message is required. This can include the transmission of an encryption key in wireless sensor networks. In such a situation the keys can be transmitted with the least amount of computation carried out by receivers - in this case wireless sensors, which is an important factor for the preservation of the wireless sensor battery life.

**Note: Transforming the Desmedt-Wang Protocol to a $(0, 0, \gamma)$ Protocol.** We now describe how to transform the protocol presented in Section 3 of [15] as a $(0, \delta)$ protocol (following the security definition of [4]) to a $(0, 0, \gamma)$ protocol with $\gamma = \delta$.

The protocol works by sharing a secret using a $(t + 1)$-out-of-$(2t + 1)$ secret sharing scheme. Using message authentication codes (MAC's) each share is authenticated $(2t + 1)$ different times using authentication keys specific to each wire. Each share is then sent to the receiver only once and upon each wire only one share is sent. The authentication codes of the shares are sent over the same wire the share is sent on and the authentication keys are sent on their respective wires. Although not stated in the description of the protocol, this process can be carried out in a single phase. Correct shares are classified as those shares which

are authenticated by at least $(t+1)$ different MAC's. As the authors considered $(0, \delta)$ security the protocol reliability fails with a small probability.

The protocol can be transformed to a $(0, 0, \gamma)$ protocol by simply carrying out error detection on the correct shares in the same way as described in Section 3.2. The same decision as described in the presented error detection scheme will also be taken.

With this transformation, this protocol also becomes a one phase $(0, 0, \gamma)$ protocol for a single message with a communication complexity of $O(n^2)$. In effect this protocol is equivalent to the $(0, 0, \gamma)$ protocol presented in the previous section[2].

## 5 Efficient Perfectly Secure Message Transmission

We now turn our attention to two-phase perfectly secure message transmission. The protocol we present achieves perfectly secure message transmission of a single message with $O(n^2)$ communication complexity and transmission rate in polynomial time. This greatly improves on previous protocols [7,10] which achieve this with $O(n^3)$ communication complexity and transmission rate.

We first present the main idea of our initial protocol and proceed to describe the main techniques that will be used in the protocol. In Section 5.3 we formally present our protocol and then present the security and complexity proof.

### 5.1 Main Idea

As our protocol is a two phase protocol, like most two phase protocols, in the first phase the receiver will send random elements of the finite field to the sender.

At the end of the first phase, the sender will observe the received data and identify possible errors that may have occurred in the transmission of the first phase. Different types of errors may have occurred and these will be outlined in Section 5.2. These errors will be sent via broadcast to the receiver in the transmission phase of the second phase. The sender will also send via broadcast correcting information so that the random elements sent in the first phase will constitute shares of the secret shared message of the transmission.

At the end of the second phase, using the identified errors, the receiver is able to identify all wires which were active during the transmission of the first phase. Using the correcting information, the receiver is able to securely obtain the secret message of the communication. The receiver is able to do this as it can ignore the shares that correspond to the identified faulty wires.

### 5.2 Main Protocol Techniques

**Broadcast.** Broadcast will be used in the second phase of our protocol. When the sender broadcasts information, the sender will send the same information over all $n$ wires which connect the sender and receiver. As the adversary is able to corrupt at most $t$ of these $n$ wires, the receiver correctly receives the information via a majority vote.

[2] Using other authentication codes it is trivial to lower the transmission complexity.

**Transmission of Random Elements.** We now describe how random elements are sent from the receiver to the sender in the first phase of the protocol [3]. For simplicity we first describe the transmission of one random element $r$.

The receiver constructs the shares of $r$ using a $(t+1)$-out-of-$n$ secret sharing scheme. The receiver thus chooses a random polynomial $p$ of degree at most $t$ such that $p(x) = r + a_1x^1 + \cdots + a_tx^t$. The $n$ shares $(s_1, s_2, \ldots, s_n)$ are computed by evaluating $p(x)$ at $x_1, x_2, \ldots, x_n$.

The receiver then proceeds to send share $s_i$ via wire $w_i$ $(1 \leq i \leq n)$. The receiver also transmits the $t$ coefficients $(a_1, \ldots, a_t)$ and $r$ - which define $p$, across a single wire.

In our protocol $n$ parallel executions of the above will be carried out. $n$ random elements $(r_1, \ldots, r_n)$ will be selected. The corresponding $n$ random polynomials $(p_1, \ldots, p_n)$ will also be constructed. For each random element, using the corresponding polynomial $n$ shares will be constructed. For reasons of clarity we denote as $(s_{i1}, \ldots, s_{in})$ the $n$ shares for the $i^{th}$ random element. Upon each of the $n$ wires, $n$ shares will be transmitted as will the definition of a single polynomial. The definition of polynomial $p_i$ will be transmitted on wire $w_i$ $(1 \leq i, j \leq n)$. The $s_{ij}$ share constructed from this polynomial will be sent on wire $w_j$.

**Error Detection and Identification of Faulty Wires.** The above technique described what will occur and what will be transmitted in the first phase of the protocol. At the end of the first phase, the sender will receive $n$ shares and a definition of a polynomial from each wire. Using this, the sender carries out error detection as follows.

For $i, j := 1, \ldots, n$ and for polynomial $p_i$ received from wire $w_i$ the sender considers the $n$ shares received as the $i^{th}$ share from each wire. The sender checks each of the shares and identifies as error shares the shares whose value does not agree with the definition of polynomial $p_i$. Share $s_{ij}$ received from wire $w_j$ is thus identified as an error share if $s_{ij} \neq p_i(x_j)$. The sender proceeds to broadcast the identified error shares to the receiver [4].

We now show that with this information the receiver can identify wires that were active in the first phase of the protocol. For clarity we assume that each error share is denoted as $es_{ij}$ - with $j$ indicating the wire $w_j$ and $i$ indicating the position from which the share was received by the sender $(1 \leq i, j \leq n)$. The $i$ position of the share indicates that the share corresponds to the $i^{th}$ polynomial received by the sender (from wire $w_i$).

The receiver checks the following cases to identify faulty wires:

**Case 1:** If the value of error share $es_{ij}$ is different to the corresponding share sent out by the receiver in phase 1 then wire $j$ is identified as a faulty wire.

**Case 2:** If the value of error share $es_{ij}$ is equal to the corresponding share sent out by the receiver in phase 1 then wire $i$ is identified as a faulty wire.

[3] This scheme is similar to the encoding scheme of Section 3.2. It also resembles techniques used in [14].

[4] This requires the sender to send a triple $(j, i, v)$ to the receiver for each error share - indicating the $i^{th}$ share of wire $w_j$ with its value $v$. Alternatively if $n^2 \leq |\mathbb{F}|$ a pair of values $(k, v)$ could be sent - where $k = j * n + i$.

**Lemma 2.** *The above cases correctly identify faulty wires of the first phase.*

*Proof.* Case 1: In Case 1, the error value received by the sender at the end of the first phase is identified to be different to the value sent by the receiver at the start of the phase. The only way this could have occurred is if the wire upon which the share was transmitted was actively controlled and the share was altered. The specific wire is thus correctly identified as a faulty wire.

Case 2: In Case 2, the value of the error received by the sender at the end of the first phase is identified to be the same as that sent by the receiver. The only way that a correct value of a share could be identified as an error by the sender is if it does not correspond to the corresponding polynomial. The only way this could occur is if the polynomial had been altered from its original form. The wire upon which the specific polynomial was sent is thus identified as a faulty wire. Following on from this we also prove the following lemma. □

**Lemma 3.** *If the adversary alters the polynomial transmitted across an adversary controlled wire, the specific wire will always be identified as faulty.*

*Proof.* As all polynomials are of degree at most $t$ and as shares sent on honest wires cannot be altered, if the adversary alters a polynomial the maximum number of shares transmitted on honest wires that can be included on the altered polynomial is $t$. [5] This is a direct result from coding theory which states that polynomials of degree at most $t$ can share at most $t$ common points between them. As a result of this, there will always be at least one share transmitted on an honest wire which will be identified as an error by the sender at the end of the first phase. Following on from this, the adversary controlled wire will be identified as earlier described. □

### 5.3 Formal Protocol Description

We now formally present our protocol. For our protocol we assume the message of the transmission is $M^A \in \mathbb{F}$.

**Step 1:** The receiver does the following for $i, j := 1, \ldots, n$:

1. The receiver selects random element $r_i$.
2. The receiver constructs a $(t+1)$-out-of-$n$ secret sharing scheme of $r_i$ using the random polynomial $p_i$ of degree at most $t$ to obtain $n$ shares $(s_{1i}, s_{2i}, \ldots, s_{ni})$.
3. The receiver sends polynomial $p_i$ on wire $w_i$ and share $s_{ij}$ is sent on wire $w_j$.

**Step 2:** The sender does the following

1. The sender constructs a $(t+1)$-out-of-$n$ secret sharing scheme of $M^A$ to obtain $n$ shares $(m_1, m_2, \ldots, m_n)$.

[5] The adversary can trivially carry out this alteration as the adversary knows the polynomial definition and thus all the shares.

2. For $i := 1, \ldots, n$ the sender receives polynomial $p_i$ from wire $w_i$. The sender evaluates $p_i(0)$ as $r_i$. The sender calculates the value $d_i := r_i \oplus m_i$. These are termed correcting information.
3. For $i := 1, \ldots, n$ using the $i^{th}$ shares received from each wire, error shares are identified. $s_{ij}$ received from wire $w_j$ is an error share if $s_{ij} \neq p_i(x_j)$.
4. The set of all identified error shares is sent to the receiver via broadcast.
5. The set of correcting information - $(d_1, d_2, \ldots, d_n)$, is sent to the receiver via broadcast.

**Step 3:** The receiver does the following:

1. The receiver uses the technique of Section 5.2 to identify the set of active wires of the first phase. The set of honest wires (indicated as $HONEST$) is also constructed.
2. Using $HONEST$ the receiver computes shares of the secret message $M^A$. This is done by computing $m_{w_i} := r_{w_i} \oplus d_{w_i}$ where $w_i \in HONEST$.
3. Using the computed shares from the step above, the receiver interpolates and obtains the secret message.

### 5.4 Security and Efficiency

**Theorem 2.** *The above protocol achieves perfectly secure message transmission* $((0, 0, 0)$*-security).*

*Proof.* We first prove the perfect privacy of the protocol. As the secret message is secret shared using a $(t+1)$-out-of-$n$ secret sharing scheme and as the adversary is $t$-bounded, the adversary can only learn a maximum of $t$ shares. This is because only $t$ of the random elements received by the sender in Step 2 are learned by the adversary. These are the random elements whose polynomial definitions were transmitted on adversary controlled wires (these may have been altered by the adversary). The remaining $t + 1$ random elements are not learned by the adversary. This is because all random elements are secret shared using a $(t + 1)$-out-of-$n$ secret sharing scheme and the adversary only learns $t$ shares of each one. As a result, the adversary can only learn $t$ shares of the secret shared message. Perfect privacy is therefore achieved.

Perfect authenticity of the protocol is achieved as the receiver only considers shares of the secret message whose corresponding random element was correctly received by the sender. This is achieved using the technique of identifying faulty wires described in Section 5.2. As shown, if the adversary alters the polynomial transmitted on an adversary controlled wire, the wire will always be identified as a faulty wire. Because of this, only correct shares are used in the reconstruction of the secret and thus perfect authenticity is achieved. Perfect availability is achieved as the receiver always accepts a message. The protocol is thus a perfectly secure message transmission protocol. □

We now analyze the complexity of the protocol. We denote as $|\mathbb{F}|$ the bit length of the field elements, $COM(1)$ and $COM(2)$ the communication complexity of the first and second phase of the protocol.

As the receiver in the first phase of the protocol sends $n$ shares and a polynomial (defined by $t+1$ field elements) across each wire the communication complexity of $COM(1)$ is $O(n^2)$.

The most expensive part of phase two in terms of communication complexity is the broadcast of the error shares identified in Step 2 of the protocol. As there are only $t+1$ honest wires, the minimum number of shares not identified as error shares by the sender will always be $(t+1)^2$. The maximum number of error shares is $n^2-(t+1)^2$. This is $O(n^2)$. Therefore, for the broadcast of the error shares $O(n^3)$ communication complexity is required. The communication complexity of $COM(2)$ is thus $O(n^3)$. As only one message is sent, the transmission rate of the protocol is $O(n^3)$.

It is easy to see that the computational costs of both sender and receiver are both polynomial.

### 5.5 2-Phase PSMT with $O(n^2)$ Transmission Rate

The protocol of Section 5.3 in its current form achieves $O(n^3)$ transmission rate. This is because of the $O(n^3)$ communication complexity of $COM(2)$ for the transmission of only one secret. We now describe how to decrease the transmission rate of the protocol to $O(n^2)$. The most expensive step in our two-phase protocol is the broadcast of the error shares by the sender to the receiver. The protocol is optimized by using a technique which allows for the reliable transmission of the error shares to take place with $O(n^2)$ communication complexity (as opposed to its current $O(n^3)$).

The technique of generalized broadcast was first presented in [14] and later used in [1,7]. The technique assumes the receiver knows the location of a number $f$ of faulty wires. Generalized broadcast is then able to authentically transmit up to $(f+1)$ field elements between a sender and a receiver using codes that can correct any (remaining) errors that may occur. This enables the authentic transmission of $(f+1)$ field elements between a sender and a receiver with a communication complexity of $O(n)$ instead of $O(n^2)$ which allows for more efficient protocols. For a definition of generalized broadcast the reader is referred to Appendix A or [1,7,14].

As everything else in the protocol remains the same, the security proof of this version of the protocol remains the same as before. In the second phase of the protocol the sender uses generalized broadcast to transmit the error shares and broadcasts the $n$ correcting information to allow for the secret message recovery on the receiver side. The second phase communication complexity - similar to the first phase, is now $O(n^2)$ and as only one message is transmitted so is the transmission rate of the protocol.

To the best of our knowledge this version of our protocol is the most efficient 2-phase polynomial perfectly secure transmission protocol for the transmission of a single message which exists in the literature. Previous efficient protocols included [10] and the protocol of Section 4 of [7].

## 6 Conclusion

In this paper we have introduced and formalized new security parameters to message transmission protocols.

We have also presented a polynomial protocol achieving almost secure message transmission for a single message. The protocol is of polynomial time and has $O(n^2)$ communication complexity. It remains an open question whether a polynomial time protocol with the lower bound of $O(n)$ communication complexity (as proven in [8]) can be created.

We have also presented a polynomial 2-phase perfectly secure message transmission protocol with $O(n^2)$ communication complexity for a single message. It would be nice to see if a more efficient protocol with lower communication complexity (for either one or both phases) could be achieved. Following on from this, it is also an open question to find a polynomial time protocol with linear transmission rate and a communication complexity lower than $O(n^3)$.

**Acknowledgements.** The authors would like to thank the anonymous referees for their valuable comments on improving the presentation of this paper.

## References

1. Agarwal, S., Cramer, R., de Haan, R.: Asymptotically optimal two-round perfectly secure message transmission. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 394–408. Springer, Heidelberg (2006)
2. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly secure message transmission. Journal of the ACM 40(1), 17–47 (1993)
3. Franklin, M., Galil, Z., Yung, M.: Eavesdropping games: A graph-theoretic approach to privacy in distributed systems. Journal of the ACM 47(2), 225–243 (2000)
4. Franklin, M.K., Wright, R.N.: Secure communication in minimal connectivity models. Journal of Cryptology 13(1), 9–30 (2000)
5. Jaggi, S., Langberg, M., Katti, S., Ho, T., Katabi, D., Medard, M.: Resilient network coding in the presence of Byzantine adversaries. In: INFOCOM, pp. 616–624. IEEE, Los Alamitos (2007)
6. Kumar, M.V.N.A., Goundan, P.R., Srinathan, K., Rangan, C.P.: On perfectly secure communication over arbitrary networks. In: PODC 2002, pp. 193–202 (2002)
7. Kurosawa, K., Suzuki, K.: Truly efficient 2-round perfectly secure message transmission scheme. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 324–340. Springer, Heidelberg (2008)
8. Kurosawa, K., Suzuki, K.: Almost secure (1-round, n-channel) message transmission scheme. In: Desmedt, Y. (ed.) ICITS 2007. LNCS, vol. 4883, pp. 99–112. Springer, Heidelberg (2009)
9. Patra, A., Shankar, B., Choudhary, A., Srinathan, K., Rangan, C.P.: Perfectly secure message transmission in directed networks tolerating threshold and non threshold adversary. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 80–101. Springer, Heidelberg (2007)
10. Sayeed, H.M., Abu-Amara, H.: Efficient perfectly secure message transmission in synchronous networks. Inf. Comput. 126(1), 53–61 (1996)
11. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)

12. Srinathan, K., Choudhary, A., Patra, A., Rangan, C.P.: Efficient single phase unconditionally secure message transmission with optimum communication complexity. In: PODC 2008, p. 457 (2008)
13. Srinathan, K., Kumar, M.V.N.A., Rangan, C.P.: Asynchronous secure communication tolerating mixed adversaries. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 224–242. Springer, Heidelberg (2002)
14. Srinathan, K., Narayanan, A., Rangan, C.P.: Optimal perfectly secure message transmission. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 545–561. Springer, Heidelberg (2004)
15. Wang, Y., Desmedt, Y.: Perfectly Secure Message Transmission Revisited. In: Knudsen, L. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 502–517. Springer, Heidelberg (2002)

# Appendix

## A Generalized Broadcast

In this section we describe generalized broadcast and its use in the protocol of Section 5.5. Generalized broadcast is a technique which combines broadcast and error correcting codes thus achieving a more efficient broadcast of the error shares. As mentioned in Section 5.5 generalized broadcast is used to decrease the communication complexity of phase two of the protocol from $O(n^3)$ to $O(n^2)$.

As generalized broadcast only concerns the second phase of the protocol, we assume that phase one has completed and the sender has identified all error shares that may have occurred. We now describe the further steps the sender carries out in order to transmit the error shares more efficiently.

The sender first defines the undirected graph $G_e$. An edge of the graph represents an error share that has been identified by the sender. The two vertices of an edge are the two wires involved with the error share - the wire from which the share was received and the wire whose polynomial definition the share is meant to correspond to. What is important to note here is that each edge of $G_e$ always involves at least one faulty wire. This is because two honest wires can never cause an error share as no alterations occur on honest wires.

**Definition 2.** *A matching $M$ of a graph $G = (V, E)$ is a set of pairwise non-adjacent edges. This means that no two edges in $M$ share a common vertex. A maximum matching of a graph $G$ is a matching that contains the largest possible number of edges.*

The sender proceeds to compute a maximum matching $M_{G_e}$ of $G_e$. Denoting as $M_s$ the size of $M_{G_e}$ this indicates that there are $M_s$ number of edges in $M_{G_e}$. It should be noted that $M_s \leq t$. Because of this, the sender is able to broadcast the maximum matching $M_{G_e}$ of $G_e$ to the receiver with $O(n^2)$ communication complexity. For each edge of $M_{G_e}$ the sender will broadcast the received value $es_{ij}$ of the error share, the wire $w_r$ from which the share was received and the wire $w_a$ with which it is associated with. As every edge in $G_e$ (and thus in $M_{G_e}$) always involves at least one faulty wire, this allows the receiver to identify $M_s$

number of faulty wires (in the same way as described in Section 5.2). What is important for the encoding and transmission of all the error shares is that the sender is aware of this.

Suppose that the sender wants to send $M_s$ number of elements $(e_1, \ldots, e_{M_s})$ to the receiver. The sender finds a polynomial $p$ of degree at most $M_s$ such that $p(1) = e_1, \ldots, p(M_s) = e_{M_s}$. The sender computes $p(M_s + i)$ and transmits the value on wire $w_i$ where $1 \leq i \leq n$. The receiver in turn will receive $n$ shares of an $(M_s + 1)$-out-of-$n$ secret sharing scheme. This kind of code has a minimum Hamming distance of $n - M_s = 2t + 1 - M_s$. As $M_s \leq t$ this code does not allow the receiver to correct the maximum number of errors that may occur. However, as the receiver knows $M_s$ number of faulty wires - through the broadcast of $M_{G_e}$, the receiver can ignore all shares received from these wires. The shortened code the receiver now considers is an $(M_s + 1)$-out-of-$(n - M_s)$ secret sharing scheme with a minimum Hamming distance of $n - M_s - M_s = 2t + 1 - M_s - M_s = 2(t - M_s) + 1$. This kind of code allows the receiver to correct $t - M_s$ number of errors which also equates to the number of faulty wires that have yet to be identified. The use of this shortened code thus allows the receiver to correct all remaining errors that may occur, reconstruct the same polynomial $p$ and with perfect authenticity obtain the $M_s$ elements $(e_1, \ldots, e_{M_s})$. With the transmission of $n$ elements (one share per wire) the sender is thus able to communicate with perfect authenticity $M_s$ elements to the receiver.

Following on from the above, as the size of the maximum matching is $M_s$ this means that $2M_s$ vertices (which correspond to wires) appear in $M_{G_e}$ and $G_e$. As at most $n$ error shares can be associated with each wire the number of error shares that will need to be transmitted to the receiver are at most $2M_s n$. As shown, $M_s$ elements can be transmitted using $n$ elements and as there are at most $2M_s n$ error shares to transmit this can be carried out in $2n$ independent executions of the generalized broadcast method.

All of the error shares can thus be communicated from sender to received with perfect authenticity with $O(n^2)$ communication complexity. Because of this, the communication complexity of the second phase of the protocol is now reduced from $O(n^3)$ to $O(n^2)$.

# Communication Efficient Perfectly Secure VSS and MPC in Asynchronous Networks with Optimal Resilience

Arpita Patra*, Ashish Choudhury**, and C. Pandu Rangan***

Dept of Computer Science and Engineering
IIT Madras, Chennai, India 600036
{arpitapatra10,prangan55}@gmail.com, partho_31@yahoo.co.in

**Abstract.** Verifiable Secret Sharing (VSS) is a fundamental primitive used in many distributed cryptographic tasks, such as Multiparty Computation (MPC) and Byzantine Agreement (BA). It is a two phase (sharing, reconstruction) protocol. The VSS and MPC protocols are carried out among $n$ parties, where $t$ out of $n$ parties can be under the influence of a *Byzantine (active)* adversary, having *unbounded computing power*. It is well known that protocols for perfectly secure VSS and perfectly secure MPC exist in an asynchronous network iff $n \geq 4t+1$. Hence, we call any perfectly secure VSS (MPC) protocol designed over an asynchronous network with $n = 4t + 1$ as *optimally resilient* VSS (MPC) protocol.

A secret is $d$-shared among the parties if there exists a random degree-$d$ polynomial whose constant term is the secret and each honest party possesses a distinct point on the degree-$d$ polynomial. Typically VSS is used as a primary tool to generate $t$-sharing of secret(s). In this paper, we present an *optimally resilient, perfectly secure* Asynchronous VSS (AVSS) protocol that can generate $d$-sharing of a secret for any $d$, where $t \leq d \leq 2t$. This is the first *optimally resilient, perfectly secure* AVSS of its kind in the literature. Specifically, our AVSS can generate $d$-sharing of $\ell \geq 1$ secrets from $\mathbb{F}$ concurrently, with a communication cost of $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits, where $\mathbb{F}$ is a finite field. Communication complexity wise, the best known optimally resilient, perfectly secure AVSS is reported in [2]. The protocol of [2] can generate $t$-sharing of $\ell$ secrets concurrently, with the same communication complexity as our AVSS. However, the AVSS of [2] and [4] (the only known optimally resilient perfectly secure AVSS, other than [2]) does not generate $d$-sharing, for any $d > t$.

Interpreting in a different way, we may also say that our AVSS shares $\ell(d + 1 - t)$ secrets simultaneously with a communication cost of $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits. Putting $d = 2t$ (the maximum value of $d$), we notice that the amortized cost of sharing a single secret using our AVSS is only $\mathcal{O}(n \log |\mathbb{F}|)$ bits. This is a clear improvement over the AVSS of [2] whose amortized cost of sharing a single secret is $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.

* Financial Support from Microsoft Research India Acknowledged.
** Financial Support from Infosys Technology India Acknowledged.
*** Work Supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation Sponsored by Department of Information Technology, Government of India.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 184–202, 2010.

As an interesting application of our AVSS, we propose a new *optimally resilient, perfectly secure Asynchronous Multiparty Computation* (AMPC) protocol that communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits *per multiplication gate*. The best known *optimally resilient perfectly secure* AMPC is due to [2], which communicates $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits per multiplication gate. Thus our AMPC improves the communication complexity of the best known AMPC of [2] by a factor of $\Omega(n)$.

**Keywords:** Verifiable Secret Sharing, Multiparty Computation.

## 1 Introduction

VSS or MPC protocol is carried out among a set of $n$ parties, say $\mathcal{P} = \{P_1, \ldots, P_n\}$, where every two parties are directly connected by a secure channel and $t$ parties can be under the influence of a *computationally unbounded Byzantine (active) adversary* $\mathcal{A}_t$. The adversary $\mathcal{A}_t$ completely dictates the parties under its control and can force them to deviate from a protocol, in any arbitrary manner.

**VSS:** Any VSS scheme consists of two phases: (i) a *sharing phase* in which a special party in $\mathcal{P}$, called *dealer* (denoted as $D$), on having a secret $s \in \mathbb{F}$ (an element from a finite field $\mathbb{F}$), shares it among all the parties; (ii) a *reconstruction phase*, in which the parties reconstruct the secret from their shares. Informally, the goal of any VSS scheme is to allow $D$ to share his secret $s$ during the sharing phase, among the parties in $\mathcal{P}$ in such a way that the shares would later allow for a unique reconstruction of $s$ in the reconstruction phase. Moreover, if $D$ is *honest*, then the secrecy of $s$ from $\mathcal{A}_t$ should be preserved until the reconstruction phase. VSS is one of the fundamental building blocks for many secure distributed computing tasks, such as MPC, Byzantine Agreement (BA), etc. VSS has been studied extensively over the past three decades in different settings and computational models (see [5,23,15,14,16,17] and their references).

**MPC:** MPC [25,10,5,23] allows the parties in $\mathcal{P}$ to securely compute an agreed function $f$, even in the presence of $\mathcal{A}_t$. More specifically, assume that the agreed function $f$ can be expressed as $f : \mathbb{F}^n \rightarrow \mathbb{F}^n$ and party $P_i$ has input $x_i \in \mathbb{F}$, where $\mathbb{F}$ is a finite field and $|\mathbb{F}| \geq n$. At the end of the computation of $f$, each honest $P_i$ gets $y_i \in \mathbb{F}$, where $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$, irrespective of the behavior of $\mathcal{A}_t$ (**Correctness**). Moreover, $\mathcal{A}_t$ should not get any information about the input and output of the honest parties, other than what can be inferred from the input and output of the corrupted parties (**Secrecy**). In any general MPC protocol, the function $f$ is specified by an arithmetic circuit over $\mathbb{F}$, consisting of input, linear (e.g. addition), multiplication, random and output gates. We denote the number of gates of these types in the circuit by $c_I, c_A, c_M, c_R$ and $c_O$ respectively. *Among all the different type of gates, evaluation of a multiplication gate requires the most communication complexity. So the communication complexity of any general MPC is usually given in terms of the communication complexity per multiplication gate* [3,2,1,20].

The VSS and MPC problem has been studied extensively over synchronous network, which assumes that there is a global clock and the delay of any message in the network channel is bounded. However, VSS and MPC in asynchronous network has got comparatively less attention, due to its inherent hardness. As asynchronous networks model the real life networks like Internet more appropriately than synchronous networks, the fundamental problems like VSS and MPC are worthy of deep investigation over asynchronous networks.

### 1.1 Definitions

**Asynchronous Networks:** In an asynchronous network, the communication channels have arbitrary, yet finite delay (i.e the messages are guaranteed to reach eventually). To model this, $\mathcal{A}_t$ is given the power to schedule delivery of *all* messages in the network. However, $\mathcal{A}_t$ can *only schedule* the messages communicated between honest parties, without having any access to them. Here the inherent difficulty in designing a protocol comes from the fact that when a party does not receive an expected message then he cannot decide whether the sender is corrupted (and did not send the message at all) or the message is just delayed. So a party can not wait to consider the values sent by all parties, as waiting for them could turn out to be endless. Hence the values of up to $t$ (potentially honest) parties may have to be ignored. Due to this the protocols in asynchronous network are generally involved in nature and require new set of primitives. For comprehensive introduction to asynchronous protocols, see [8].

**Asynchronous Verifiable Secret Sharing (AVSS) [4,8]:** Let (Sh, Rec) be a pair of protocols in which a dealer $D \in \mathcal{P}$ shares a secret $s$ from a finite field $\mathbb{F}$ using Sh. We say that (Sh, Rec) is a $t$-resilient *perfectly secure* AVSS scheme with $n$ parties if the following hold for every possible $\mathcal{A}_t$:

- **Termination:** (1) If $D$ is honest then each honest party will eventually terminate protocol Sh. (2) If some honest party has terminated protocol Sh, then irrespective of the behavior of $D$, each honest party will eventually terminate Sh. (3) If all the honest parties have terminated Sh and all the honest parties invoke protocol Rec, then each honest party will eventually terminate Rec.

- **Correctness:** (1) If $D$ is *honest* then each honest party upon terminating protocol Rec, outputs the shared secret $s$. (2) If $D$ is *faulty* and some honest party has terminated Sh, then there exists a fixed $\overline{s} \in \mathbb{F}$, such that each honest party upon completing Rec, will output $\overline{s}$.

- **Secrecy:** If $D$ is honest and no honest party has begun Rec, then $\mathcal{A}_t$ has no information about $s$.

The above definition of AVSS can be extended for secret $S$ containing multiple elements (say $\ell$ with $\ell > 1$) from $\mathbb{F}$.

**Asynchronous Multi Party Computation (AMPC) [8]:** A *perfectly secure* AMPC should satisfy the **Correctness** and **Secrecy** property of MPC. In addition, it should also satisfy **Termination** property, according to which, every honest party should eventually terminate the protocol.

**$d$-sharing and $(t, 2t)$-sharing [1,3]:** A value $s \in \mathbb{F}$ is $d$-shared among a set of parties $\overline{\mathcal{P}} \subseteq \mathcal{P}$ with $|\overline{\mathcal{P}}| \geq d + 1$ if there exists a degree-$d$ polynomial $f(x)$ with $f(0) = s$ such that each honest $P_i \in \overline{\mathcal{P}}$ holds $s_i = f(i)$. The vector of shares of $s$ belonging to the honest parties is called $d$-sharing of $s$ and is denoted by $[s]_d$. A value $s$ is said to be $(t, 2t)$-shared among the parties in $\mathcal{P}$, denoted as $[s]_{(t,2t)}$, if $s$ is simultaneously $t$-shared, as well as $2t$-shared among the parties in $\mathcal{P}$.

**A-cast [9,8]:** It is an asynchronous broadcast primitive, which allows a special party in $\mathcal{P}$ (called the sender) to distribute a message identically among the parties in $\mathcal{P}$. If sender is honest, then every honest party eventually terminates A-cast with the sender's message. For a corrupted sender, if some honest party terminates with some message, then every other honest party will eventually terminate with **same** message. A-cast is elegantly implemented in [7] with $n = 3t+1$, which incurs a private communication of $\mathcal{O}(n^2 b)$ bits, for a $b$-bit message.

**Agreement on Common Subset (ACS) [2,6]:** It is an asynchronous primitive presented in [4,6]. It is used to determine and output a common set, containing at least $n-t$ parties, who correctly shared their values. Each honest party will eventually get a share, corresponding to each value, shared by the parties in the common set. ACS requires private communication of $\mathcal{O}(\text{poly}(n) \log |\mathbb{F}|)$ bits.

### 1.2 Our Contributions and Comparison with Existing Results

**Our Contribution:** From [4,8], perfectly secure AVSS and AMPC is possible iff $n \geq 4t + 1$. Hence, we call any *perfectly secure* AVSS (AMPC) protocol with $n = 4t + 1$ as *optimally resilient, perfectly secure* AVSS (AMPC) protocol. Typically, AVSS is used as a tool for generating $t$-sharing of secrets. In this paper, we present a novel *optimally resilient, perfectly secure* AVSS protocol that can generate $d$-sharing of $\ell$ secrets concurrently for any $d$, where $t \leq d \leq 2t$, with a private communication of $\mathcal{O}(\ell n^2 \log(|\mathbb{F}|))$ bits and A-cast of $\mathcal{O}(n^2 \log(|\mathbb{F}|))$ bits. Interpreting in a different way, we may also say that the amortized cost of sharing a single secret using our AVSS is only $\mathcal{O}(n \log |\mathbb{F}|)$ bits (this will be discussed in detail after the presentation of AVSS protocol in section 4).

To design our AVSS, we exploit several interesting properties of $(n, t)$-star (a graph theoretic concept presented in section 4.4.2 of [8]) in conjunction with some properties of bivariate polynomial with *different* degree in variable $x$ and $y$. The $(n, t)$-star was used to design a perfectly secure optimally resilient AVSS protocol in [8] (the details of $(n, t)$-star is presented in Section 2 of current article). While the properties of $(n, t)$-star that our AVSS explores were not required in the AVSS of [8] (which generates only $t$-sharing of secrets), our AVSS uses them for the first time for generating $d$-sharing of secrets, where $t \leq d \leq 2t$.

As an interesting application of our AVSS, we design a new *optimally resilient, perfectly secure* AMPC protocol that communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits *per multiplication gate*. This solves an open problem posed in [20]. Using our AVSS, we

first design an efficient protocol that generates $(t, 2t)$-sharing of a secret with a communication cost of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. Then following the approach of [11,2], given $(t, 2t)$-sharing of a secret, a multiplication gate is evaluated by communicating $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. This is how our AMPC attains quadratic communication complexity per multiplication gate.

**Comparison of Our AVSS with Existing AVSS Protocols:** In Table 1, we compare our AVSS with existing optimally resilient, perfectly secure AVSS protocols. We emphasize that the *AVSS protocols of [4,2] does not generate d-sharing for any* $d > t$. When negligible error probability is allowed in **Termination** and/or **Correctness**, we arrive at the notion of *statistical* AVSS. Statistical AVSS is possible iff $n \geq 3t + 1$ [9,6]. To the best of our knowledge, the AVSS protocols of [9,6,19,18] are the only known *optimally resilient statistical* AVSS (i.e., with $n = 3t + 1$). As these protocols have comparatively high communication complexity and are designed with $n = 3t + 1$ (with which perfectly secure AVSS is impossible), we do not compare our AVSS with the statistical AVSS protocols of [9,6,19,18]. Recently in [20], a statistical AVSS with $n = 4t + 1$ (i.e., with non-optimal resilience) is reported. Clearly, our AVSS achieves stronger properties than the AVSS of [20] (*the AVSS of [20] involves a negligible error probability, while our AVSS is perfect*), though both of them generate $d$-sharing for any $t \leq d \leq 2t$, with same communication complexity. For achieving perfectness, we use techniques which are completely different from [20].

**Table 1.** Our AVSS vs with Existing Optimally Resilient Perfectly Secure AVSS

| Reference | # Secrets Shared | Type of Sharing Generated | Communication Complexity In Bits (CCIB) |
|---|---|---|---|
| [4] | 1 | Only $t$-sharing | Private–$\mathcal{O}(n^3 \log(\|\mathbb{F}\|))$; A-cast–$\mathcal{O}(n^2 \log(\|\mathbb{F}\|))$ |
| [2] | $\ell \geq 1$ | Only $t$-sharing | Private– $\mathcal{O}(\ell n^2 \log(\|\mathbb{F}\|))$; A-cast–$\mathcal{O}(n^2 \log(\|\mathbb{F}\|))$ |
| This article | $\ell \geq 1$ | $d$-sharing, for any $t \leq d \leq 2t$ | Private– $\mathcal{O}(\ell n^2 \log(\|\mathbb{F}\|))$; A-cast–$\mathcal{O}(n^2 \log(\|\mathbb{F}\|))$ |

**Comparison of Our AMPC with Existing AMPC Protocols:** In Table 2, we compare our AMPC protocol with the existing optimally resilient, perfectly secure AMPC protocols in terms of communication complexity. We observe that our AMPC gains by a factor of $\Omega(n)$ as compared to the best known perfectly secure AMPC of [2]. If negligible error probability is allowed in **Termination** and/or **Correctness**, we arrive at the notion of *statistical* AMPC. Statistical AMPC is possible iff $n \geq 3t + 1$ [6]. Optimally resilient statistical AMPC protocols (i.e., with $n = 3t + 1$) are reported in [6,18]. As these

**Table 2.** Comparison of Our AMPC with Existing Optimally Resilient Perfectly Secure AMPC Protocols

| Reference | CCIB / Multiplication Gate |
|---|---|
| [4,8] | $\mathcal{O}(n^6 \log(\|\mathbb{F}\|))$ |
| [2] | $\mathcal{O}(n^3 \log(\|\mathbb{F}\|))$ |
| This Article | $\mathcal{O}(n^2 \log(\|\mathbb{F}\|))$ |

protocols have very high communication complexity and are designed with $n = 3t+1$ (with which perfectly secure AMPC is impossible), we do not compare them with our AMPC protocol. Statistical AMPC with $n = 4t+1$ (i.e., with non-optimal resilience) are reported in [24,22,20]. Among them, the AMPC reported in [20] is the best, which communicates $\mathcal{O}(n^2 \log(|\mathbb{F}|))$ bits per multiplication gate. Though the protocol of [20] attains the same communication complexity as our AMPC protocol (per multiplication gate), *our protocol is perfect in all respects, while the protocol of [20] has error probability in both Termination and Correctness.*

## 2 Finding $(n, t)$-star Structure in a Graph

We now describe an existing solution for a graph theoretic problem, called finding $(n, t)$-star in an undirected graph $G = (V, E)$. Our AVSS protocol exploits several interesting properties of $(n, t)$-star.

**Definition 1 ($(n, t)$-star [8,4]:).** *Let $G$ be an undirected graph with the $n$ parties in $\mathcal{P}$ as its vertex set. We say that a pair $(\mathcal{C}, \mathcal{D})$ of sets with $\mathcal{C} \subseteq \mathcal{D} \subseteq \mathcal{P}$ is an $(n, t)$-star in $G$, if the following hold: (i) $|\mathcal{C}| \geq n - 2t$; (ii) $|\mathcal{D}| \geq n - t$; (iii) for every $P_j \in \mathcal{C}$ and every $P_k \in \mathcal{D}$ the edge $(P_j, P_k)$ exists in $G$.*

Ben-Or et. al [4] have presented an elegant and efficient algorithm for finding an $(n, t)$-star in a graph of $n$ nodes, *provided that the graph contains a clique of size $n - t$*. The algorithm, called Find-STAR outputs either an $(n, t)$-star or the message `star-Not-Found`. Whenever the input graph contains a clique of size $n - t$, Find-STAR always outputs an $(n, t)$-star in the graph.

Actually, algorithm Find-STAR takes the complementary graph $\overline{G}$ of $G$ as input and tries to find $(n, t)$-$\overline{\text{star}}$ in $\overline{G}$ where $(n, t)$-$\overline{\text{star}}$ is a pair $(\mathcal{C}, \mathcal{D})$ of sets with $\mathcal{C} \subseteq \mathcal{D} \subseteq \mathcal{P}$, satisfying the following conditions: (i) $|\mathcal{C}| \geq n - 2t$; (ii) $|\mathcal{D}| \geq n - t$; (iii) there are no edges between the nodes in $\mathcal{C}$ and nodes in $\mathcal{C} \cup \mathcal{D}$ in $\overline{G}$. Clearly, a pair $(\mathcal{C}, \mathcal{D})$ representing an $(n, t)$-$\overline{\text{star}}$ in $\overline{G}$, is an $(n, t)$-star in $G$. Recasting the task of Find-STAR in terms of complementary graph $\overline{G}$, we say that Find-STAR outputs either a $(n, t)$-$\overline{\text{star}}$, or a message `star-Not-Found`. Whenever, *the input graph $\overline{G}$ contains an independent set of size $n - t$*, Find-STAR always outputs an $(n, t)$-$\overline{\text{star}}$. For simple notation, we denote $\overline{G}$ by $H$. The algorithm Find-STAR is presented in the following table. For properties of Find-STAR, please see [8].

**Algorithm Find-STAR($H$)**

1. Find a maximum matching $M$ in $H$. Let $N$ be the set of matched nodes (namely, the endpoints of the edges in $M$), and let $\overline{N} = \mathcal{P} \setminus N$.
2. Compute output as follows (which could be either $(n, t)$-$\overline{\text{star}}$ or a message `star-Not-Found`):
   (a) Let $T = \{P_i \in \overline{N} | \exists P_j, P_k$ s.t $(P_j, P_k) \in M$ and $(P_i, P_j), (P_i, P_k) \in E\}$. $T$ is called the set of triangle-heads.
   (b) Let $\mathcal{C} = \overline{N} \setminus T$.
   (c) Let $B$ be the set of matched nodes that have neighbors in $\mathcal{C}$. So $B = \{P_j \in N | \exists P_i \in \mathcal{C}$ s. t. $(P_i, P_j) \in E\}$.
   (d) Let $\mathcal{D} = \mathcal{P} \setminus B$. If $|\mathcal{C}| \geq n - 2t$ and $|\mathcal{D}| \geq n - t$, output $(\mathcal{C}, \mathcal{D})$. Otherwise, output `star-Not-Found`.

## 3 AVSS for Generating $d$-Sharing of a Single Secret

We now present a novel AVSS protocol consisting of two sub-protocols, namely AVSS-Share-SS and AVSS-Rec-SS. The AVSS-Share-SS allows a dealer $D \in \mathcal{P}$ (dealer can be any party from $\mathcal{P}$) to $d$-share a single secret from $\mathbb{F}$, among the parties in $\mathcal{P}$, where $t \leq d \leq 2t$. Protocol AVSS-Rec-SS allows the parties in $\mathcal{P}$ to reconstruct the secret, given its $d$-sharing. The structure of AVSS-Share-SS is divided into a sequence of following three phases.

1. **Distribution Phase:** As the name suggests, in this phase, $D$ on having a secret $s$, distributes information to the parties in $\mathcal{P}$.
2. **Verification & Agreement on CORE Phase:** Here parties jointly perform some computation and communication in order to verify consistency of the information distributed by $D$ in **Distribution Phase**. In case of successful verification, all honest parties agree on a set of at least $3t+1$ parties called $CORE$, satisfying certain property (mentioned in the sequel).
3. **Generation of $d$-sharing Phase:** If $CORE$ is agreed upon in previous phase, then here every party performs local computation on the data received (during **Verification & Agreement on CORE Phase**) from the parties in $CORE$ to finally generate the $d$-sharing of secret $s$.

An honest party will terminate AVSS-Share-SS, if it successfully completes the last phase, namely **Generation of $d$-sharing Phase**. If $D$ is honest then each honest party will eventually terminate the last phase. Moreover, if $D$ is corrupted and some honest party terminates the last phase, then each honest party will also eventually terminate the last phase (and hence AVSS-Share-SS).

*Remark 1.* The sharing phase of statistical AVSS protocol of [20] is also structured into above three phases. However, our implementation of **Verification & Agreement on CORE Phase** is completely different from [20]. More importantly, the last two phases in [20] involves a negligible error probability, whereas our implementation of all the three phases are perfect (error free) in all respects.

### 3.1 Distribution Phase

Here $D$ on having a secret $s$, selects a random bivariate polynomial $F(x,y)$ of degree-$(d,t)$ (i.e., the degree of the polynomial in $x$ is $d$ and the degree of the polynomial in $y$ is $t$), such that $F(0,0) = s$ and sends $f_i(x) = F(x,i)$ and $p_i(y) = F(i,y)$ to party $P_i$. As in [20], we will call the degree-$d$ $f_i(x)$ polynomials as *row polynomials* and degree-$t$ $p_i(y)$ polynomials as *column polynomials*.

**Protocol Distribution-SS($D, \mathcal{P}, s, d$)**

Code for $D$:

1. Select a random bivariate polynomial $F(x,y)$ of degree-$(d,t)$ over $\mathbb{F}$, such that $F(0,0) = s$. Send $f_i(x) = F(x,i)$ and $p_i(y) = F(i,y)$ to party $P_i$, for $i = 1, \ldots, n$.

### 3.2 Verification and Agreement on CORE Phase

The goal of this phase is to check the existence of a set of parties called $CORE$. If a $CORE$ exists then every honest party will agree on $CORE$, where $CORE$ is defined as follows

**Definition 2 (Property of $CORE$).** *$CORE$ is a set of at least $3t+1$ parties such that the row polynomials (received in* **Distribution Phase***) of the honest parties in $CORE$ define a unique bivariate polynomial say, $\overline{F}(x,y)$ of degree-$(d,t)$. Moreover, if $D$ is honest, then $\overline{F}(x,y) = F(x,y)$, where $F(x,y)$ was chosen by $D$ in* **Distribution Phase***.*

The property of $CORE$ ensures that for every $j \in \{1,\ldots,n\}$, the $j^{th}$ point on row polynomials of honest parties in $CORE$ define degree-$t$ column polynomial $\overline{p_j}(y) = \overline{F}(j,y)$. So once $CORE$ is constructed and agreed upon by each honest party then $\overline{p_j}(0)$ can be privately reconstructed by $P_j$ with the help of the parties in $CORE$ by using *online error correction* (OEC) [8] [1]. This will generate $d$-sharing of $\overline{s} = \overline{F}(0,0)$, where $\overline{s}$ will be $d$-shared using degree-$d$ polynomial $\overline{f_0}(x) = \overline{F}(x,0)$ and each (honest) $P_j$ will have his share $\overline{f_0}(j) = \overline{p_j}(0)$ of $\overline{s}$. Moreover, if $D$ is honest, then $\overline{s} = s$ as $\overline{F}(x,y) = F(x,y)$. Note that even though the degree of row polynomials is more than $t$ (if $d > t$), we create a situation where parties need not have to reconstruct them. To obtain the shares corresponding to $d$-sharing of $s$, the parties need to reconstruct degree-$t$ column polynomials only. We now give an outline of this phase.

**Outline of Current Phase:** Here the parties upon receiving row and column polynomials (from $D$), interact with each other to check the consistency of their common values (on their polynomials). After successfully verifying the consistency, parties A-cast OK signals. Using these signals, a graph with the parties as vertex set is formed and applying Find-STAR on the graph, a sequence of distinct $(n,t)$-stars are obtained. The reason for constructing a sequence of $(n,t)$-stars will be clear in the sequel. Each $(n,t)$-star in such a graph defines a unique bivariate polynomial of degree-$(d,t)$.

For every generated $(n,t)$-star, $D$ tries to find whether $CORE$ can be generated from it. The generation process of $CORE$ attempts to use several interesting features of $(n,t)$-star (mainly its $\mathcal{C}$ component). We show that if $D$ is honest and $\mathcal{C}$ component of some $(n,t)$-star $(\mathcal{C},\mathcal{D})$ contains at least $2t+1$ honest parties, then $CORE$ will be eventually generated from $(\mathcal{C},\mathcal{D})$. Moreover, we also show that if $D$ is honest, then eventually some $(n,t)$-star $(\mathcal{C},\mathcal{D})$ will be generated, where $\mathcal{C}$ will contain at least $2t+1$ honest parties (though the dealer $D$ may not know which $(n,t)$-star it is). *These two important properties of $(n,t)$-star in our context are the heart our AVSS protocol.* Furthermore, we show that if $CORE$ is generated from some $(n,t)$-star $(\mathcal{C},\mathcal{D})$ (irrespective of whether $D$ is honest or corrupted), then both $CORE$, as well as $(\mathcal{C},\mathcal{D})$ define the same bivariate polynomial of degree-$(d,t)$.

[1] OEC allows to reconstruct a degree-$t$ polynomial by applying error correction in an online fashion in asynchronous settings. For details see [8].

The generation of many $(n,t)$-stars in our case is essential as the $\mathcal{C}$ component of the first $(n,t)$-star may not contain at least $2t+1$ honest parties and hence may never lead to $CORE$. *This implies that if our protocol stops after generating the first $(n,t)$-star then the protocol may not terminate even for an honest $D$.* However, we stress that existing AVSS of [4] need not generate a sequence of

**Protocol Verification-SS($D, \mathcal{P}, s, d$)**

i. CODE FOR $P_i$: Every party $P_i \in \mathcal{P}$ (including $D$) executes this code.

1. Wait to receive polynomials $\overline{f_i}(x)$ of degree-$d$ and $\overline{p_i}(y)$ of degree-$t$ from $D$. Upon receiving, send $\overline{f_{ij}} = \overline{f_i}(j)$ and $\overline{p_{ij}} = \overline{p_i}(j)$ to party $P_j$, for $j = 1, \ldots, n$.
2. Upon receiving $\overline{f_{ji}}$ and $\overline{p_{ji}}$ from $P_j$, check if $\overline{f_i}(j) \stackrel{?}{=} \overline{p_{ji}}$ and $\overline{p_i}(j) \stackrel{?}{=} \overline{f_{ji}}$. If both the equalities hold, A-cast $\mathtt{OK}(P_i, P_j)$.
3. Construct an undirected graph $G_i$ with $\mathcal{P}$ as vertex set. Add an edge $(P_j, P_k)$ in $G_i$ upon receiving (a) $\mathtt{OK}(P_k, P_j)$ from the A-cast of $P_k$ and (b) $\mathtt{OK}(P_j, P_k)$ from the A-cast of $P_j$.

ii. CODE FOR $D$: Only $D$ executes this code.

1. For every new receipt of some $\mathtt{OK}(*,*)$ update $G_D$. If a new edge is added to $G_D$, then execute Find-STAR($\overline{G_D}$). Let there are $\alpha \geq 0$ distinct $(n,t)$-stars that are found in the past from different executions of Find-STAR($\overline{G_D}$).
   (a) Now if an $(n,t)$-star is found from the current execution of Find-STAR($\overline{G_D}$) that is distinct from all the $\alpha$ $(n,t)$-stars obtained before, do the following:
      i. Call the new $(n,t)$-star as $(\mathcal{C}^{\alpha+1}, \mathcal{D}^{\alpha+1})$.
      ii. Create a list $\mathcal{F}^{\alpha+1}$ as follows: Add $P_j$ to $\mathcal{F}^{\alpha+1}$ if $P_j$ has at least $2t+1$ neighbors in $\mathcal{C}^{\alpha+1}$ in $G_D$.
      iii. Create a list $\mathcal{E}^{\alpha+1}$ as follows: Add $P_j$ to $\mathcal{E}^{\alpha+1}$ if $P_j$ has at least $d+t+1$ neighbors in $\mathcal{F}^{\alpha+1}$ in $G_D$.
      iv. For every $\gamma$, with $\gamma = 1, \ldots, \alpha$ update $\mathcal{F}^{\gamma}$ and $\mathcal{E}^{\gamma}$:
         A. Add $P_j$ to $\mathcal{F}^{\gamma}$, if $P_j \notin \mathcal{F}^{\gamma}$ and $P_j$ has at least $2t+1$ neighbors in $\mathcal{C}^{\gamma}$ in $G_D$.
         B. Add $P_j$ to $\mathcal{E}^{\gamma}$, if $P_j \notin \mathcal{E}^{\gamma}$ and $P_j$ has at least $d+t+1$ neighbors in $\mathcal{F}^{\gamma}$ in $G_D$.
   (b) If no $(n,t)$-star is found or an $(n,t)$-star that has been already found in the past is obtained, then execute step (a).iv(A-B) to update existing $\mathcal{F}^{\gamma}$'s and $\mathcal{E}^{\gamma}$'s.
   (c) Now let $\beta$ be the first index among already generated $\{(\mathcal{E}^1, \mathcal{F}^1), \ldots, (\mathcal{E}^{\delta}, \mathcal{F}^{\delta})\}$ such that both $\mathcal{E}^{\beta}$ and $\mathcal{F}^{\beta}$ contains at least $3t+1$ parties (Note that if step (a) is executed, then $\delta = \alpha + 1$; else $\delta = \alpha$). Assign $CORE = \mathcal{E}^{\beta}$ and A-cast $((\mathcal{C}^{\beta}, \mathcal{D}^{\beta}), (\mathcal{E}^{\beta}, \mathcal{F}^{\beta}))$.

iii. CODE FOR $P_i$: Every party $P_i \in \mathcal{P}$ (including $D$) executes this code.

1. Wait to receive $((\mathcal{C}^{\beta}, \mathcal{D}^{\beta}), (\mathcal{E}^{\beta}, \mathcal{F}^{\beta}))$ from the A-cast of $D$.
2. Wait until $(\mathcal{C}^{\beta}, \mathcal{D}^{\beta})$ becomes a valid $(n,t)$-star in $G_i$.
3. Wait until every party $P_j \in \mathcal{F}^{\beta}$ has at least $2t+1$ neighbors in $\mathcal{C}^{\beta}$ in $G_i$.
4. Wait until every party $P_j \in \mathcal{E}^{\beta}$ has at least $d+t+1$ neighbors in $\mathcal{F}^{\beta}$ in $G_i$.
5. Accept $CORE = \mathcal{E}^{\beta}$.

$(n,t)$-stars because it has to generate only $t$-sharing. Hence the AVSS of [4] stops after generating the first $(n,t)$-star and then using the $\mathcal{D}$ component of the generated $(n,t)$-star, it could generate $t$-sharing of $s$. Finally, once $CORE$ is obtained, it is then verified and agreed among the set of all honest parties in $\mathcal{P}$. The steps of this phase are given in protocol Verification-SS.

**Lemma 1.** *For any $(n,t)$-star $(\mathcal{C},\mathcal{D})$ in graph $G_k$ of honest $P_k$, the row polynomials held by honest parties in $\mathcal{C}$ define a unique polynomial of degree-$(d,t)$, say $\overline{F}(x,y)$, such that column polynomial $\overline{p_j}(y)$ held by every honest $P_j \in \mathcal{D}$ satisfies $\overline{p_j}(y) = \overline{F}(j,y)$. Moreover, if $D$ is honest, then $\overline{F}(x,y) = F(x,y)$.*

*Proof.* For any $(n,t)$-star $(\mathcal{C},\mathcal{D})$, $|\mathcal{C}| \geq n-2t$ and $|\mathcal{D}| \geq n-t$. So $\mathcal{C}$ and $\mathcal{D}$ contain at least $n-3t \geq t+1$ and $n-2t \geq 2t+1$ honest parties, respectively. Let $l$ and $m$ be the number of honest parties in $\mathcal{C}$ and $\mathcal{D}$ respectively where $l \geq t+1$ and $m \geq 2t+1$. Without loss of generality, we assume that $P_1,\ldots,P_l$, respectively $P_1,\ldots,P_m$ are the set of honest parties in $\mathcal{C}$ and $\mathcal{D}$. Now by the construction of $(n,t)$-star, for every pair of honest parties $(P_i,P_j)$ with $P_i \in \mathcal{C}$ and $P_j \in \mathcal{D}$, the row polynomial $\overline{f_i}(x)$ of honest $P_i$ and the column polynomial $\overline{p_j}(y)$ of honest $P_j$ satisfy $\overline{f_i}(j) = \overline{p_j}(i)$. We now prove that the above statement implies that there exists a unique bivariate polynomial $\overline{F}(x,y)$ of degree-$(d,t)$, such that for $i=1,\ldots,l$, we have $\overline{F}(x,i) = \overline{f_i}(x)$ and for $j=1,\ldots,m$, we have $\overline{F}(j,y) = \overline{p_j}(y)$. The proof is similar to the proof of Lemma 4.26 of [8]. Due to space constraints, we give the complete proof in full version of the paper [21].

**Lemma 2.** *For an honest $D$, an $(n,t)$-star $(\mathcal{C}^\beta,\mathcal{D}^\beta)$ with $\mathcal{C}^\beta$ containing at least $2t+1$ honest parties will be generated eventually.*

*Proof.* For an honest $D$, eventually the edges between each pair of honest parties will vanish from the complementary graph $\overline{G_D}$. So the edges in $\overline{G_D}$ will be either (a) between an honest and a corrupted party OR (b) between a corrupted and another corrupted party. Let $\beta$ be the first index, such that $(n,t)$-star $(\mathcal{C}^\beta,\mathcal{D}^\beta)$ is generated in $\overline{G_D}$, when $\overline{G_D}$ contains edges of above two types only. Now, by construction of $\mathcal{C}^\beta$ (see Algorithm Find-STAR), it excludes the parties in $N$ (set of parties that are endpoints of the edges of maximum matching $M$) and $T$ (set of parties that are triangle-head). An honest $P_i$ belonging to $N$ implies that $(P_i,P_j) \in M$ for some $P_j$ and hence $P_j$ is corrupted (as the current $\overline{G_D}$ does not have edge between two honest parties). Similarly, an honest party $P_i$ belonging to $T$ implies that there is some $(P_j,P_k) \in M$ such that $(P_i,P_j)$ and $(P_j,P_k)$ are edges in $\overline{G_D}$. This clearly implies that both $P_j$ and $P_k$ are surely corrupted. So for every honest $P_i$ outside $\mathcal{C}^\beta$, at least one (if $P_i$ belongs to $N$, then one; if $P_i$ belongs to $T$, then two) corrupted party also remains outside $\mathcal{C}^\beta$. As there are at most $t$ corrupted parties, $C^\beta$ may exclude at most $t$ honest parties. But still $\mathcal{C}^\beta$ is bound to contain at least $2t+1$ honest parties.

Notice that the above event happens after finite number of steps. This is because $D$ applies Find-STAR on $\overline{G_D}$ every time after an edge is added to $G_D$ and there can be $\mathcal{O}(n^2)$ edges in $G_D$. So there can be $\mathcal{O}(n^2)$ distinct $(n,t)$-stars generated by $D$. Now $(\mathcal{C}^\beta,\mathcal{D}^\beta)$ with $\mathcal{C}^\beta$ containing at least $2t+1$ honest parties will be one among these $\mathcal{O}(n^2)$ $(n,t)$-stars. □

**Lemma 3.** *In protocol Verification-SS, if $D$ is honest, then eventually $CORE$ will be generated.*

*Proof.* By Lemma 2, the honest $D$ will eventually generate an $(\mathcal{C}^\beta, \mathcal{D}^\beta)$ in $G_D$, with $\mathcal{C}^\beta$ containing at least $2t+1$ honest parties. Furthermore, if $D$ is honest then eventually there will be edges between every pair of honest parties in the graph $G_i$ of every honest $P_i$ (including $G_D$). Thus, as all honest parties in $\mathcal{P}$ will have edges with the honest parties in $\mathcal{C}^\beta$, they will be eventually added to $\mathcal{F}^\beta$. Similarly, as all honest parties in $\mathcal{P}$ will have edges with the honest parties in $\mathcal{F}^\beta$, they will be eventually added to $\mathcal{E}^\beta$. Hence $|\mathcal{E}^\beta| \geq n-t$ and $|\mathcal{F}^\beta| \geq n-t$ will be satisfied and $CORE$ will be obtained by $D$. □

**Lemma 4.** *If an honest $P_i$ has accepted $CORE$, then the row polynomials of the honest parties in $CORE$ define a unique bivariate polynomial of degree-$(d,t)$.*

*Proof.* If an honest $P_i$ has accepted $CORE$, then he has received $((\mathcal{C}^\beta, \mathcal{D}^\beta), (\mathcal{E}^\beta, \mathcal{F}^\beta))$ from the A-cast of $D$ and checked their validity with respect to his own graph $G_i$. By Lemma 1, the row polynomials of the honest parties in $\mathcal{C}^\beta$ define a unique bivariate polynomial of degree-$(d,t)$, say $\overline{F}(x,y)$. So the row polynomial held by an honest $P_i \in \mathcal{C}$ satisfies $\overline{f_i}(x) = \overline{F}(x,i)$. Now by the construction of $\mathcal{F}^\beta$, every *honest* $P_j \in \mathcal{F}^\beta$ has at least $2t+1$ neighbors in $\mathcal{C}^\beta$ which implies that $\overline{f_{kj}}$ values received from at least $2t+1$ parties in $\mathcal{C}^\beta$ lie on column polynomial $\overline{p_j}(y)$. This clearly implies $\overline{p_j}(y) = \overline{F}(j,y)$, as $t+1$ out of these $2t+1$ values are sent by honest parties in $\mathcal{C}$, who define $\overline{F}(j,y)$.

Similarly, by construction of $\mathcal{E}^\beta$, every *honest* $P_j \in \mathcal{E}^\beta$ has at least $d+t+1$ neighbors in $\mathcal{F}^\beta$ which implies that $\overline{p_{kj}}$ values received from at least $d+t+1$ parties in $\mathcal{F}^\beta$ lie on $\overline{f_j}(x)$. This implies that $\overline{f_j}(x) = \overline{F}(x,j)$, as at least $d+1$ out of these $d+t+1$ values are sent by honest parties in $\mathcal{F}^\beta$, who define $\overline{F}(x,j)$. Hence row polynomials of the honest parties in $CORE$ define $\overline{F}(x,y)$. □

### 3.3 Generation of $d$-Sharing Phase

Assuming that the honest parties in $\mathcal{P}$ have agreed upon a $CORE$, protocol d-Share-Generation-SS generates $d$-sharing in the following way: From the properties of $CORE$, the row polynomials of honest parties in $CORE$ define a unique bivariate polynomial say $\overline{F}(x,y)$ of degree-$(d,t)$, such that each honest party $P_i$ in $CORE$ possesses $\overline{f_i}(x) = \overline{F}(x,i)$. So the $j^{th}$ point on $\overline{f_i}(x)$ polynomials corresponding to all honest $P_i$'s in $CORE$, define degree-$t$ polynomial $\overline{p_j}(y) = \overline{F}(j,y)$. Furthermore, $|CORE| \geq 3t+1$. So the parties in $CORE$ can enable each $P_j \in \mathcal{P}$ to privately reconstruct $\overline{p_j}(y)$ using OEC [8]. Once this is done, every $P_j$ can output $\overline{p_j}(0)$ as the share of $D$'s committed secret. Since $\overline{f_0}(j) = \overline{p_j}(0)$, it follows that $\overline{f_0}(0)(= \overline{F}(0,0))$ will be $d$-shared using the degree-$d$ polynomial $\overline{f_0}(x) = \overline{F}(x,0)$. Clearly if $D$ is honest, $D$'s secret $s$ will be $d$-shared using polynomial $f_0(x) = F(x,0)$, as $\overline{F}(x,y) = F(x,y)$ for honest $D$. The protocol for this phase is as follows:

**Protocol d-Share-Generation-SS($D, \mathcal{P}, s, d$)**

CODE FOR $P_i$:

1. Apply On-line Error Correcting (OEC) technique [8] on $\overline{f_{ji}}$'s received from every $P_j$ in $CORE$ (during Protocol Verification-SS) and reconstruct degree-$t$ polynomial $\overline{p_i}(y)$ and output $\overline{s_i} = \overline{p_i}(0) = \overline{f_0}(i)$ as the $i^{th}$ share of $\overline{s}$ and terminate. $\overline{s}$ is now $d$-shared using polynomial $\overline{f_0}(x)$.

**Lemma 5.** *Assume that every honest party has agreed on $CORE$ where the row polynomials of the honest parties in $CORE$ define a unique bivariate polynomial of degree-$(d,t)$, say $\overline{F}(x,y)$. Then protocol d-Share-Generation-SS will generate $d$-sharing of $\overline{s} = \overline{F}(0,0)$.*

*Proof.* See [21] for complete details. □

### 3.4 Protocol AVSS-Share-SS and AVSS-Rec-SS

**Protocol AVSS-Share-SS($D, \mathcal{P}, s, d$)**

(a) $D$ executes Distribution-SS$(D, \mathcal{P}, s, d)$;
(b) Each party $P_i$ participates in Verification-SS$(D, \mathcal{P}, s, d)$;
(c) After agreeing on $CORE$, each party $P_i$ participates in d-Share-Generation-SS$(D, \mathcal{P}, s, d)$ and terminates AVSS-Share-SS after locally outputting the share corresponding to $D$'s committed secret.

**Protocol AVSS-Rec-SS($D, \mathcal{P}, s, d$)**

Party $P_i \in \mathcal{P}$ sends $s_i$, the $i^{th}$ share of $s$ to every $P_j \in \mathcal{P}$. Party $P_i$ applies OEC on received $s_j$'s, reconstructs $s$ and terminates AVSS-Rec-SS.

**Theorem 1.** *Protocols (AVSS-Share-SS, AVSS-Rec-SS) constitute a valid perfectly secure AVSS scheme for $d$-sharing a single secret from $\mathbb{F}$.*

*Proof.* **Termination:** Part (1) of **Termination** says that if $D$ is honest then every honest party will terminate AVSS-Share-SS eventually. By Lemma 3, $D$ will eventually generate $CORE$ and A-cast the corresponding information i.e. $((\mathcal{C}^\beta, \mathcal{D}^\beta), (\mathcal{E}^\beta, \mathcal{F}^\beta))$. By the property of A-cast (and as graph $G_i$ is constructed on the basis of A-casted information) every honest party will receive, verify the validity of $D$'s A-casted information with respect to his own graph $G_i$ and agree on the $CORE$. Now the proof for this part follows from Lemma 5.

Part (2) of **Termination** says that if an honest party terminated AVSS-Share-SS, then every other honest party will terminate AVSS-Share-SS eventually. An honest $P_i$ has terminated the protocol implies that he has agreed on $CORE$. This means that $P_i$ has received and verified the validity of $D$'s A-casted information with respect to his own graph $G_i$. The same will happen eventually for all other

honest parties. Hence they will agree on $CORE$. Now the proof follows from Lemma 5. Part (3) of **Termination** follows from the correctness of OEC.

**Correctness:** If the honest parties terminate AVSS-Share-SS, then it implies that $\overline{s}(= \overline{F}(0,0))$ is properly $d$-shared among the parties in $\mathcal{P}$ (by Lemma 5), where $\overline{F}(x,y)$ is the unique bivariate polynomial of degree-$(d,t)$ defined by the honest parties in $CORE$. Moreover if $D$ is honest then $\overline{F}(x,y) = F(x,y)$ (follows from Lemma 1 and Lemma 4) and hence $\overline{s} = s$. Now the **Correctness** follows from the correctness of OEC.

**Secrecy:** Let $\mathcal{A}_t$ controls $P_1, \ldots, P_t$. So $\mathcal{A}_t$ will know $f_1(x), \ldots, f_t(x)$ and $p_1(y), \ldots, p_t(y)$. Throughout the protocol, the parties exchange common values (on row and column polynomials), which do not add any extra information to the view of $\mathcal{A}_t$. Now by the property of bivariate polynomial of degree-$(d,t)$, $d-t+1$ coefficients of $f_0(x) = F(x,0)$ will remain secure, where $F(x,y)$ is the polynomial used by $D$ to hide his secret $s$. So $s = f_0(0) = F(0,0)$ will remain secure. □

**Theorem 2.** *AVSS-Share-SS privately communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits and A-casts $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. Protocol AVSS-Rec-SS privately communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$.*

*Proof.* See the full version of the paper [21]. □

## 4 AVSS for Generating $d$-Sharing of Multiple Secrets

We now present an AVSS protocol consisting of two sub-protocols, namely AVSS-Share-MS and AVSS-Rec-MS: AVSS-Share-MS allows a dealer $D \in \mathcal{P}$ to $d$-share $\ell \geq 1$ secret(s) from $\mathbb{F}$, denoted as $S = (s^1, \ldots, s^\ell)$, among the parties in $\mathcal{P}$, with $t \leq d \leq 2t$; AVSS-Rec-MS allows the parties to reconstruct the secrets, given their $d$-sharing. Notice that we can generate $d$-sharing of $S$ by concurrently executing protocol AVSS-Share-SS (given in the previous section) $\ell$ times, once for each $s^i \in S$. But this will require a private communication of $\mathcal{O}(\ell n^2 \log(|\mathbb{F}|))$ and A-cast of $\mathcal{O}(\ell n^2 \log(|\mathbb{F}|))$ bits. However, our protocol AVSS-Share-MS requires a private communication of $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ and A-cast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. *Thus, the A-cast communication of our AVSS-Share-MS protocol is independent of $\ell$.* The idea behind protocol AVSS-Share-MS is same as AVSS-Share-SS. Protocol AVSS-Share-MS is divided into a sequence of same three phases, as in AVSS-Share-SS. We now present the corresponding protocols.

**Protocol Distribution-MS$(D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d)$**

CODE FOR $D$:

1. For $l = 1, \ldots, \ell$, select a random bivariate polynomials $F^l(x,y)$ of degree-$(d,t)$, such that $F^l(0,0) = s^l$ and send the row polynomial $f_i^l(x) = F^l(x,i)$ and column polynomial $p_i^l(y) = F^l(i,y)$ to $P_i$.

**Protocol Verification-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d$)**

i. CODE FOR $P_i$: Every party $P_i \in \mathcal{P}$ (including $D$) executes this code.

1. Wait to receive $\overline{f_i^l}(x)$ and $\overline{p_i^l}(y)$ for all $l = 1, \ldots, \ell$, from $D$.
2. Upon receiving, check whether (i) $\overline{f_i^l}(x)$ is a degree-$d$ polynomial for all $l = 1, \ldots, \ell$; and (ii) $\overline{p_i^l}(y)$ is a degree-$t$ polynomial for all $l = 1, \ldots, \ell$. If yes, then send $\overline{f_{ij}^l} = \overline{f_i^l}(j)$ and $\overline{p_{ij}^l} = \overline{p_i^l}(j)$ for all $l = 1, \ldots, \ell$, to $P_j$.
3. Upon receiving $\overline{f_{ji}^1}, \ldots, \overline{f_{ji}^\ell}$ and $\overline{p_{ji}^1}, \ldots, \overline{p_{ji}^\ell}$ from $P_j$, check if $\overline{f_i^l}(j) \stackrel{?}{=} \overline{p_{ji}^l}$ and $\overline{f_{ji}^l} \stackrel{?}{=} \overline{p_i^l}(j)$ for all $l = 1, \ldots, \ell$. If the equality holds, then confirm the consistency by A-casting $\mathtt{OK}(P_i, P_j)$.
4. Construct an undirected graph $G_i$ with $\mathcal{P}$ as vertex set. Add an edge $(P_j, P_k)$ in $G_i$ upon receiving (a) $\mathtt{OK}(P_k, P_j)$ from the A-cast of $P_k$ and (b) $\mathtt{OK}(P_j, P_k)$ from the A-cast of $P_j$.

ii. CODE FOR $D$: (Only $D$ executes this code): Same as in Protocol Verification-SS.

iii. CODE FOR $P_i$: (Every party $P_i \in \mathcal{P}$ (including $D$) executes this code): Same as in Protocol Verification-SS.

**Protocol d-Share-Generation-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d$)**

CODE FOR $P_i$:

1. For $l = 1, \ldots, \ell$, apply On-line Error Correcting (OEC) technique on $\overline{f_{ji}^l}$'s received from every $P_j$ in $CORE$ (during Protocol Verification-SS) and reconstruct degree-$t$ polynomial $\overline{p_i^l}(y)$ and output $\overline{s_i^l} = \overline{p_i^l}(0) = \overline{f_0^l}(i)$ as the $i^{th}$ share of $\overline{s^l}$ and terminate. $\overline{s^l}$ is now $d$-shared using polynomial $\overline{f_0^l}(x)$.

Protocol AVSS-Share-MS and AVSS-Rec-MS are now given in the following table.

**Protocol AVSS-Share-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d$)**

(a) $D$ executes Distribution-MS$(D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d)$;
(b) Each party $P_i$ participates in Verification-MS$(D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d)$;
(c) After agreeing on $CORE$, each party $P_i$ participates in d-Share-Generation-MS$(D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d)$ and terminates AVSS-Share-MS after locally outputting the shares corresponding to $D$'s committed secrets.

**Protocol AVSS-Rec-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d$)**

1. For $l = 1, \ldots, \ell$, each party $P_i \in \mathcal{P}$ privately sends the $i^{th}$ share of $s^l$, namely $s_i^l$, to every party $P_j \in \mathcal{P}$.
2. For $l = 1, \ldots, \ell$, party $P_i \in \mathcal{P}$ applies OEC on the received $s_j^l$'s to privately reconstruct $s^l$ and terminate AVSS-Rec-MS.

**Theorem 3.** (*AVSS-Share-MS*, *AVSS-Rec-MS*) *constitute a perfectly secure AVSS scheme for sharing $\ell \geq 1$ secret(s) from $\mathbb{F}$. AVSS-Share-MS privately communicates $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and A-casts $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. AVSS-Rec-MS privately communicates $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits.*

### 4.1 A Different Interpretation of AVSS-Share-MS

In AVSS-Share-MS, every secret $s^l$ for $l = 1, \ldots, \ell$ is $d$-shared using degree-$d$ polynomial $f_0^l(x) = F^l(x, 0)$. Now by the **Secrecy** proof of AVSS-Share-SS, given in Theorem 1, we can claim that $(d+1)-t$ coefficients of $f_0^l(x)$ are information theoretically secure for every $l = 1, \ldots, \ell$. This implies that AVSS-Share-MS shares $\ell(d+1-t)$ secrets with a private communication of $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and A-cast $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. As the A-cast communication is independent of $\ell$, we may ignore it and conclude that the amortized cost of sharing a single secret using AVSS-Share-MS is only $\mathcal{O}(n \log |\mathbb{F}|)$. This is because by setting $d = 2t$, we see that AVSS-Share-MS can share $\ell(t+1) = \Theta(\ell n)$ secrets by privately communicating $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits. Now putting it in other way, $D$ can share $\ell(t+1)$ secrets using AVSS-Share-MS by choosing a random polynomial $f_0^l(x)$ (of degree $d = 2t$) with lower order $t+1$ coefficients as secrets and then choosing a random degree-$(d, t)$ bivariate polynomial $F^l(x, y)$ with $F^l(x, 0) = f_0^l(x)$ for $l = 1, \ldots, \ell$ and finally executing AVSS-Share-MS with $F^1(x, y), \ldots, F^\ell(x, y)$.

Through we do not elaborate, we now mention another application of AVSS-Share-MS which uses the above interpretation. We can design an Asynchronous BA (ABA) protocol with an amortized communication cost of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits for reaching agreement on a single bit. To the best of our knowledge, there is only one ABA with $4t+1$ due to [13] which requires very high communication complexity (though polynomial in $n$).

*Remark 2.* The best known AVSS of [2] requires an amortized cost $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits for sharing a single secret. Hence AVSS-Share-MS shows a clear improvement over the AVSS of [2].

*Remark 3.* The idea of hiding multiple secrets in a single polynomial was explored earlier in [12] in the context of passive adversary in synchronous network. Doing the same in asynchronous network, in the presence of active adversary is bit tricky and calls for new techniques. Though we can hide $(d+1-t)$ secrets in each degree-$d$ polynomial $f_0^l(x)$ using protocol AVSS-Share, we hide only one secret, namely $s^l$ in $f_0^l(x)$. This is because in our AMPC protocol, we require that each degree-$d$ polynomial hides only one secret. However, hiding multiple secrets in a degree-$d$ polynomial will be useful in the context of ABA.

## 5 Protocol for Generating $(t, 2t)$-Sharing

We now present a protocol called (t,2t)-Share-MS that allows a dealer $D \in \mathcal{P}$ to concurrently generate $(t, 2t)$-sharing of $\ell \geq 1$ secrets. We explain the idea of the protocol for a single secret $s$. $D$ invokes AVSS-Share-MS to $t$-share $s$. Let $f(x)$ be the degree-$t$ polynomial using which $s$ is $t$-shared. $D$ also invokes AVSS-Share-MS to $(2t-1)$-share a random value $r$. Let $g(x)$ be the degree-$(2t-1)$ polynomial using which $r$ is $(2t-1)$-shared. It is easy to see that $h(x) = f(x) + xg(x)$ will be a degree-$2t$ polynomial, such that $h(0) = s$. So if party $P_i$ locally computes $h(i) = f(i) + i \cdot g(i)$, then this will generate the $2t$-sharing of $s$. Protocol (t,2t)-Share-MS follows this principle for all the $\ell$ secrets concurrently.

**Theorem 4.** *Protocol (t,2t)-Share-MS satisfies the following properties:*

1. TERMINATION: *(a) If $D$ is honest, then each honest $P_i$ will terminate (t,2t)-Share-MS. (b) If $D$ is corrupted and some honest $P_i$ terminates (t,2t)-Share-MS, then all honest parties will also eventually terminate the protocol.*
2. CORRECTNESS: *If honest parties terminate (t,2t)-Share-MS, then there are $\ell$ values, that are $(t, 2t)$-shared among the parties in $\mathcal{P}$.*
3. SECRECY: *$\mathcal{A}_t$ will have no information about the secrets of an honest $D$.*
4. COMMUNICATION COMPLEXITY: *The protocol privately communicates $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and A-cast $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.*

*Proof.* Follows from the properties of AVSS-Share-MS and protocol steps. □

**Protocol (t,2t)-Share-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell)$)**

CODE FOR $D$:

1. Invoke AVSS-Share-MS$(D, \mathcal{P}, S = (s^1, \ldots, s^\ell), t)$ and AVSS-Share-MS$(D, \mathcal{P}, R = (r^1, \ldots, r^\ell), 2t-1)$, where the elements of $R$ are random.

CODE FOR $P_i$:

1. Participate in AVSS-Share-MS$(D, \mathcal{P}, S = (s^1, \ldots, s^\ell), t)$ and AVSS-Share-MS$(D, \mathcal{P}, R = (r^1, \ldots, r^\ell), 2t-1)$. Wait to terminate AVSS-Share-MS$(D, \mathcal{P}, S, t)$ with $i^{th}$ shares of $S = (s^1, \ldots, s^\ell)$, say $(\varphi_i^1, \ldots, \varphi_i^\ell)$. Wait to terminate AVSS-Share-MS$(D, \mathcal{P}, R, 2t-1)$ with $i^{th}$ shares of $R = (r^1, \ldots, r^\ell)$, say $(\chi_i^1, \ldots, \chi_i^\ell)$.
2. For $l = 1, \ldots, \ell$, locally compute $\psi_i^l = \varphi_i^l + i \cdot \chi_i^l$, output $\varphi_i^l$ and $\psi_i^l$ as $i^{th}$ share of $s$ corresponding to $t$ and $2t$-sharing respectively and terminate (t,2t)-Share-MS.

*Remark 4.* In [2] the authors presented a perfectly secure protocol, that privately communicates $\mathcal{O}(\ell n^3 \log |\mathbb{F}|)$ bits and A-casts $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits to generate $(t, 2t)$-sharing of $\ell$ secrets (For complete details, see [21]). Thus (t,2t)-Share-MS gains a factor of $\Omega(n)$ in communication complexity for generating $(t, 2t)$-sharing. In fact, it is this gain of $\Omega(n)$, which helps our AMPC protocol to gain $\Omega(n)$ in communication complexity, compared to the AMPC of [2]. In [20], a protocol with same communication complexity as ours is given for generating $(t, 2t)$-sharing. However, the protocol has negligible error probability in CORRECTNESS and TERMINATION.

## 6 AMPC Protocol Overview

Once we have an efficient protocol for generating $(t, 2t)$-sharing, our AMPC protocol proceeds in the same way as that of [2,20]. Specifically, our AMPC protocol is a sequence of three phases: preparation, input and computation. In the preparation phase, corresponding to each multiplication and random gate, a $(t, 2t)$-sharing of random secret will be generated. In the input phase the parties $t$-share their inputs and agree on a common set of at least $n - t$ parties who correctly $t$-shared their inputs. In the computation phase, based on the inputs of the parties in this common set, the actual circuit will be computed gate by gate, such that the output of the intermediate gates are always kept as secret and are $t$-shared among the parties. We now elaborate on each of the three phases.

### 6.1 Preparation Phase

The goal of protocol PreparationPhase is to generate $(t, 2t)$-sharing of $c_M + c_R$ random *secrets*. For this, each individual party acts as a dealer and $(t, 2t)$-share $\frac{c_M+c_R}{n-2t}$ random values. Then an instance of ACS protocol is executed to agree on a common set $C$ of $n-t$ parties, who have correctly $(t, 2t)$-shared $\frac{c_M+c_R}{n-2t}$ values. Out of these $n-t$ parties, at least $n-2t$ are honest, who have indeed $(t, 2t)$-shared random values, which are unknown to $\mathcal{A}_t$. So if we consider the $(t, 2t)$-sharing done by the honest parties (each of them has done $\frac{c_M+c_R}{n-2t}$ $(t, 2t)$-sharing) in common set $C$, then we will get $\frac{c_M+c_R}{n-2t} * (n-2t) = c_M + c_R$ random $(t, 2t)$-sharing. For this, we use *Vandermonde Matrix* [11] and its ability to extract randomness which has been exploited in [11,2]. The same approach is also used in the preparation phase of [20]. Due to space constraint, we present the protocol in [21] and state only the following lemma:

**Lemma 6.** *Each honest party will eventually terminate PreparationPhase. The protocol generates $(t, 2t)$-sharing of $c_M + c_R$ secret random values, unknown to $\mathcal{A}_t$, by privately communicating $O((c_M + c_R)n^2 \log |\mathbb{F}|)$ bits, A-casting $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and executing one instance of ACS.*

### 6.2 Input Phase

In protocol InputPhase, each party acts as a dealer and $t$-share his input(s) by executing an instance of AVSS-Share-MS. The parties then execute ACS to agree on a common set $C$ of $n-t$ parties, whose instances of AVSS-Share-MS have terminated. As the input(s) of the parties in $C$ will be considered for computation (of the circuit), each party considers the $t$-sharing of all the inputs shared by parties, only in $C$. As the protocol is very straight forward, we present it in [21] and state the following lemma:

**Lemma 7.** *Each honest party terminates InputPhase. The protocol generates $t$-sharing of the inputs of the parties in $C$, such that $\mathcal{A}_t$ has no information about the inputs of the honest parties in $C$, by privately communicating $O(c_I n^2 \log |\mathbb{F}|)$ bits, A-casting $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and executing one ACS.*

### 6.3 Computation Phase

Once the input phase is over, in the computation phase, the circuit is evaluated gate by gate, where all inputs and intermediate values are $t$-shared among the parties. As soon as a party holds his shares of the input values of a gate, he joins the computation of the gate. Due to the linearity of the used $t$-sharing, linear gates can be computed locally simply by applying the linear function to the shares. With every random gate, one random $(t, 2t)$-sharing (from the preparation phase) is associated, whose $t$-sharing is directly used as outcome of the random gate. With every multiplication gate, one random $(t, 2t)$-sharing (from the preparation phase) is associated, which is then used to compute $t$-sharing of the product, following the technique of [11]: Let $z = xy$, where $x, y$

are the inputs of the multiplication gate, where $x, y$ are $t$-shared, i.e. $[x]_t, [y]_t$. Moreover, let $[r]_{(t,2t)}$ be the $(t, 2t)$-sharing associated with the multiplication gate, where $r$ is a secret random value. For computing $[z]_t$, the parties compute $[\Lambda]_{2t} = [x]_t.[y]_t + [r]_{2t}$. Then $\Lambda$ is privately reconstructed by every $P_i \in \mathcal{P}$. Now every party defines $[\Lambda]_t$ as the default sharing of $\Lambda$ and computes $[z]_t = [\Lambda]_t - [r]_t$. The secrecy of $z$ follows from [11,2]. The same approach is also used in the computation phase of AMPC protocol of [2,20]. Due to space constraint, we present the protocol in [21] and state the following lemma:

**Lemma 8.** *Each honest party will eventually terminate* ComputationPhase. *Given $(t, 2t)$-sharing of $c_M + c_R$ secret random values, the protocol securely evaluates the circuit by privately communicating $O((c_M n^2 + c_O n) \log |\mathbb{F}|)$ bits.*

### 6.4 Final AMPC Protocol

Now our new AMPC protocol called AMPC for evaluating function $f$ is: (1). Invoke PreparationPhase (2). Invoke InputPhase (3). Invoke ComputationPhase.

**Theorem 5.** *Protocol AMPC is an optimally resilient, perfectly secure AMPC protocol that privately communicates $\mathcal{O}(((c_I + c_M + c_R)n^2 + c_O n) \log |\mathbb{F}|)$ bits, A-casts $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and requires 2 invocations to ACS. Each honest party will eventually terminate AMPC.*

**Acknowledgement.** We would sincerely like to thank Tal Rabin for giving several insightful remarks on the earlier version of this paper which significantly improved the presentation of the protocols.

## References

1. Beerliová-Trubíniová, Z., Hirt, M.: Efficient multi-party computation with dispute control. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 305–328. Springer, Heidelberg (2006)
2. Beerliová-Trubíniová, Z., Hirt, M.: Simple and efficient perfectly-secure asynchronous MPC. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 376–392. Springer, Heidelberg (2007)
3. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure MPC with linear communication complexity. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 213–230. Springer, Heidelberg (2008)
4. Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous secure computation. In: STOC, pp. 52–61 (1993)
5. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, pp. 1–10 (1988)
6. BenOr, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience. In: PODC, pp. 183–192 (1994)
7. Bracha, G.: An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In: PODC, pp. 154–162 (1984)
8. Canetti, R.: Studies in Secure Multiparty Computation and Applications. PhD thesis, Weizmann Institute, Israel (1995)

9. Canetti, R., Rabin, T.: Fast asynchronous Byzantine Agreement with optimal resilience. In: STOC, pp. 42–51 (1993)
10. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC, pp. 11–19 (1988)
11. Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 572–590. Springer, Heidelberg (2007)
12. Franklin, M.K., Yung, M.: Communication complexity of secure computation. In: STOC, pp. 699–710 (1992)
13. Feldman, P., Micali, S.: An optimal algorithm for synchronous Byzantine Agreemet. In: STOC, pp. 639–648 (1988)
14. Fitzi, M., Garay, J., Gollakota, S., Pandu Rangan, C., Srinathan, K.: Round-optimal and efficient verifiable secret sharing. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 329–342. Springer, Heidelberg (2006)
15. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The round complexity of verifiable secret sharing and secure multicast. In: STOC, pp. 580–589 (2001)
16. Katz, J., Koo, C., Kumaresan, R.: Improving the round complexity of VSS in point-to-point networks. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 499–510. Springer, Heidelberg (2008)
17. Patra, A., Choudhary, A., Rabin, T., Pandu Rangan, C.: The round complexity of verifiable secret sharing revisited. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 487–504. Springer, Heidelberg (2009)
18. Patra, A., Choudhary, A., Pandu Rangan, C.: Efficient asynchronous multiparty computation with optimal resilience. Cryptology ePrint Archive, Report 2008/425 (2008)
19. Patra, A., Choudhary, A., Pandu Rangan, C.: Efficient asynchronous Byzantine Agreement with optimal resilience. In: PODC, pp. 92–101 (2009)
20. Patra, A., Choudhary, A., Pandu Rangan, C.: Unconditionally secure asynchronous multiparty computation with quadratic communication per multiplication gate. Cryptology ePrint Archive, Report 2009/087 (2009)
21. Patra, A., Choudhary, A., Pandu Rangan, C.: Communication Efficient Perfectly Secure VSS and MPC in Asynchronous Networks with Optimal Resilience Cryptology ePrint Archive, Report 2010/007 (2010)
22. Prabhu, B., Srinathan, K., Pandu Rangan, C.: Trading players for efficiency in unconditional multiparty computation. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 342–353. Springer, Heidelberg (2003)
23. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: STOC, pp. 73–85 (1989)
24. Srinathan, K., Pandu Rangan, C.: Efficient asynchronous secure multiparty distributed computation. In: Roy, B., Okamoto, E. (eds.) INDOCRYPT 2000. LNCS, vol. 1977, pp. 117–129. Springer, Heidelberg (2000)
25. Yao, A.C.: Protocols for secure computations. In: FOCS, pp. 160–164 (1982)

# Avoiding Full Extension Field Arithmetic in Pairing Computations

Craig Costello*, Colin Boyd, Juan Manuel González Nieto, and Kenneth Koon-Ho Wong

Information Security Institute
Queensland University of Technology, GPO Box 2434,
Brisbane QLD 4001, Australia
{craig.costello,c.boyd,j.gonzaleznieto,kk.wong}@qut.edu.au

**Abstract.** The most costly operations encountered in pairing computations are those that take place in the full extension field $\mathbb{F}_{p^k}$. At high levels of security, the complexity of operations in $\mathbb{F}_{p^k}$ dominates the complexity of the operations that occur in the lower degree subfields. Consequently, full extension field operations have the greatest effect on the runtime of Miller's algorithm. Many recent optimizations in the literature have focussed on improving the overall operation count by presenting new explicit formulas that reduce the number of subfield operations encountered throughout an iteration of Miller's algorithm. Unfortunately, almost all of these improvements tend to suffer for larger embedding degrees where the expensive extension field operations far outweigh the operations in the smaller subfields. In this paper, we propose a new way of carrying out Miller's algorithm that involves new explicit formulas which reduce the number of full extension field operations that occur in an iteration of the Miller loop, resulting in significant speed ups in most practical situations of between 5 and 30 percent.

**Keywords:** Pairings, Miller's algorithm, Tate pairing, ate pairing.

## 1 Introduction

At the beginning of this century, pairing-based cryptography became extremely popular after the first practical identity-based encryption scheme was constructed using the powerful bilinearity property of pairings [13]. Accompanied by many other exciting breakthroughs that resulted from pairings, the discovery of ID-based encryption heightened the demand for practical pairings which can be computed efficiently. Since then, much research has been invested towards achieving faster pairings and consequently the speed of computing Miller's algorithm [34] for calculating pairings has significantly increased. Initial improvements in pairing computations were spearheaded by evidence that the Tate pairing was much

* The first author acknowledges funding from the Queensland Government Smart State PhD Scholarship. This work has been supported in part by the Australian Research Council through Discovery Project DP0666065.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 203–224, 2010.

more efficient than the Weil pairing, since the final exponentiation in the Tate pairing facilitates several clever simplifications in the Miller iterations [4,6,7,35]. The continual evolution of security requirements and standards has lead to a large emphasis being placed on obtaining secure curve constructions for a range of embedding degrees. As a result, the construction of pairing-friendly curves has become an active field of research in itself [5,14,36,20,8,24,10,21,30], so that cryptographers can now choose from an array of flexible curve options that offer high levels of efficiency in pairing computations [22]. More recently, Hess, Smart and Vercauteren [27] generalized prior work by Duursma and Lee [19] and Barreto *et al.* [3] to develop the ate pairing which benefits from a truncated loop length and is usually much faster than the Tate pairing. The ate pairing has since enjoyed its own improvements [33,32], to the point where ate pairing variants can now be computed with optimal loop lengths [37,26].

In very recent times, researchers have achieved further speedups by deriving fast explicit formulas for specific stages of a Miller iteration [15,18,28,1,16,17], so that each iteration requires less subfield operations, resulting in a faster pairing. Unfortunately, such improvements are less effective when applied to the Tate pairing because the operations that are saved occur in the base field $\mathbb{F}_p$, and as the embedding degree $k$ gets large, the complexity of the operations occurring in the full extension field $\mathbb{F}_{p^k}$ dominates the complexity of those operations occurring in $\mathbb{F}_p$, so that the relative speedup resulting from savings in the base field becomes much less. In the ate pairing with a twist of degree $d$, faster explicit formulas save operations in the subfield $\mathbb{F}_{p^{k/d}}$, the complexity of which grows at the same rate as the complexity of operations in $\mathbb{F}_{p^k}$, so that an increased embedding degree will not drastically effect the relative speedup. Nevertheless, optimized implementations of the ate pairing make use of the highest available twist for a given $k$, so that the complexity of operations in $\mathbb{F}_{p^{k/d}}$ is much less than those in $\mathbb{F}_{p^k}$. For example, the ate pairing computed on a BN curve [8] where $k = 12$ uses a sextic twist ($d = 6$), so that any computations saved through faster explicit formulas are those in the much smaller field $\mathbb{F}_{p^2}$. An optimized construction of the extension field [31,9] results in the complexity of operations in $\mathbb{F}_{p^{12}}$ being no less than 15 times greater than the analogous operations in $\mathbb{F}_{p^2}$, so that any speedups that result from faster explicit formulas are still greatly overshadowed by the expensive operations in $\mathbb{F}_{p^{12}}$. At any level, full extension field operations greatly outweigh subfield operations for both Tate-and ate-like pairings.

Eisenträger, Lauter and Montgomery [29] managed to avoid full extension field arithmetic in pairing computations by combining two linear Miller functions into a single function of degree 2, which they call a parabola, and achieving a speedup by replacing two multiplications by the two linear functions with a single multiplication by the parabola. However, the algorithm in [29] has limited application in state-of-the-art pairing implementations because it only applies to stages of the algorithm that require point addition, and optimized implementations will choose loop parameters with low Hamming weight that minimize the occurrence of these additions. Blake, Murty and Xu [12] extended the observations in [29] to form combinations of Miller lines that apply to every

iteration of the Miller loop, proposing a version of Miller's algorithm that is somewhat analogous to the $2^n$-ary windowing methods for general exponentiation (cf. [2, §9]), using a window of size $n = 2$. Again, the techniques proposed in [12] are not optimized for modern implementations of Miller's algorithm because the main benefit of the combined linear functions in their case was to avoid field divisions, a problem that became obsolete after the introduction of denominator elimination in [4]. In this paper, we extend the notion of combining Miller lines into higher degree polynomials and present a more general approach, which we call Miller $2^n$-tuple-and-add. Specifically, we show how to combine explicit formulas from $n$ consecutive Miller double-and-add iterations into more complicated explicit formulas for one Miller $2^n$-tuple-and-add iteration. The price we pay for spending more subfield operations to evaluate these more complicated formulas is greatly rewarded by the large savings that result from avoiding costly arithmetic in the full extension field. For both Tate and ate-like pairings, we show that the Miller $2^n$-tuple-and-add algorithm achieves significant speedups over the standard Miller double-and-add routine for the majority of pairing-friendly embedding degrees. Our method offers (among others) the following important advantages over the prior work in [12]:

- Our method works for general $n \geq 1$. All prior work (except for $n = 2$ in [12]) has used $n = 1$.
- Our method handles any addition steps encountered in Miller's $2^n$-tuple-and-add algorithm in exactly the same way, regardless of the $2^n$-ary representation of the loop parameter. The method in [12] for $n = 2$ uses formulas that differ depending on the quarternary representation of the loop parameter. An important consequence of this is that higher values of $n$ do not result in more complex additions, as they do for $n = 2$ in [12].
- The techniques and analyses in [12] focus on reducing the number of field divisions (inversions) that occur in the affine representation of the Miller lines. Field inversions are extremely costly in pairing implementations and have been phased out thanks to denominator elimination and the application of non-affine (projective) coordinate systems to pairing computations that eliminate field inversions altogether. The explicit formulas herein are derived using projective coordinates, and these formulas are reduced to give much faster operation counts.

The rest of this paper is organized as follows. Section 2 provides a brief background on pairings and Miller's algorithm. In Section 3 we describe the general Miller $2^n$-tuple-and-add algorithm, before discussing a general strategy to obtain explicit formulas for $2^n$-tuple-and-add in Section 4. In Section 5, we derive explicit formulas for the cases of Miller quadruple-and-add ($n = 2$) and Miller octuple-and-add ($n = 3$), and obtain operation counts for a typical iteration in each scenario. In Section 6, we compare the operation counts for the quadruple-and-add and octuple-and-add algorithm with the standard double-and-add algorithm. We draw conclusions in the same section.

## 2 Background

Let $E$ be an elliptic curve over $\mathbb{F}_p$. Assume $E$ is given by the short Weierstrass equation $y^2 = x^3 + ax + b$ and let $\mathcal{O}$ be the neutral element on $E$. For the points $R, S \in E$, let $l_{R,S}$ and $v_{R,S}$ respectively be the sloped and vertical lines in the standard chord-and-tangent addition of $R$ and $S$, the divisors of which are $\text{div}(l_{R,S}) = (R)+(S)+(-(R+S))-3(\mathcal{O})$ and $\text{div}(v_{R,S}) = (-(R+S))+(R+S)-2(\mathcal{O})$. When $R = S$, we have $l_{R,R}$ and $v_{R,R}$ as the sloped and vertical lines in the point doubling of $R$. Herein we let $g_{R,S}$ represent the quotient $g_{R,S} = l_{R,S}/v_{R,S}$, with associated divisor $\text{div}(g_{R,S}) = (R) + (S) - (R + S) - (\mathcal{O})$. For $v \in \mathbb{Z}$, let $f_{v,R}$ be a function with divisor

$$f_{v,R} = v(R) - ([v]R) - (v - 1)(\mathcal{O}).$$

Let $k$ be the embedding degree of $E$ with respect to some large prime $r$ and let $E[r]$ denote the group of $r$-torsion points on $E$. We use $\pi_p$ to denote the $p$-power Frobenius endormorphism on $E$ and we define two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ using the two eigenspaces of $\pi_p$ as $\mathbb{G}_1 = E[r] \cap \ker(\pi_p - [1])$ and $\mathbb{G}_2 = E[r] \cap \ker(\pi_p - [p])$.

For two points $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, the Tate pairing $e_r : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ is computed as $e_r(P, Q) = f_{r,P}(Q)^{(p^k-1)/r}$. Let $T = t - 1$, where $t$ is the trace of the Frobenius on $E$. The ate pairing $a_T : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_3$ is computed as $a_T(Q, P) = f_{T,Q}(P)^{(p^k-1)/r}$. In the coming sections, we treat both pairings simultaneously by letting the required pairing be computed as $f_{m,R}(S)^{(p^k-1)/r}$, where it is understood that in the Tate pairing we have $m = r$, $R \in \mathbb{G}_1$ and $S \in \mathbb{G}_2$, whilst in the ate pairing we have $m = T$, $R \in \mathbb{G}_2$ and $S \in \mathbb{G}_1$.

When counting field operations, we use $\mathbf{M}$ and $\mathbf{S}$ to denote the respective costs of a multiplication and a squaring in the field $\mathbb{F}_{p^k}$, and we use $\mathbf{m}$ and $\mathbf{s}$ to represent the costs of a multiplication and a squaring in the subfield $\mathbb{F}_{p^e}$, where $e = 1$ for Tate-like pairings and $e = k/d$ for ate-like pairings with twists of degree $d$. In some instances it is necessary to count operations in more than two fields, in which case we avoid ambiguities by letting $\mathbf{m}_i$ and $\mathbf{s}_i$ denote the costs of a multiplication and a squaring in the field $\mathbb{F}_{p^i}$. Lastly, we report the cost of a multiplication by a curve constant (or a small power of a curve constant) as $\mathbf{d}$.

Since the introduction of the original ate pairing [27], several variants with even shorter loop lengths have emerged [33], including the R-ate pairing [32] which often achieves the optimal loop length [37,26]. All of these variants also take $R \in \mathbb{G}_2$ and $S \in \mathbb{G}_1$ and compute $f_{m,R}(S)$, the only difference being the construction (and size) of the loop parameter $m$. We refer to all such pairings collectively as ate-like pairings ($a : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_3$), and hereafter we make no specifications regarding the loop length, since it plays no role in the results of this paper. Identically, we put the twisted ate pairing [27] under the umbrella of Tate-like pairings ($e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$), since the twisted ate pairing takes its respective inputs from the same groups as the Tate pairing.

Using $f_{i+j,R} = f_{i,R} \cdot f_{j,R} \cdot g_{[i]R,[j]R}$, the usual version of Miller's algorithm computes the required function in $\lfloor \log_2(m) \rfloor$ iterations by initializing $f_{1,R}(S) = 1$ and progressively building the functions $f_{v,R}(S)$ (for $v < m$) to approach $f_{m,R}(S)$ in a double-and-add-like fashion, as summarized in Algorithm 1.

**Algorithm 1.** Miller double-and-add Algorithm

**Input:** $R$, $S$, $m = (m_{l-1}...m_1, m_0)_2$.
**Output:** $f_{m,R}(S)$.

1: $T \leftarrow R$, $f \leftarrow 1$.
2: **for** $i = l - 2$ to $0$ **do**
3: Compute $g = g_{T,T}(S)$
4: $T \leftarrow [2]T$.
5: $f \leftarrow f^2 \cdot g$.
6: **if** $m_i \neq 0$ **then**
7: Compute $g = g_{T,R}(S)$
8: $T \leftarrow T + R$.
9: $f \leftarrow f \cdot g$.
10: **end if**
11: **end for**
12: **return** $f$.

At the beginning of an iteration of Algorithm 1, let the intermediate multiple of the point $R$ be $T = [v]R$, so that the current Miller function $f$ relating to the point $T$ has divisor

$$\mathrm{div}(f_{v,R}) = v(R) - ([v]R) - (v-1)(\mathcal{O}).$$

Miller's double-and-add algorithm forms the function $f_{2v,R}$ relating to the point $[2]T = [2v]R$ as $f_{2v,R} = f_{v,R}^2 \cdot g_{[2]T}$, where $\mathrm{div}(g_{[2]T}) = 2(T) - ([2]T) - (\mathcal{O})$, so that $f_{2v,R}$ has divisor

$$\begin{aligned}\mathrm{div}(f_{2v,R}) &= \mathrm{div}(f_{v,R}^2 \cdot g_{[2]T}) = 2 \cdot \mathrm{div}(f_{v,R}) + \mathrm{div}(g_{[2]T}) \\ &= 2 \cdot \big(v(R) - ([v]R) - (v-1)(\mathcal{O})\big) + (2(T) - ([2]T) - (\mathcal{O})) \\ &= 2v(R) - ([2v]R) - (2v-1)(\mathcal{O}).\end{aligned}$$

We obtain the Miller function $f_{2v,R}$ by squaring the Miller function $f_{v,R}$ and multiplying this result by the "line" function(s) involved in the point doubling of $T$. In a standard implementation of Miller's algorithm, the functions $f_{v,R}$ and $g_{T,T}$ are contained in the full extension field, so that the function update (step 5 of Algorithm 1) comes at a cost of $1\mathbf{M} + 1\mathbf{S}$. Assuming (for now) that no intermediate addition operations are required (i.e. $m_i = 0$ for $n$ consecutive $i$'s in Algorithm 1), $n$ consecutive iterations of Miller's double-and-add algorithm above transform the function $f_{v,R}$ into the function $f_{2^n v,R}$. The cost of the $n$ function updates that occur in $n$ such iterations is then $n\mathbf{M} + n\mathbf{S}$.

## 3 $2^n$-ary Pairings: Miller $2^n$-tuple-and-add

In this section we generalize the above (double-and-add) method by combining $n$ consecutive doubling steps into one $2^n$-tupling step and we show that this reduces the number of expensive function updates that occur in $\mathbb{F}_{p^k}$. For any

$n$, we naturally refer to this process as the Miller $2^n$-tuple-and-add algorithm. Consider $n$ consecutive squarings on the function $f_{v,R}$, which equates to raising $f_{v,R}$ to the power $2^n$. The divisor of the resulting function is given as

$$\mathrm{div}\big((f_{v,R})^{2^n}\big) \quad = \quad 2^n \cdot \mathrm{div}(f_{v,R}) \quad = \quad 2^n v(R) - 2^n([v]R) - 2^n(v-1)(\mathcal{O}). \tag{1}$$

To obtain the desired Miller function $f_{2^n v,R}$ from $f_{v,R}$, we must now find a function $f^*$ such that $\mathrm{div}((f_{v,R})^{2^n}) + \mathrm{div}(f^*) = \mathrm{div}(f_{2^n v,R}) = 2^n v(R) - ([2^n v]R) - (2^n v - 1)(\mathcal{O})$. We construct $f^*$ as

$$f^* = \prod_{i=1}^{n} (g_{[2^{i-1}]T,[2^{i-1}]T})^{2^{n-i}}, \tag{2}$$

the divisor of which is

$$\begin{aligned} \mathrm{div}(f^*) \quad &= \quad \sum_{i=1}^{n} 2^{n-i} \cdot \mathrm{div}(g_{[2^{i-1}]T,[2^{i-1}]T}) \\ &= \quad \sum_{i=1}^{n} 2^{n-i} \cdot (2([2^{i-1}]T) - ([2^i]T) - (\mathcal{O})) \\ &= \quad 2^n(T) - ([2^n]T) - (2^n - 1)(\mathcal{O}). \end{aligned} \tag{3}$$

Substituting $T = [v]R$ into (3) and combining this with (1) reveals that $\mathrm{div}((f_{v,R})^{2^n}) + \mathrm{div}(f^*) = \mathrm{div}(f_{2^n v,R})$, so that $f^*$ is indeed the required function. We note that the construction of $f^*$ is intuitive. Namely, $f^*$ is simply the product of the $n$ different $g$'s that are formed throughout each of the $n$ equivalent double-and-add iterations, each of which accumulates a different exponent depending on how many squarings it encounters in the iterations that follow. In this light, Miller $2^n$-tuple-and-add is much the same as Miller double-and-add; the major difference is that in Miller $2^n$-tuple-and-add we do not multiply the Miller function by its update $g$ immediately after it is squared. Rather, we form a product $f^*$ of $n$ powers of such $g$'s and we delay the multiplication of $f^*$ by $f$ so that it occurs only once in what is the equivalent of $n$ double-and-add iterations.

For the addition step in the Miller $2^n$-tuple-and-add algorithm, we now have to consider adding some multiple $[w]R$ of $R$ ($w < 2^n$) to the intermediate point and updating the Miller function accordingly. Suppose the intermediate point is $T = [v]R$ and the related Miller function prior to the addition has divisor $\mathrm{div}(f_{v,R}) = v(R) - ([v]R) - (v-1)(\mathcal{O})$ as before. We require a function $f^+$ such that $\mathrm{div}(f_{v,R}) + \mathrm{div}(f^+) = \mathrm{div}(f_{(v+w),R}) = (v+w)(R) - ([v+w]R) - (v+w-1)(\mathcal{O})$. The straightforward way to construct such a function is

$$f^+ = \prod_{i=0}^{w-1} g_{T+[i]R,R}, \tag{4}$$

the divisor of which is

$$\begin{aligned}\operatorname{div}(f^+) &= \sum_{i=0}^{w-1} \operatorname{div}(g_{T+[i]R,R}) \\ &= \sum_{i=0}^{w-1} \left[(R) + (T+[i]R) - (T+[i+1]R) - (\mathcal{O})\right] \\ &= w(R) + (T) - (T+[w]R) - w(\mathcal{O}).\end{aligned}$$

Again, substituting $T = [v]R$ gives $\operatorname{div}(f^+) = w(R)+([v]R)-([v+w]R)-w(\mathcal{O})$, so that $\operatorname{div}(f_{v,R}) + \operatorname{div}(f^+) = \operatorname{div}(f_{(v+w),R})$, and we see that $f^+$ is clearly the desired function. However, if we compute $f^+$ in the above fashion, we have to compute the product of $w$ different addition lines, and since $w$ can take any value between 1 and $2^n - 1$, computing the addition step with the explicit formulas that result from the product in (4) can become quite costly. Instead, consider an alternative method of computing the addition line as follows. Let $f^+_{\text{alt}}$ be such that $\operatorname{div}(f^+_{\text{alt}}) = \operatorname{div}(f^+)$ and take

$$f^+_{\text{alt}} = f_{w,R} \cdot g_{[v]R,[w]R}, \tag{5}$$

so that $\operatorname{div}(f^+_{\text{alt}}) = \operatorname{div}(f_{w,R})+\operatorname{div}(g_{[v]R,[w]R}) = w(R)+([v]R)-([v+w]R)-w(\mathcal{O})$. The advantage of the computation of $f^+_{\text{alt}}$ over the computation of $f^+$ is that $f^+_{\text{alt}}$ is comprised of only two functions, regardless of the size of $w$. Moreover, the function $f_{w,R}$ is the same function throughout the entire Miller $2^n$-tupling loop and does not change depending on where the addition/s occurs. Thus, the $f_{w,R}$'s can be precomputed (for all necessary values of $w$) prior to entering the Miller $2^n$-tupling loop so that we must only construct one new line function ($g_{[v]R,[w]R}$) at each addition stage. Importantly, this addition line is computed by applying the standard addition formulas to the coordinates of the point $[v]R$, which changes in each iteration, and the point $[w]R$ whose coordinates can be cached initially. From here on, the construction of $f^+$ refers to the construction of $f^+_{\text{alt}}$ described in (5). We summarize in Algorithm 2, where we note that the first value in the base $2^n$ representation of $m$ will not be $m_{l-1} = 1$ in general, so that we begin with an addition before entering the loop when $m_{l-1} \neq 1$.

In regards to full extension field arithmetic only, one standard iteration of Algorithm 2 (which usually has $m_i = 0$) requires $1\mathbf{M} + n\mathbf{S}$. When $n = 1$, we recover the usual Miller double-and-add algorithm which requires $\lfloor \log_2(m) \rfloor$ iterations, each incurring $1\mathbf{M} + 1\mathbf{S}$. For $n = 2$, the algorithm requires half as many iterations ($\lfloor \log_4(m) \rfloor$) that each incur a cost of $1\mathbf{M} + 2\mathbf{S}$, offering a $1\mathbf{M}$ saving over two equivalent standard double-and-add iterations. For general $n$, we save $(n-1)\mathbf{M}$ for each of the $\lfloor \log_{2^n}(m) \rfloor$ iterations of the Miller $2^n$-tuple-and-add algorithm, giving a relative saving of $\frac{n-1}{n}\mathbf{M}$ over each equivalent standard double-and-add iteration. Therefore the larger we allow $n$ to become, the more full extension field arithmetic we can avoid in the pairing computation.

The price we pay for increasing $n$ is an increase in the complexity of the formulas required to compute the function $f^*$. As $n$ grows, the size of $f^*$ (in its explicit form) grows rapidly so that many more operations are required to compute it. However, these operations are performed in substantially smaller subfields of the full extension field, where the computations are much cheaper. We can achieve significant speedups in the pairing computation if the price we pay for computing the more complex product of line functions $f^*$ in the smaller subfields of $\mathbb{F}_{p^k}$ is less than the savings we obtain in $\mathbb{F}_{p^k}$ itself.

In the following section we shed light on the details concerning the combination of steps 6 and 7 and the combination of steps 10 and 11 that are summarized in Algorithm 2.

**Algorithm 2.** Miller $2^n$-tuple-and-add Algorithm

**Input:** $R$, $S$, $m = (m_{l-1}...m_1, m_0)_{2^n}$, and the necessary precomputed values of $w[R]$ where $w < 2^n$.
**Output:** $f_{m,R}(S)$.

1: $T \leftarrow R$, $f \leftarrow 1$.
2: Compute function $f^+$ as the product described in (5) with $w = m_{l-1}$.
3: $f \leftarrow f \cdot f^+$.
4: $T \leftarrow T + [m_{l-1}]R$.
5: **for** $i = l - 2$ to 0 **do**
6: Compute function $f^*$ in the $2^n$-tupling of $T$.
7: $T \leftarrow [2^n]T$.
8: $f \leftarrow f^{2^n} \cdot f^*$.
9: **if** $m_i \neq 0$ **then**
10: Compute function $f^+$ as the product described in (5) with $w = m_i$.
11: $T \leftarrow T + [m_i]R$.
12: $f \leftarrow f \cdot f^+$.
13: **end if**
14: **end for**
15: **return** $f$.

## 4 A Strategy for Obtaining Explicit Formulas

This section provides the details for deriving explicit formulas for Miller $2^n$-tuple-and-add implementations. We pay close attention to the steps in Algorithm 2 that require deeper explanations.

*Line 6 of Algorithm 2:* Algorithm 3 (below) uses the standard doubling formulas to construct the affine line product $f^*$ for Miller $2^n$-tupling in accordance with (2). We note that Algorithm 3 computes the product $g$ under the assumption of an even embedding degree, so that the denominator $v_i$ of the $i$-th product update $g_i = l_i/v_i$ can be eliminated and the $g_i$'s simply become the $l_i$'s described at the beginning of Section 2. In the following sections we use different projections on the affine form of $f^*$ depending on the curve model.

**Algorithm 3.** Constructing explicit formulas for $f^*$

**Input:** $R = (x_1, y_1)$ and $S = (x_S, y_S)$.
**Output:** $f^*$.

1: $(x, y) \leftarrow (x_1, y_1)$, $f^* \leftarrow 1$.
2: **for** $i = 1$ to $n$ **do**
3: $\lambda \leftarrow (3x^2 + a)/(2y)$.
4: $x' \leftarrow \lambda^2 - 2x$.
5: $y' \leftarrow \lambda(x - x') - y$.
6: $g \leftarrow \lambda(x - x_S) + y_S - y$.
7: $f^* \leftarrow f^* \cdot g^{2^{n-i}}$.
8: $(x, y) \leftarrow (x', y')$.
9: **end for**
10: **return** $f^*$.

*Line 7 of Algorithm 2:* Depending on the formulas derived for $f^*$, there are two possibilities that need to be considered for computing the point multiplication $[2^n]T$. The first option would be to output the explicit formulas for $x'$ and $y'$ in Algorithm 3. These compounded formulas would obviously be much more complicated than the standard point doubling formulas (i.e. computing $[2]T$), however the more complicated explicit formulas for computing $[2^n]T = (x', y')$ may end up sharing many common subexpressions with the explicit formula for $f^*$ so that the overall count would be less. The second option simply involves repeating $n$ consecutive doublings on the point $T$. The heuristic argument would suggest that optimized formulas for computing $[2^n]T$ directly should require no more operations than those required in the repetitive doublings, suggesting that the first option should always take preference. However, our experiments indicated that attempts to optimize $2^n$-tupling formulas always tend to reduce to the same formulas that arise from $n$ repeated doublings. For the sake of simplicity, we therefore opt for the latter suggestion and perform $n$ repetitive doublings to compute $[2^n]T$. Furthermore, it also tends to be the case that the higher degree subexpressions obtained in the explicit formulas for computing $[2^n]T$ directly do not appear in the simplified expressions for $f^*$. However, many operations used in the very first doubling of $T$ also appear readily in the components of $f^*$ and we make use of these common subexpressions. Namely, the doubling formulas used to compute $[2]T$ are chosen so that the simultaneous computation of $f^*$ and $[2]T$ comes at minimal cost. Therefore, it is often the case that the formulas used to compute $[2]T$ may not be the same formulas as those used to compute the $n - 1$ doublings that follow.

*Lines 10 and 11 of Algorithm 2:* In the addition stage of Miller $2^n$-tuple-and-add, we are required to add some multiple $w[R]$ of $R$ ($w < 2^n$) to the intermediate point $T$. Here we simply cache the value $[w]R$ before the iterations start and perform a standard point addition. The Miller function update $f^+$ required in line 7 of Algorithm 2 requires the computation of the product $f^+ = f_{w,R}(S) \cdot g_{T,[w]R}(S)$. By definition, $g_{T,[w]R}(S)$ is the line function corresponding to the

addition of $T$ to $[w]R$, evaluated at the point $S$. Therefore, the combination of lines 11 and 12 of Algorithm 2 can simply be viewed as a standard point addition between $T$ and $[w]R$, as well as the extra multiplication of $g_{T,[w]R}(S)$ by the cached value $f_{w,R}(S)$.

## 5 Miller Quadrupling and Octupling

In this section we focus on applying the generalized algorithm in Section 3 to the cases $n = 2$ and $n = 3$. We present reduced explicit formulas that arise for the Miller quadruple-and-add and Miller octuple-and-add algorithms on curves of the form $E : y^2 = x^3 + b$ ($j(E) = 0$) and $E : y^2 = x^3 + ax$ ($j(E) = 1728$), since these are the most efficient curve shapes used in practice [22]. We focus solely on the $2^n$-tupling stage of the algorithm (i.e. steps 6 and 7 in Algorithm 2), since optimized loop parameters will result in very few additions. We therefore delay any discussion of the additions until the following section.

### 5.1 Miller Quadruple-and-add

We begin by setting $n = 2$ in (3) to obtain the Miller update $f^*$ corresponding to the quadrupling of $T$ as

$$f^* = \prod_{i=1}^{2} (g_{[2^{i-1}]T,[2^{i-1}]T})^{2^{2-i}} = \left(g_{T,T}\right)^2 \cdot \left(g_{[2]T,[2]T}\right),$$

which has divisor $4(T) - ([4]T) - 3(\mathcal{O})$.

**Quadruple-and-add on $y^2 = x^3 + b$.** We obtain $f^*$ as the affine output of Algorithm 3 with $n = 2$. For curves of this form, the fastest explicit formulas for the $n = 1$ case were derived using homogeneous projective coordinates [16,17]. Our experiments[1] indicated that these coordinates also give the fastest results for $n \geq 1$, so we substitute $x_1 = X_1/Z_1$ and $y_1 = Y_1/Z_1$ into $f^*$ to obtain the projectified version, $F^*$, as

$$F^* = \alpha \cdot (L_{1,0} \cdot x_S + L_{2,0} \cdot x_S^2 + L_{0,1} \cdot y_S + L_{1,1} \cdot x_S y_S + L_{0,0}),$$

where $\alpha = -Z_1^3(X_1(X_1^3 - 8bZ_1^3) - 4Z_1(X_1^3 + bZ_1^3) \cdot x_S)^2/(64Z_1^7Y_1^5(27X_1^6 - 36X_1^3Y_1^2Z_1 + 8Y_1^4Z_1^2))$ can be eliminated to give $\hat{F}^* = F^*/\alpha$, where the $L_{i,j}$ coefficients are

$$\begin{aligned}
L_{2,0} &= -6X_1^2Z_1(5Y_1^4 + 54bY_1^2Z_1^2 - 27b^2Z_1^4),\\
L_{0,1} &= 8X_1Y_1Z_1(5Y_1^4 + 27b^2Z_1^4),\\
L_{1,1} &= 8Y_1Z_1^2(Y_1^4 + 18bY_1^2Z_1^2 - 27b^2Z_1^4),\\
L_{0,0} &= 2X_1(Y_1^6 - 75bY_1^4Z_1^2 + 27b^2Y_1^2Z_1^4 - 81b^3Z_1^6).\\
L_{1,0} &= -4Z_1(5Y_1^6 - 75bZ_1^2Y_1^4 + 135Y_1^2b^2Z_1^4 - 81b^3Z_1^6).
\end{aligned}$$

[1] We searched through a range of different coordinate systems (cf. [11]) to find the coordinate system which gave the most simple projectified line coefficients.

We let $(X_{D^n} : Y_{D^n} : Z_{D^n}) = [2^n](X_1 : Y_1 : Z_1)$ and compute the first doubling with small extra computation as

$$X_{D^1} = 4X_1Y_1(Y_1^2 - 9bZ_1^2),\ \ Y_{D^1} = 2Y_1^4 + 36bY_1^2Z_1^2 - 54b^2Z_1^4,\ \ Z_{D^1} = 16Y_1^3Z_1$$

The calculation of the $L_{i,j}$ coefficients and the intermediate point $(X_{D^1} : Y_{D^1} : Z_{D^1}) = [2](X_1, Y_1, Z_1)$ requires $11\mathbf{m}_e + 11\mathbf{s}_e + 3\mathbf{d}$. To calculate $(X_{D^2} : Y_{D^2} : Z_{D^2}) = [4](X_1, Y_1, Z_1)$, we double the point $(X_{D^1} : Y_{D^1} : Z_{D^1})$ using the doubling formulas in [17] which cost $3\mathbf{m}_e + 5\mathbf{s}_e + 1\mathbf{d}$. The multiplication of each of the four $L_{i,j} \neq L_{0,0}$ by $x_S^i y_S^j$ costs $e\mathbf{m}_1$ (cf. [17]). As discussed in Section 3, the extension field arithmetic required in line 8 of Algorithm 2 costs $1\mathbf{M} + 2\mathbf{S}$. Thus, the total cost for the quadrupling stage is $14\mathbf{m}_e + 16\mathbf{s}_e + 4e\mathbf{m}_1 + 4\mathbf{d} + 1\mathbf{M} + 2\mathbf{S}$ (see Appendix A.1 for the sequence of operations, and see Appendix B for a Magma script that computes the Miller quadruple-and-add algorithm using the formulas in A.1).

**Quadruple-and-add on $y^2 = x^3 + ax$.** For curves of this shape, the fastest formulas for the standard double-and-add case were derived in weight-$(1, 2)$ coordinates in [17]. Again, our experiments agree with these coordinates for such curves for $n \geq 1$, so we subsitute $x_1 = X_1/Z_1$ and $y_1 = Y_1/Z_1^2$ into $f^*$ (the output of Algorithm 3) to obtain $F^*$ as

$$F^* = \alpha \cdot (L_{1,0} \cdot x_S + L_{2,0} \cdot x_S^2 + L_{0,1} \cdot y_S + L_{1,1} \cdot x_S y_S + L_{0,0}),$$

where $\alpha = -Z_1^6(-4X_1Z_1(X_1^2 + aZ_1^2)x_S + (X_1^2 - aZ_1^2)^2)^2$ can be eliminated to give $\hat{F}^* = F^*/\alpha$, where the $L_{i,j}$ coefficients are

$$\begin{aligned}
L_{1,0} &= -2X_1Z_1(5X_1^8 + 4aX_1^6Z_1^2 + 38a^2X_1^4Z_1^4 + 20a^3X_1^2Z_1^6 - 3a^4Z_1^8),\\
L_{2,0} &= -Z_1^2(15X_1^8 + 68aX_1^6Z_1^2 + 10a^2X_1^4Z_1^4 - 28a^3X_1^2Z_1^6 - a^4Z_1^8),\\
L_{0,1} &= 4Y_1X_1Z_1(5X_1^6 + 13aX_1^4Z_1^2 + 15a^2X_1^2Z_1^4 - a^3Z_1^6),\\
L_{1,1} &= 4Y_1Z_1^2(X_1^2 - aZ_1^2)(X_1^4 + 6aX_1^2Z_1^2 + a^2Z_1^4),\\
L_{0,0} &= X_1^2(X_1^8 - 20aX_1^6Z_1^2 - 26a^2X_1^4Z_1^4 - 20a^3X_1^2Z_1^6 + a^4Z_1^8).
\end{aligned}$$

Again, we compute the first doubling with small extra computation as

$$X_{D^1} = (X_1^2 - aZ_1^2)^2,\ Y_{D^1} = 2Y_1(X_1^2 - aZ_1^2)(X_1^4 + 6X_1^2aZ_1^2 + a^2Z_1^4),\ Z_{D^1}{=}4Y_1^2.$$

The calculation of the $L_{i,j}$ coefficients and the intermediate point $(X_{D^1} : Y_{D^1} : Z_{D^1}) = [2](X_1, Y_1, Z_1)$ requires $10\mathbf{m} + 14\mathbf{s} + 2\mathbf{d}$. To calculate $(X_{D^2} : Y_{D^2} : Z_{D^2}) = [4](X_1, Y_1, Z_1)$, we double the point $(X_{D^1} : Y_{D^1} : Z_{D^1})$ using the doubling formulas in [17] which cost $1\mathbf{m} + 6\mathbf{s} + 1\mathbf{d}$. Thus, the total cost for the quadrupling stage is $11\mathbf{m}_e + 20\mathbf{s}_e + 4e\mathbf{m}_1 + 3\mathbf{d} + 1\mathbf{M} + 2\mathbf{S}$ (see Appendix A.2).

### 5.2 Miller Octuple-and-add

We begin by setting $n = 3$ in (3) to obtain the Miller update $f^*$ corresponding the octupling of $T$ as

$$f^* = \prod_{i=1}^{3} (g_{[2^{i-1}]T,[2^{i-1}]T})^{2^{3-i}} = \left(g_{T,T}\right)^4 \cdot \left(g_{[2]T,[2]T}\right)^2 \cdot \left(g_{[4]T,[4]T}\right),$$

which has divisor $8(T) - ([8]T) - 7(\mathcal{O})$.

**Octuple-and-add on $y^2 = x^3 + b$.** For the octupling line product, we use homogeneous projective coordinates to give $F^*$ as

$$\begin{aligned} F^* = \alpha \cdot (&L_{4,0} \cdot x_S^4 + L_{3,0} \cdot x_S^3 + L_{2,0} \cdot x_S^2 + L_{1,0} \cdot x_S \\ &+ L_{3,1} \cdot x_S^3 y_S + L_{2,1} \cdot x_S^2 y_S + L_{1,1} \cdot x_S y_S + L_{0,0}), \end{aligned}$$

where $\alpha$ is again contained in a proper subfield of $\mathbb{F}_{p^k}$ and can be eliminated to give $\hat{F}^* = F^*/\alpha$. The $L_{i,j}$ coefficients are

$$\begin{aligned} &L_{4,0} = (-9X_1^2Z_1^2) \cdot \mathcal{S}_{4,0}, \;\; L_{3,0} = (-12Z_1^2Y_1^2) \cdot \mathcal{S}_{3,0}, \;\; L_{2,0} = (-54X_1Y_1^2Z_1) \cdot \mathcal{S}_{2,0} \\ &L_{1,0} = (-36X_1^2Y_1^2) \cdot \mathcal{S}_{1,0}, \;\; L_{0,0} = ((Y_1^2 + 3bZ_1^2)Y_1^2) \cdot \mathcal{S}_{0,0}, \;\; L_{3,1} = (8Y_1Z_1^3) \cdot \mathcal{S}_{3,1} \\ &L_{2,1} = (216X_1Y_1Z_1^2) \cdot \mathcal{S}_{2,1}, \;\; L_{1,1} = (72X_1^2Y_1Z_1) \cdot \mathcal{S}_{1,1}, \;\; L_{0,1} = (8Y_1^3Z_1) \cdot \mathcal{S}_{0,1} \end{aligned}$$

with

$$\mathcal{S}_{i,j} = \sum_{k=0}^{11} c_{i,j,k} \cdot (Y_1^2)^{11-k} (bZ_1^2)^k,$$

where $c_{i,j,k}$ is the coefficient of $(Y_1^2)^{11-k}(bZ_1^2)^k$ belonging to $L_{i,j}$ (see Appendix A.3). As an example, we have

$$\begin{aligned} L_{0,0} = (Y_1^2(Y_1^2 + 3bZ_1^2)) \cdot \big(&Y_1^{22} - 3375bY_1^{20}Z_1^2 - 262449b^2Y_1^{18}Z_1^4 \\ &- 2583657b^3Y_1^{16}Z_1^6 + 47678058b^4Y_1^{14}Z_1^8 - 40968342b^5Y_1^{12}Z_1^{10} \\ &- 272740770b^6Y_1^{10}Z_1^{12} + 738702990b^7Y_1^8Z_1^{14} - 669084219b^8Y_1^6Z_1^{16} \\ &- 23914845b^{10}Y_1^2Z_1^{20} + 14348907b^{11}Z_1^{22} + 206730549b^9Y_1^4Z_1^{18}\big). \end{aligned}$$

We describe a general method to compute each of the terms of the form $(Y_1^2)^{11-k}(bZ_1^2)^k$ that are required to compute the $L_{i,j}$ coefficients, where $0 \leq k \leq 11$. In general, it is best to compute each one of these products rather than attempting to factorize, particular when each of these terms is present in every $L_{i,j}$. We compute every required even power of $Y_1$ by first repetitively squaring $Y_1$ until we have all necessary terms of the form $Y_1^{2^t}$ that are less than the largest power of $Y_1$ occuring in the summations of the $L_{i,j}$. That is, we compute $Y_1^{2^t}$ for $t = 1, 2, 3, 4$ since $Y_1^{22}$ is the largest power of $Y_1$ occuring in the $L_{i,j}$ summations. Using $\{Y_1^2, Y_1^4, Y_1^8, Y_1^{16}\}$, we can compute all other $(Y_1^2)^z < (Y_1^2)^{16}$, $z \neq 2^t$ using one squaring each for each $z$. For example, we can compute $Y_1^{12}$ as $Y_1^{12} = Y_1^8 \cdot Y_1^4 = ((Y_1^8 + Y_1^4)^2 - Y_1^{16} - Y_1^8)/2$, although in practice we compute

$2Y_1^{12}$ to avoid the division by 2. To compute the remaining $(Y_1^2)^t > Y_1^{16}$, we use a field multiplication[2]. We do the same for each of the $(bZ_1^2)^k$ terms.

We do not count multiplications by the $c_{i,j,k}$, although we make no attempt to disguise the extra cost that is incurred as their sizes grow. We do however, point out that it is often the case that the $c_{i,j,k}$'s for a fixed $k$ (but different $i$,$j$'s) share large common factors so that we need not multiply $(Y_1^2)^{11-k}(bZ_1^2)^k$ by each of the $c_{i,j,k}$'s, but rather we combine previous products to obtain most of these multiplications at a much smaller (mostly negligible) cost.

The total operation count for the point octupling and the computation of the octupling line product is $40\mathbf{m}_e + 31\mathbf{s}_e + 8e\mathbf{m}_1 + 2\mathbf{d} + 1\mathbf{M} + 3\mathbf{S}$ (see Appendix A.3).

**Octuple-and-add on $y^2 = x^3 + ax$.** Following the trend of the fastest formulas for the $n = 1$ and $n = 2$ cases for curves of this shape, we again projectify $f^*$ using weight-$(1, 2)$ coordinates to give

$$F^* = \alpha \cdot (L_{4,0} \cdot x_S^4 + L_{3,0} \cdot x_S^3 + L_{2,0} \cdot x_S^2 + L_{1,0} \cdot x_S + L_{3,1} \cdot x_S^3 y_S + L_{2,1} \cdot x_S^2 y_S + L_{1,1} \cdot x_S y_S + L_{0,0}),$$

where we ignore the subfield cofactor $\alpha$ to give $\hat{F}^* = F^*/\alpha$. The $L_{i,j}$ coefficients are given as

$$L_{4,0} = (-4X_1^2Z_1^4) \cdot \mathcal{S}_{4,0}, \quad L_{3,0} = (-16X_1^3Z_1^3) \cdot \mathcal{S}_{3,0}, \quad L_{2,0} = (-8X_1^4Z_1^2) \cdot \mathcal{S}_{2,0}$$
$$L_{1,0} = (16X_1^5Z_1) \cdot \mathcal{S}_{1,0}, \quad L_{0,0} = (4X_1^6) \cdot \mathcal{S}_{0,0}, \quad L_{3,1} = (4Y_1Z_1^4) \cdot \mathcal{S}_{3,1}$$
$$L_{2,1} = (4X_1Y_1Z_1^3) \cdot \mathcal{S}_{2,1}, \quad L_{1,1} = (4X_1^2Y_1Z_1^2) \cdot \mathcal{S}_{1,1}, \quad L_{0,1} = (4X_1^3Y_1Z_1) \cdot \mathcal{S}_{0,1},$$

with

$$\mathcal{S}_{i,j} = \sum_{k=0}^{16} c_{i,j,k} \cdot (X_1^2)^{16-k}(bZ_1^2)^k,$$

where $c_{i,j,k}$ is the coefficient of $(X_1^2)^{16-k}(bZ_1^2)^k$ belonging to $L_{i,j}$ (see Appendix A.4). As an example, we have

$$\begin{aligned} L_{2,0} = -8X1^4Z1^2 \cdot (&189X_1^{32} + 882bX_1^{30}Z_1^2 + 6174b^2X_1^{28}Z_1^4 - 26274b^3X_1^{26}Z_1^6 \\ &- 1052730b^4X_1^{24}Z_1^8 - 449598b^5X_1^{22}Z_1^{10} - 1280286b^6X_1^{20}Z_1^{12} \\ &- 1838850b^7X_1^{18}Z_1^{14} - 23063794b^8X_1^{16}Z_1^{16} - 1543290b^9X_1^{14}Z_1^{18} \\ &+ 539634b^{10}X_1^{12}Z_1^{20} + 646922b^{11}X_1^{10}Z_1^{22} + 1386918b^{12}X_1^8Z_1^{24} \\ &+ 75846b^{13}X_1^6Z_1^{26} + 17262b^{14}X_1^4Z_1^{28} + 922b^{15}X_1^2Z_1^{30} - 35b^{16}Z_1^{32}). \end{aligned}$$

The total operation count for the point octupling and the computation of the octupling line product is $31\mathbf{m}_e + 57\mathbf{s}_e + 8e\mathbf{m}_1 + 5\mathbf{d} + 1\mathbf{M} + 3\mathbf{S}$ (see Appendix A.4).

[2] We point out that if higher degree terms also required computation it may be advantageous to compute $Y_1^{32}$ so that each of the terms $(Y_1^2)^t > Y_1^{16}$ can be computed using field squarings instead of multiplications. This advantage would depend on the platform (the $\mathbf{s}$:$\mathbf{m}$ ratio) and the number of $(Y_1^2)^t > Y_1^{16}$ terms required.

## 6 Comparisons

We draw comparisons between 6 standard loops of Miller double-and-add, 3 standard loops of Miller quadruple-and-add and 2 standard loops of Miller octuple-and-add, since each of these equates to one 64-tuple-and-add loop, and this is the most primitive level at which a fair comparison can be made. We note that the estimated percentage speedups in Table 1 are for the computation of the Miller loop only and do not take into account the significant fixed cost of final exponentiation. We neglect additions since low hamming-weight loop parameters used in pairing implementations will result in a similar amount of additions regardless of $n$, and we saw in sections 3 and 4 that additions come at approximately the same cost for different $n$. The counts for $n = 1$ are due to the fastest formulas given for curves with $j(E) = 0$ and $j(E) = 1728$ in [17]. We multiply these counts and those obtained for $n = 2$ and $n = 3$ in Section 5 accordingly.

**Table 1.** Operation counts for the equivalent number of iterations of $2^n$-tuple and add for $n = 1, 2, 3$

| $j(E)$ | Doubling: $n = 1$ (6 loops) | Quadrupling: $n = 2$ (3 loops) | Octupling: $n = 3$ (2 loops) |
|---|---|---|---|
| 0 | $12\mathbf{m}_e + 42\mathbf{s}_e + 12e\mathbf{m}_1$ $+6\mathbf{M} + 6\mathbf{S}$ | $42\mathbf{m}_e + 48\mathbf{s}_e + 12e\mathbf{m}_1$ $+3\mathbf{M} + 6\mathbf{S}$ | $80\mathbf{m}_e + 64\mathbf{s}_e + 16e\mathbf{m}_1$ $+2\mathbf{M} + 6\mathbf{S}$ |
| 1728 | $12\mathbf{m}_e + 48\mathbf{s}_e + 12e\mathbf{m}_1$ $+6\mathbf{M} + 6\mathbf{S}$ | $33\mathbf{m}_e + 60\mathbf{s}_e + 12e\mathbf{m}_1$ $+3\mathbf{M} + 6\mathbf{S}$ | $64\mathbf{m}_e + 114\mathbf{s}_e + 16e\mathbf{m}_1$ $+2\mathbf{M} + 6\mathbf{S}$ |

Table 1 shows that the number of subfield operations increases when $n$ gets larger, whilst the number of full extension field multiplications decreases. To determine whether these trade-offs become favorable for $n = 2$ or $n = 3$, we adopt the standard procedure of estimating the equivalent number of base field operations for each operation count [27,17]. We assume that the higher degree fields are constructed as a tower of extensions, so that for pairing-friendly fields of extension degree $z = 2^i \cdot 3^j$, we can assume that $\mathbf{m}_z = 3^i \cdot 5^j \mathbf{m}_1$ [31]. We split the comparison between pairings on $\mathbb{G}_1 \times \mathbb{G}_2$ (the Tate pairing, the twisted ate pairing) and pairings on $\mathbb{G}_2 \times \mathbb{G}_1$ (the ate pairing, R-ate pairing, etc). For each pairing-friendly embedding degree reported, we assume that the highest degree twist is utilized in both settings; the curves with $j(E) = 0$ utilize degree 6 twists whilst the curves with $j(E) = 1728$ utilize degree 4 twists. To compare across operations, we follow the EFD [11] and present two counts in each scenario: the top count assumes that $\mathbf{s}_z = \mathbf{m}_z$, whilst the bottom count assumes that $\mathbf{s}_z = 0.8\mathbf{m}_z$. When quadrupling ($n = 2$) or octupling ($n = 3$) gives a faster operation count, we provide an approximate percentage speedup for the computation of the Miller loop, ignoring any additions that occur.

Unsurprisingly, Table 2 illustrates that the relative speed up for pairings on $\mathbb{G}_1 \times \mathbb{G}_2$ grows as the embedding degree grows. This is due to the increasing gap between the complexity of operations in $\mathbb{G}_1$ (which is defined over $\mathbb{F}_q$) and $\mathbb{G}_2$

**Table 2.** Comparisons for Miller double-and-add, Miller quadruple-and-add and Miller octuple-and-add at various embedding degrees

| | | Pairings on $\mathbb{G}_1 \times \mathbb{G}_2$ (Tate, twisted ate) | | | Best (%) | Pairings on $\mathbb{G}_2 \times \mathbb{G}_1$ (ate, R-ate) | | | Best (%) |
|---|---|---|---|---|---|---|---|---|---|
| $k$ | $j(E)$ | $n=1$ (6 loops) | $n=2$ (3 loops) | $n=3$ (2 loops) | | $n=1$ (6 loops) | $n=2$ (3 loops) | $n=3$ (2 loops) | |
| 4 | 1728 | 180<br>159.6 | 186<br>163.2 | 266<br>232.4 | $n=1$<br>- | 180<br>159.6 | 186<br>163.2 | 266<br>232.4 | $n=1$<br>- |
| 6 | 0 | 246<br>219.6 | 237<br>209.4 | 280<br>249.2 | $n=2$<br>5% | 246<br>219.6 | 237<br>209.4 | 280<br>249.2 | $n=2$<br>5% |
| 8 | 1728 | 408<br>366 | 360<br>315.6 | 426<br>370.8 | $n=2$<br>14% | 528<br>466.8 | 546<br>477.6 | 782<br>681.2 | $n=1$<br>- |
| 12 | 0 | 618<br>555.6 | 519<br>455.4 | 536<br>469.2 | $n=2$<br>18% | 726<br>646.8 | 699<br>616.2 | 824<br>731.6 | $n=2$<br>5% |
| 16 | 1728 | 1080<br>973.2 | 870<br>760.8 | 890<br>770 | $n=2$<br>22% | 1560<br>1376.4 | 1614<br>1408.8 | 2314<br>2011.6 | $n=1$<br>- |
| 18 | 0 | 990<br>891.6 | 801<br>701.4 | 792<br>689.2 | $n=3$<br>22% | 1206<br>1074 | 1161<br>1023 | 1368<br>1214 | $n=2$<br>5% |
| 24 | 0 | 1722<br>1551.6 | 1353<br>1181.4 | 1288<br>1113.2 | $n=3$<br>28% | 2154<br>1916.4 | 2073<br>1824.6 | 2440<br>2162.8 | $n=2$<br>5% |
| 32 | 1728 | 3072<br>2770.8 | 2376<br>2072.4 | 2250<br>1935.6 | $n=3$<br>30% | 4632<br>4081.2 | 4794<br>4178.4 | 6878<br>5970.8 | $n=1$<br>- |
| 36 | 0 | 2826<br>2547.6 | 2187<br>1907.4 | 2040<br>1757.6 | $n=3$<br>31% | 3582<br>3186 | 3447<br>3033 | 4056<br>3594 | $n=2$<br>5% |
| 48 | 0 | 5010<br>4515.6 | 3831<br>3335.4 | 3512<br>3013.2 | $n=3$<br>33% | 6414<br>5701.2 | 6171<br>5425.8 | 7256<br>6424.4 | $n=2$<br>5% |

(which is defined over $\mathbb{F}_{q^k}$). In this case we see that $6 \leq k \leq 16$ favor Miller quadruple-and-add, whilst Miller octuple-and-add takes over for $k > 16$, where it is clear that it is worthwhile spending many more operations in the base field in order to avoid costly arithmetic in $\mathbb{F}_{q^k}$. For pairings on $\mathbb{G}_2 \times \mathbb{G}_1$, we have a consistent speed up across all embedding degrees that utilize sextic twists. This is due to the complexity of the subfield operations in $\mathbb{F}_{q^e}$ growing at the same rate as the complexity of operations in $\mathbb{F}_{q^k}$. Table 2 indicates that Miller double-and-add is still preferred for ate-like pairings using quartic twists, where we could conclude that the gap between operations in $\mathbb{F}_{q^{k/4}}$ and those in $\mathbb{F}_{q^k}$ isn't large enough to favor higher Miller tupling.

The large improvements in Table 2 certainly present a case for the investigation of higher degree Miller tupling ($n \geq 4$). At these levels however, the formulas become quite complex and we have not reported any discoveries from these degrees due to space considerations. Namely, the size of the $2^n$-tupling line in (2) grows exponentially as $n$ increases (i.e. the degree of the affine $2^n$-tupling line formula is twice that of the $2^{n-1}$-tupling line). The fact that quadrupling was still preferred over octupling in most cases seems to suggest that larger $n$ might not result in significant savings, at least for embedding degrees of this size.

We conclude by acknowledging that (in optimal implementations) the speedups in Table 2 may not be as large as we have claimed. In generating the comparisons, we reported the multiplication of the intermediate Miller value $f$ by the Miller update $g$ as a full extension field multiplication in $\mathbb{F}_{p^k}$, with complexity $\mathbf{M} = \mathbf{m}_k = 3^i \cdot 5^j$ for $k = 2^i \cdot 3^j$. Although the value $f$ is a general full extension field element, $g$ tends to be sparse, especially when sextic twists are employed. For even degree twists, $g$ takes the form $g = g_1\alpha + g_2\beta + g_0$, where $g \in F_{p^k}$, $g_0, g_1, g_2 \in \mathbb{F}_{p^{k/d}}$ and $\alpha$ and $\beta$ are algebraic elements that do not affect multiplication costs (cf. [17]). For sextic twists, a general element of $\mathbb{F}_{p^k}$ would be written as a polynomial over $\mathbb{F}_{p^e}$ with six (rather than three) different coefficients belonging to $\mathbb{F}_{p^{k/6}}$. In this case, multiplying two general elements of $\mathbb{F}_{p^k}$ would clearly require more multiplications than performing a multiplication between a general element (like $f$) and a sparse element (like $g$). Since the techniques in this paper gain advantage by avoiding multiplications between $f$ and $g$, reporting a lesser complexity for this multiplication would decrease the relative speedup. Nevertheless, Miller quadruple-and-add and Miller octuple-and-add still strongly outperform the standard Miller double-and-add routine if we take $\mathbf{m}_k \ll 3^i \cdot 5^j$, particularly for pairings on $\mathbb{G}_1 \times \mathbb{G}_2$ with large embedding degrees.

## Acknowledgements

The authors wish to thank Huseyin Hisil, Douglas Stebila, Nicole Verna, and the anonymous referees for helpful comments and suggestions on earlier versions of this paper.

## References

1. Arene, C., Lange, T., Naehrig, M., Ritzenthaler, C.: Faster pairing computation. Cryptology ePrint Archive, Report 2009/155 (2009), http://eprint.iacr.org/
2. Avanzi, R.M., Cohen, H., Doche, C., Frey, G., Lange, T., Nguyen, K., Vercauteren, F.: The Handbook of Elliptic and Hyperelliptic Curve Cryptography. CRC, Boca Raton (2005)
3. Barreto, P.S.L.M., Galbraith, S.D., O'Eigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular abelian varieties. Des. Codes Cryptography 42(3), 239–271 (2007)
4. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
5. Barreto, P.S.L.M., Lynn, B., Scott, M.: Constructing elliptic curves with prescribed embedding degrees. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 257–267. Springer, Heidelberg (2003)
6. Barreto, P.S.L.M., Lynn, B., Scott, M.: On the selection of pairing-friendly groups. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 17–25. Springer, Heidelberg (2004)

7. Barreto, P.S.L.M., Lynn, B., Scott, M.: Efficient implementation of pairing-based cryptosystems. J. Cryptology 17(4), 321–334 (2004)
8. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S.E. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
9. Benger, N., Scott, M.: Constructing tower extensions for the implementation of pairing-based cryptography. Cryptology ePrint Archive, Report 2009/556 (2009), http://eprint.iacr.org/
10. Benits Jr., W.D., Galbraith, S.D.: Constructing pairing-friendly elliptic curves using Gröbner basis reduction. In: Galbraith, S.D. (ed.) [23], pp. 336–345
11. Bernstein, D.J., Lange, T.: Explicit-formulas database, http://www.hyperelliptic.org/EFD
12. Blake, I.F., Kumar Murty, V., Xu, G.: Refinements of miller's algorithm for computing the weil/tate pairing. J. Algorithms 58(2), 134–149 (2006)
13. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
14. Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Des. Codes Cryptography 37(1), 133–141 (2005)
15. Chatterjee, S., Sarkar, P., Barua, R.: Efficient computation of Tate pairing in projective coordinate over general characteristic fields. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 168–181. Springer, Heidelberg (2005)
16. Costello, C., Hisil, H., Boyd, C., Nieto, J.M.G., Wong, K.K.-H.: Faster pairings on special Weierstrass curves. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 92–109. Springer, Heidelberg (2009)
17. Costello, C., Lange, T., Naehrig, M.: Faster pairing computations on curves with high-degree twists. In: PKC 2010. LNCS. Springer, Heidelberg (to appear, 2010)
18. Prem Laxman Das, M., Sarkar, P.: Pairing computation on twisted Edwards form elliptic curves. In: Galbraith, S.D., Paterson, K.G. (eds.) [25], 192–210
19. Duursma, I.M., Lee, H.-S.: Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 111–123. Springer, Heidelberg (2003)
20. Freeman, D.: Constructing pairing-friendly elliptic curves with embedding degree 10. In: Hess, F., Pauli, S., Pohst, M.E. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 452–465. Springer, Heidelberg (2006)
21. Freeman, D.: A generalized Brezing-Weng algorithm for constructing pairing-friendly ordinary abelian varieties. In: Galbraith, S.D., Paterson, K.G. (eds.) [25], pp. 146–163
22. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. J. Cryptology 23(2), 224–280 (2010)
23. Galbraith, S.D. (ed.): Cryptography and Coding 2007. LNCS, vol. 4887. Springer, Heidelberg (2007)
24. Galbraith, S.D., McKee, J.F., Valença, P.C.: Ordinary abelian varieties having small embedding degree. Finite Fields and their Applications 13, 800–814 (2007)
25. Galbraith, S.D., Paterson, K.G. (eds.): Pairing 2008. LNCS, vol. 5209. Springer, Heidelberg (2008)
26. Hess, F.: Pairing lattices. In: Galbraith, S.D., Paterson, K.G. (eds.) [25], pp. 18–38

27. Hess, F., Smart, N.P., Vercauteren, F.: The eta pairing revisited. IEEE Transactions on Information Theory 52(10), 4595–4602 (2006)
28. Ionica, S., Joux, A.: Another approach to pairing computation in Edwards coordinates. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 400–413. Springer, Heidelberg (2008), http://eprint.iacr.org/2008/292
29. Joye, M. (ed.): CT-RSA 2003. LNCS, vol. 2612. Springer, Heidelberg (2003)
30. Kachisa, E.J., Schaefer, E.F., Scott, M.: Constructing Brezing-Weng pairing-friendly elliptic curves using elements in the cyclotomic field. In: Galbraith, S.D., Paterson, K.G. (eds.) [25], pp. 126–135
31. Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 13–36. Springer, Heidelberg (2005)
32. Lee, E., Lee, H.-S., Park, C.-M.: Efficient and generalized pairing computation on abelian varieties. IEEE Transactions on Information Theory 55(4), 1793–1803 (2009)
33. Matsuda, S., Kanayama, N., Hess, F., Okamoto, E.: Optimised versions of the ate and twisted ate pairings. In: Galbraith, S.D. (ed.) [23], pp. 302–312
34. Miller, V.S.: The Weil pairing, and its efficient calculation. Journal of Cryptology 17, 235–261 (2004)
35. Scott, M.: Computing the Tate pairing. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 293–304. Springer, Heidelberg (2005)
36. Scott, M., Barreto, P.S.L.M.: Generating more MNT elliptic curves. Des. Codes Cryptography 38(2), 209–217 (2006)
37. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory 56(1), 455–461 (2010)

# A Explicit Formulas

In each of the following four scenarios, we provide the sequence of operations required to compute the first point doubling and the $2^n$-tupling line function, followed by the additional formulae required to compute the subsequent point doublings.

## A.1 Quadrupling Formulas for $y^2 = x^3 + b$

$$A = Y_1^2,\ B = Z_1^2,\ C = A^2,\ D = B^2,\ E = (Y_1 + Z_1)^2 - A - B,\ F = E^2,\ G = X_1^2,\ H = (X_1 + Y_1)^2 - A - G,$$
$$I = (X_1 + E)^2 - F - G,\ J = (A + E)^2 - C - F,\ K = (Y_1 + B)^2 - A - D,\ L = 27b^2D,\ M = 9bF,\ N = A \cdot C,$$
$$R = A \cdot L,\ S = bB,\ T = S \cdot L,\ U = S \cdot C,\ X_{D1} = 2H \cdot (A - 9S),\ Y_{D1} = 2C + M - 2L,\ Z_{D1} = 4J,$$
$$L_{1,0} = -4Z_1 \cdot (5N + 5R - 3T - 75U),\ L_{2,0} = -3G \cdot Z_1 \cdot (10C + 3M - 2L),\ L_{0,1} = 2I \cdot (5C + L),$$
$$L_{1,1} = 2K \cdot Y_{D1},\ L_{0,0} = 2X_1 \cdot (N + R - 3T - 75U).$$
$$F^* = L_{1,0} \cdot x_S + L_{2,0} \cdot x_S^2 + L_{0,1} \cdot y_S + L_{1,1} \cdot x_S y_S + L_{0,0}, A_2 = Y_{D1}^2,\ B_2 = Z_{D1}^2,\ C_2 = 3bB_2,$$
$$D_2 = 2X_{D1} \cdot Y_{D1},\ E_2 = (Y_{D1} + Z_{D1})^2 - A_2 - B_2,\ F_2 = 3C_2,\ X_{D2} = D_2 \cdot (A_2 - F_2),$$
$$Y_{D2} = (A_2 + F_2)^2 - 12C_2^2,\ Z_{D2} = 4A_2 \cdot E_2.$$

The above sequence of operations costs $14\mathbf{m}_e + 16\mathbf{s}_e + 4e\mathbf{m}_1$.

## A.2 Quadrupling Formulas for $y^2 = x^3 + ax$

$A = X_1^2,\ B = Y_1^2,\ C = Z_1^2,\ D = aC,\ X_{D1} = (A - D)^2,\ E = 2(A + D)^2 - X_{D1},$
$F = ((A - D + Y_1)^2 - B - X_{D1}),\ Y_{D1} = E \cdot F,\ Z_{D1} = 4B,\ G = A^2,\ H = D^2,\ I = G^2,\ J = H^2,$
$K = (X_1 + Z_1)^2 - A - C,\ L = K^2,\ M = (Y_1 + K)^2 - L - B,\ N = ((G + H)^2 - I - J),\ R = aL,\ S = R \cdot G,$
$T = R \cdot H,\ L_{1,1} = 2C \cdot Y_{D1},\ L_{0,1} = M \cdot (5A \cdot (G + 3H) + D \cdot (13G - H)),$
$L_{2,0} = -C \cdot (15I + 17S + 5N - 7T - J),\ L_{1,0} = -K \cdot (5I + S + 19N + 5T - 3J),$
$L_{0,0} = A \cdot (I - 5S - 13N - 5T + J).\ F^* = L_{1,0} \cdot x_S + L_{2,0} \cdot x_S^2 + L_{0,1} \cdot y_S + L_{1,1} \cdot x_S y_S + L_{0,0}, A_2 = X_{D1}^2,$
$B_2 = Y_{D1}^2,\ C_2 = Z_{D1}^2,\ D_2 = aC_2,\ X_{D2} = (A_2 - D_2)^2,\ E_2 = 2(A_2 + D_2)^2 - X_{D2},\ Z_{D2} = 4B_2,$
$F_2 = ((A_2 - D_2 + Y_{D1})^2 - B_2 - X_{D2},\ Y_{D2} = E_2 \cdot F_2.$

The above sequence of operations costs $11\mathbf{m}_e + 20\mathbf{s}_e + 4e\mathbf{m}_1$.

## A.3 Octupling Formulas for $y^2 = x^3 + b$

$Y_{1,2} = Y_1^2,\ Z_{1,s} = Z_1^2,\ Z_{1,2} = bZ_{1,s},\ Z_{1,s^2} = Z_{1,s}^2\ A = X_1^2,\ B = b^2 Z_{1,s^2}\ C = (X_1 + Y_1)^2 - A - Y_{1,2},$
$D = (Y_1 + Z_1)^2 - Y_{1,2} - Z_{1,s},\ E = 9Z_{1,2},\ X_{D1} = C \cdot (Y_{1,2} - E),\ Y_{D1} = (Y_{1,2} + E)^2 - 108B,$
$Z_{D1} = 4Y_{1,2} \cdot D, Y_{1,4} = Y_{1,2}^2,\ Y_{1,8} = Y_{1,4}^2,\ Y_{1,16} = Y_{1,8}^2,\ Y_{1,6} = (Y_{1,2} + Y_{1,4})^2 - Y_{1,4} - Y_{1,8},$
$Y_{1,10} = (Y_{1,8} + Y_{1,2})^2 - Y_{1,16} - Y_{1,4},\ Y_{1,12} = (Y_{1,8} + Y_{1,4})^2 - Y_{1,16} - Y_{1,8},$
$Y_{1,14} = (Y_{1,8} + Y_{1,6})^2 - Y_{1,16} - 2Y_{1,12},\ Y_{1,18} = Y_{1,16} \cdot Y_{1,2},\ Y_{1,20} = Y_{1,16} \cdot Y_{1,4},$
$Y_{1,22} = Y_{1,16} \cdot Y_{1,6},\ Z_{1,4} = B,\ Z_{1,8} = Z_{1,4}^2,\ Z_{1,16} = Z_{1,8}^2,\ Z_{1,6} = (Z_{1,2} + Z_{1,4})^2 - Z_{1,4} - Z_{1,8},$
$Z_{1,10} = (Z_{1,8} + Z_{1,2})^2 - Z_{1,16} - Z_{1,4},\ Z_{1,12} = (Z_{1,8} + Z_{1,4})^2 - Z_{1,16} - Z_{1,8},$
$Z_{1,14} = (Z_{1,8} + Z_{1,6})^2 - Z_{1,16} - 2Z_{1,12},\ Z_{1,18} = Z_{1,16} \cdot Z_{1,2},\ Z_{1,20} = Z_{1,16} \cdot Z_{1,4},\ Z_{1,22} = Z_{1,16} \cdot Z_{1,6},$
$C_0^{YZ} = Y_{1,22},\ C_1^{YZ} = Y_{1,20} \cdot Z_{1,2},\ C_2^{YZ} = Y_{1,18} \cdot Z_{1,4},\ C_3^{YZ} = Y_{1,16} \cdot Z_{1,6},\ C_4^{YZ} = Y_{1,14} \cdot Z_{1,8},$
$C_5^{YZ} = Y_{1,12} \cdot Z_{1,10},\ C_6^{YZ} = Y_{1,10} \cdot Z_{1,12},\ C_7^{YZ} = Y_{1,8} \cdot Z_{1,14},\ C_8^{YZ} = Y_{1,6} \cdot Z_{1,16},$
$C_9^{YZ} = Y_{1,4} \cdot Z_{1,18},\ C_{10}^{YZ} = Y_{1,2} \cdot Z_{1,20},\ C_{11}^{YZ} = Z_{1,22},\ F = A \cdot Z_{1,s},\ G = (Y_{1,2} + Z_{1,s})^2 - Y_{1,4} - Z_{1,s^2},$
$H = C \cdot D,\ I = C^2\ J = Y_{1,2} \cdot (Y_{1,2} + 3Z_{1,2}),\ K = D \cdot Z_{1,s},\ L = C \cdot Z_{1,s},\ M = A \cdot D,\ N = Y_{1,2} \cdot D,$

$$L_{4,0} = -18F \cdot (-9565938C_{10}^{YZ} + 95659380C_9^{YZ} - 101859525C_8^{YZ} + 14880348C_7^{YZ} + 57100383C_6^{YZ} - 52396146C_5^{YZ} + 14332383C_4^{YZ} - 4578120C_3^{YZ} - 513162C_2^{YZ} + 15732C_1^{YZ} + 7C_0^{YZ}),$$

$$L_{3,0} = -12G \cdot (-14348907C_{11}^{YZ} + 239148450C_{10}^{YZ} - 643043610C_9^{YZ} + 350928207C_8^{YZ} - 60407127C_7^{YZ} - 8575227C_6^{YZ} - 7841853C_5^{YZ} + 12011247C_4^{YZ} - 3847095C_3^{YZ} - 1325142C_2^{YZ} + 56238C_1^{YZ} + 35C_0^{YZ}),$$

$$L_{2,0} = -27H \cdot (-54206982C_{10}^{YZ} + 157660830C_9^{YZ} - 120282813C_8^{YZ} + 50368797C_7^{YZ} - 25747551C_6^{YZ} + 10693215C_5^{YZ} - 3826845C_4^{YZ} + 777789C_3^{YZ} + 35682C_2^{YZ} + 4102C_1^{YZ} + 7C_0^{YZ} + 4782969C_{11}^{YZ}),$$

$$L_{1,0} = -18I \cdot (-4782969C_{11}^{YZ} + 28697814C_{10}^{YZ} - 129317310C_9^{YZ} + 130203045C_8^{YZ} - 48479229C_7^{YZ} + 11593287C_6^{YZ} - 619407C_5^{YZ} + 1432485C_4^{YZ} - 883197C_3^{YZ} + 32814C_2^{YZ} - 1318C_1^{YZ} + C_0^{YZ}),$$

$$L_{0,0} = 2J \cdot (14348907C_{11}^{YZ} - 47829690C_{10}^{YZ} + 413461098C_9^{YZ} - 669084219C_8^{YZ} + 369351495C_7^{YZ} - 136370385C_6^{YZ} - 20484171C_5^{YZ} + 23839029C_4^{YZ} - 2583657C_3^{YZ} - 524898C_2^{YZ} - 6750C_1^{YZ} + C_0^{YZ}),$$

$$L_{3,1} = 8K \cdot (-28697814C_{10}^{YZ} + 95659380C_9^{YZ} - 61115715C_8^{YZ} + 6377292C_7^{YZ} + 19033461C_6^{YZ} - 14289858C_5^{YZ} + 3307473C_4^{YZ} - 915624C_3^{YZ} - 90558C_2^{YZ} + 2484C_1^{YZ} + C_0^{YZ}),$$

$$L_{2,1} = 216L \cdot (3188646C_{10}^{YZ} - 7085880C_9^{YZ} + 4546773C_8^{YZ} - 3779136C_7^{YZ} + 5084775C_6^{YZ} - 3601260C_5^{YZ} + 1192077C_4^{YZ} - 363744C_3^{YZ} - 56610C_2^{YZ} + 1960C_1^{YZ} + C_0^{YZ}),$$

$$L_{1,1} = 72M \cdot (-9565938C_{10}^{YZ} + 10628820C_9^{YZ} - 11160261C_8^{YZ} + 20549052C_7^{YZ} - 24360993C_6^{YZ} + 11674206C_5^{YZ} - 2214945C_4^{YZ} + 434808C_3^{YZ} - 112266C_2^{YZ} + 8148C_1^{YZ} + 7C_0^{YZ}),$$

$$L_{0,1} = 8N \cdot (-14348907C_{11}^{YZ} + 28697814C_{10}^{YZ} - 77590386C_9^{YZ} + 208856313C_8^{YZ} - 152208639C_7^{YZ} + 87333471C_6^{YZ} - 19135521C_5^{YZ} + 543105C_4^{YZ} - 2329479C_3^{YZ} + 508302C_2^{YZ} - 4138C_1^{YZ} + 21C_0^{YZ}),$$
$$F^* = \alpha \cdot (L_{4,0} \cdot x_S^4 + L_{3,0} \cdot x_S^3 + L_{2,0} \cdot x_S^2 + L_{1,0} \cdot x_S + L_{3,1} \cdot x_S^3 y_S + L_{2,1} \cdot x_S^2 y_S + L_{1,1} \cdot x_S y_S + L_{0,0}),$$
$$A_2 = Y_{D1}^2,\ B_2 = Z_{D1}^2,\ C_2 = 3bB_2,\ D_2 = 2X_{D1} \cdot Y_{D1},\ E_2 = (Y_{D1} + Z_{D1})^2 - A_2 - B_2,\ F_2 = 3C_2,$$
$$X_{D2} = D_2 \cdot (A_2 - F_2),\ Y_{D2} = (A_2 + F_2)^2 - 12C_2^2,\ Z_{D2} = 4A_2 \cdot E_2.$$

$$A_3 = Y_{D2}^2,\ B_3 = Z_{D2}^2,\ C_3 = 3bB_3,\ D_3 = 2X_{D2} \cdot Y_{D2},\ E_3 = (Y_{D2} + Z_{D2})^2 - A_3 - B_3,\ F_3 = 3C_3,$$
$$X_{D3} = D_3 \cdot (A_3 - F_3),\ Y_{D3} = (A_3 + F_3)^2 - 12C_3^2,\ Z_{D3} = 4A_3 \cdot E_3.$$

The above sequence of operations costs $40\mathbf{m}_e + 32\mathbf{s}_e + 8e\mathbf{m}_1$.

## A.4 Octupling Formulas for $y^2 = x^3 + ax$

$$X_{1,2} = X_1^2,\ B = Y_1^2,\ Z_{1,s} = Z_1^2,\ Z_{1,2} = aZ_{1,s},\ X_{D1} = (X_{1,2} - Z_{1,2})^2,\ E = 2(X_{1,2} + Z_{1,2})^2 - X_{D1},$$
$$F = (X_{1,2} - Z_{1,2} + Y_1)^2 - B - X_{D1},\ Y_{D1} = E \cdot F,\ Z_{D1} = 4B,\ Z_{1,s^2} = Z_{1,s}^2,\ Z_{1,s^4} = Z_{1,s^2}^2,\ X_{1,2} = X_1^2,$$
$$X_{1,4} = X_{1,2}^2,\ X_{1,8} = X_{1,4}^2,\ X_{1,16} = X_{1,8}^2,\ X_{1,32} = X_{1,16}^2,\ X_{1,6} = (X_{1,2} + X_{1,4})^2 - X_{1,4} - X_{1,8},$$
$$X_{1,10} = (X_{1,2} + X_{1,8})^2 - X_{1,4} - X_{1,16},\ X_{1,12} = (X_{1,4} + X_{1,8})^2 - X_{1,8} - X_{1,16},$$
$$X_{1,14} = (X_{1,8} + X_{1,6})^2 - X_{1,16} - 2X_{1,12},\ X_{1,18} = (X_{1,16} + X_{1,2})^2 - X_{1,32} - X_{1,4},$$
$$X_{1,20} = (X_{1,16} + X_{1,4})^2 - X_{1,32} - X_{1,8},\ X_{1,22} = (X_{1,16} + X_{1,6})^2 - X_{1,32} - 2X_{1,12},$$
$$X_{1,24} = (X_{1,16} + X_{1,8})^2 - X_{1,32} - X_{1,16},\ X_{1,26} = (X_{1,16} + X_{1,10})^2 - X_{1,32} - 2X_{1,20},$$
$$X_{1,28} = (X_{1,16} + X_{1,12})^2 - X_{1,32} - 2X_{1,24},\ X_{1,30} = (X_{1,16} + X_{1,14})^2 - X_{1,32} - 4X_{1,28},\ Z_{1,4} = a^2 Z_{1,s^2},$$
$$Z_{1,8} = a^4 Z_{1,s^4},\ Z_{1,16} = Z_{1,8}^2,\ Z_{1,32} = Z_{1,16}^2,\ Z_{1,6} = (Z_{1,2} + Z_{1,4})^2 - Z_{1,4} - Z_{1,8},$$
$$Z_{1,10} = (Z_{1,2} + Z_{1,8})^2 - Z_{1,4} - Z_{1,16},\ Z_{1,12} = (Z_{1,4} + Z_{1,8})^2 - Z_{1,8} - Z_{1,16},$$
$$Z_{1,14} = (Z_{1,8} + Z_{1,6})^2 - Z_{1,16} - 2Z_{1,12},\ Z_{1,18} = (Z_{1,16} + Z_{1,2})^2 - Z_{1,32} - Z_{1,4},$$
$$Z_{1,20} = (Z_{1,16} + Z_{1,4})^2 - Z_{1,32} - Z_{1,8},\ Z_{1,22} = (Z_{1,16} + Z_{1,6})^2 - Z_{1,32} - 2Z_{1,12},$$
$$Z_{1,24} = (Z_{1,16} + Z_{1,8})^2 - Z_{1,32} - Z_{1,16},\ Z_{1,26} = (Z_{1,16} + Z_{1,10})^2 - Z_{1,32} - 2Z_{1,20},$$
$$Z_{1,28} = (Z_{1,16} + Z_{1,12})^2 - Z_{1,32} - 2Z_{1,24},\ Z_{1,30} = (Z_{1,16} + Z_{1,14})^2 - Z_{1,32} - 4Z_{1,28},\ C_0^{XZ} = X_{1,32},$$
$$C_1^{XZ} = X_{1,30} \cdot Z_{1,2},\ C_2^{XZ} = X_{1,28} \cdot Z_{1,4},\ C_3^{XZ} = X_{1,26} \cdot Z_{1,6},\ C_4^{XZ} = X_{1,24} \cdot Z_{1,8},$$
$$C_5^{XZ} = X_{1,22} \cdot Z_{1,10},\ C_6^{XZ} = X_{1,20} \cdot Z_{1,12},\ C_7^{XZ} = X_{1,18} \cdot Z_{1,14},\ C_8^{XZ} = X_{1,16} \cdot Z_{1,16},$$
$$C_9^{XZ} = X_{1,14} \cdot Z_{1,18},\ C_{10}^{XZ} = X_{1,12} \cdot Z_{1,20},\ C_{11}^{XZ} = X_{1,10} \cdot Z_{1,22},\ C_{12}^{XZ} = X_{1,8} \cdot Z_{1,24},$$
$$C_{13}^{XZ} = X_{1,6} \cdot Z_{1,26},\ C_{14}^{XZ} = X_{1,4} \cdot Z_{1,28},\ C_{15}^{XZ} = X_{1,2} \cdot Z_{1,30},\ C_{16}^{XZ} = Z_{1,32},$$
$$G = (X_{1,2} + Z_{1,s^2})^2 - X_{1,4} - Z_{1,s^4},\ H = (X_1 + Z1)^2 - X_{1,2} - Z_{1,s},\ II = H^2,\ J = H \cdot II,$$
$$K = (X_{1,4} + Z_{1,s})^2 - X_{1,8} - Z_{1,s^2},\ L = (H + X_{1,4})^2 - II - X_{1,8},\ M = (Y_1 + Z_{1,s^2})^2 - B - Z_{1,s^4},$$
$$N = (Y_1 + Z_{1,s})^2 - B - Z_{1,s^2},\ R = H \cdot N,\ S = II \cdot Y_1,\ T = (X_{1,2} + Y_1)^2 - X_{1,4} - B,\ U = T \cdot H,$$

$$L_{4,0} = -2G \cdot (63C_0^{XZ} + 546C_1^{XZ} - 17646C_2^{XZ} - 86058C_3^{XZ} - 944238C_4^{XZ} - 925278C_5^{XZ} - 4412322C_6^{XZ} - 2092730C_7^{XZ} - 318342C_8^{XZ} + 1595958C_9^{XZ} + 2710846C_{10}^{XZ} + 441618C_{11}^{XZ} + 325074C_{12}^{XZ} + 21510C_{13}^{XZ} + 2930C_{14}^{XZ} - 46C_{15}^{XZ} + C_{16}^{XZ}),$$
$$L_{3,0} = -2J \cdot (105C_0^{XZ} + 756C_1^{XZ} - 15990C_2^{XZ} - 84112C_3^{XZ} - 1082058C_4^{XZ} - 610644C_5^{XZ} - 2610994C_6^{XZ} - 2003688C_7^{XZ} - 13594266C_8^{XZ} - 674868C_9^{XZ} + 164566C_{10}^{XZ} + 223168C_{11}^{XZ} + 232998C_{12}^{XZ} - 492C_{13}^{XZ} + 2226C_{14}^{XZ} + 56C_{15}^{XZ} - 7C_{16}^{XZ}),$$

$$
\begin{aligned}
L_{2,0} &= -4K \cdot (189C_0^{XZ} + 882C_1^{XZ} + 6174C_2^{XZ} - 26274C_3^{XZ} - 1052730C_4^{XZ} - 449598C_5^{XZ} \\
&\quad - 1280286C_6^{XZ} - 1838850C_7^{XZ} - 23063794C_8^{XZ} - 1543290C_9^{XZ} + 539634C_{10}^{XZ} + 646922C_{11}^{XZ} \\
&\quad + 1386918C_{12}^{XZ} + 75846C_{13}^{XZ} + 17262C_{14}^{XZ} + 922C_{15}^{XZ} - 35C_{16}^{XZ}), \\
L_{1,0} &= 4L \cdot (9C_0^{XZ} - 3666C_2^{XZ} + 2580C_3^{XZ} + 263226C_4^{XZ} + 328248C_5^{XZ} \\
&\quad + 1359882C_6^{XZ} + 1017948C_7^{XZ} + 11998650C_8^{XZ} + 1661904C_9^{XZ} + 1958226C_{10}^{XZ} + 178956C_{11}^{XZ} \\
&\quad - 315222C_{12}^{XZ} - 39560C_{13}^{XZ} - 4842C_{14}^{XZ} - 252C_{15}^{XZ} + 7C_{16}^{XZ}), \\
L_{0,0} &= 2X_{1,6} \cdot (C_0^{XZ} - 42C_1^{XZ} - 834C_2^{XZ} - 8702C_3^{XZ} - 38898C_4^{XZ} + 80886C_5^{XZ} \\
&\quad + 654642C_6^{XZ} + 450098C_7^{XZ} + 3346502C_8^{XZ} + 450098C_9^{XZ} + 654642C_{10}^{XZ} + 80886C_{11}^{XZ} \\
&\quad - 38898C_{12}^{XZ} - 8702C_{13}^{XZ} - 834C_{14}^{XZ} - 42C_{15}^{XZ} + C_{16}^{XZ}), \\
L_{3,1} &= 2M \cdot (8C_0^{XZ} + 73C_1^{XZ} - 2718C_2^{XZ} - 12087C_3^{XZ} - 110316C_4^{XZ} - 143283C_5^{XZ} \\
&\quad - 603830C_6^{XZ} - 159171C_7^{XZ} + 1273368C_8^{XZ} + 301915C_9^{XZ} + 286566C_{10}^{XZ} + 27579C_{11}^{XZ} \\
&\quad + 48348C_{12}^{XZ} + 1359C_{13}^{XZ} - 146C_{14}^{XZ} - C_{15}^{XZ}), \\
L_{2,1} &= R \cdot (216C_0^{XZ} + 1719C_1^{XZ} - 49530C_2^{XZ} - 225297C_3^{XZ} - 2336292C_4^{XZ} - 1899741C_5^{XZ} \\
&\quad - 8313570C_6^{XZ} - 3992373C_7^{XZ} - 6366840C_8^{XZ} + 1434309C_9^{XZ} + 2776722C_{10}^{XZ} + 427917C_{11}^{XZ} \\
&\quad + 107508C_{12}^{XZ} + 10017C_{13}^{XZ} + 2122C_{14}^{XZ} - 7C_{15}^{XZ}), \\
L_{1,1} &= S \cdot (504C_0^{XZ} + 3055C_1^{XZ} - 38146C_2^{XZ} - 226593C_3^{XZ} - 3358356C_4^{XZ} - 982485C_5^{XZ} \\
&\quad - 3428010C_6^{XZ} - 4734229C_7^{XZ} - 46394904C_8^{XZ} - 2925939C_9^{XZ} - 560070C_{10}^{XZ} + 510845C_{11}^{XZ} \\
&\quad + 849828C_{12}^{XZ} + 15897C_{13}^{XZ} + 3570C_{14}^{XZ} - 7C_{15}^{XZ}), \\
L_{0,1} &= U \cdot (168C_0^{XZ} + 417C_1^{XZ} + 26106C_2^{XZ} + 19449C_3^{XZ} - 808860C_4^{XZ} - 981963C_5^{XZ} \\
&\quad - 3150686C_6^{XZ} - 1673251C_7^{XZ} - 16203528C_8^{XZ} - 1636605C_9^{XZ} - 889746C_{10}^{XZ} + 58347C_{11}^{XZ} \\
&\quad + 226252C_{12}^{XZ} + 2919C_{13}^{XZ} + 630C_{14}^{XZ} - C_{15}^{XZ}).
\end{aligned}
$$

$$F^* = \alpha \cdot (L_{4,0} \cdot x_S^4 + L_{3,0} \cdot x_S^3 + L_{2,0} \cdot x_S^2 + L_{1,0} \cdot x_S + L_{3,1} \cdot x_S^3 y_S + L_{2,1} \cdot x_S^2 y_S + L_{1,1} \cdot x_S y_S + L_{0,0}),$$

$$A_2 = X_1^2,\ B_2 = Y_1^2,\ C_2 = Z_1^2,\ D_2 = aC_2,\ X_{D^2} = (A_2 - D_2)^2,\ E_2 = 2(A_2 + D_2)^2 - X_{D^2},\ Z_{D^2} = 4B_2,$$
$$F_2 = ((A_2 - D_2 + Y_1)^2 - B_2 - X_{D^2},\ Y_{D^2} = E_2 \cdot F_2.$$

$$A_3 = X_1^2,\ B_3 = Y_1^2,\ C_3 = Z_1^2,\ D_3 = aC_3,\ X_{D^3} = (A_3 - D_3)^2,\ E_3 = 2(A_3 + D_3)^2 - X_{D^3},\ Z_{D^3} = 4B_3,$$
$$F_3 = ((A_3 - D_3 + Y_1)^2 - B_3 - X_{D^3},\ Y_{D^3} = E_3 \cdot F_3.$$

The above sequence of operations costs $32\mathbf{m}_e + 57\mathbf{s}_e + 8e\mathbf{m}_1$.

## B Explicit Formulas

The following MAGMA code is a simple implementation of the Miller quadruple-and-and and Miller octuple-and-add algorithms. We specify curves of the form $y^2 = x^3 + b$ and condense the code due to space considerations. The main function `Miller2nTuple` takes as inputs the two points $R$ and $S$ on $E$, the value $r$ which is the order of $R$, the two curve constants $a$ and $b$, the integer $n$ (for $2^n$-tupling) and the full extension field $K$, so that $R, S \in E(K)$. `Miller2nTuple` either calls the function `Quadruple` or the function `Octuple` for $n = 2$ and $n = 3$ respectively (the call to `Octuple` is currently commented out).

```
function Dbl(X1,Y1,Z1, xQ, yQ,b) A:=X1^2; B:=Y1^2; C:=Z1^2; D:=3*b*C; E:=(X1+Y1)^2-A-B; F:=(Y1+Z1)^2-B-C; G:=3*D; X3:=E*(B-G);
Y3:=(B+G)^2-12*D^2; Z3:=4*B*F; L10:= 3*A; L01:=-F; L00:=D-B; F:=L10*xQ+L01*yQ+L00; return X3,Y3,Z3,F; end function;

function Add(X1, Y1, Z1, X2, Y2, Z2, xQ, yQ) c1:=X2-xQ; t1:=Z1*X2; t1:=X1-t1; t2:=Z1*Y2; t2:=Y1-t2; F:=c1*t2-t1*Y2+t1*yQ;
t3:=t1^2; X3:=t3*X1; t3:=t1*t3; t4:=t2^2; t4:=t4*Z1; t4:=t3+t4; t4:=t4-X3; t4:=t4-X3; X3:=X3-t4; t2:=t2*X3; Y3:=t3*Y1; Y3:=t2-Y3;
X3:=t1*t4; Z3:=Z1*t3; return X3, Y3, Z3, F; end function;

function Quadruple(Tx, Ty, Tz, Sx, Sy, Sx2, SxSy, b)
A:=Ty^2; B:=Tz^2; C:=A^2; D:=B^2; E:=(Ty+Tz)^2-A-B; F:=E^2; G:=Tx^2; H:=(Tx+Ty)^2-A-G; I:=(Tx+E)^2-F-G; J:=(A+E)^2-C-F;
K:=(Ty+B)^2-A-D;
L:=27*b^2*D; M:=9*b*F; N:=A*C; R:=A*L; S:=b*B; T:=S*L; U:=S*C; X3:=2*H*(A-9*S); Y3:=2*C+M-2*L; Z3:=4*J;
L10:=-4*Tz*(5*N+5*R-3*T-75*U);
L20:=-3*G*Tz*(10*C+3*M-2*L); L01:=2*I*(5*C+L); L11:=2*K*Y3; L00:=2*Tx*(N+R-3*T-75*U); F:= L10*Sx+L20*Sx2+L01*Sy+L11*SxSy+L00;
A2:=Y3^2;
B2:=Z3^2; C2:=3*b*B2; D2:= 2*X3*Y3; E2:=(Y3+Z3)^2-A2-B2; F2:=3*C2; X3:= D2*(A2-F2); Y3:=(A2+F2)^2-12*C2^2; Z3:=4*A2*E2;
return X3,Y3,Z3,F;
end function;

function Octuple(X1, Y1, Z1, Sx, Sy, Sx2, SxSy, Sx3, Sx4, Sx2Sy, Sx3Sy, b)
Y12:=Y1^2; Z1s:=Z1^2; Z12:=b*Z1s; A:=X1^2; B:=3*Z12; C:=(X1+Y1)^2-A-Y12; DD:=(Y1+Z1)^2-Y12-Z1s; E:=3*B; X3:=C*(Y12-E);
Y3:=(Y12+E)^2-12*B^2; Z3:=4*Y12*DD; Xt,Yt,Zt:=Dbl(X1,Y1,Z1,Sx,Sy,b); Z14s:=Z1s^2; Y14:=Y12^2; Y18:=Y14^2; Y116:=Y18^2;
Y16:=(Y12+Y14)^2-Y14-Y18; Y110:=(Y18+Y12)^2-Y116-Y14; Y112:=(Y18+Y14)^2-Y116-Y18; Y114:= (Y18+Y16)^2-Y116-2*Y112; Y118:=Y116*Y12;
Y120:=Y116*Y14; Y122:=Y116*Y16; Z14:=b^2*Z14s; Z18:=Z14^2; Z116:=Z18^2; Z16:=(Z12+Z14)^2-Z14-Z18; Z110:=(Z18+Z12)^2-Z116-Z14;
Z112:=(Z18+Z14)^2-Z116-Z18; Z114:= (Z18+Z16)^2-Z116-2*Z112; Z118:=Z116*Z12; Z120:=Z116*Z14; Z122:=Z116*Z16; YZ0:=Y122;
YZ1:=Y120*Z12; YZ2:=Y118*Z14; YZ3:=Y116*Z16; YZ4:=Y114*Z18; YZ5:=Y112*Z110; YZ6:=Y110*Z112; YZ7:=Y18*Z114; YZ8:=Y16*Z116;
YZ9:=Y14*Z118; YZ10:=Y12*Z120; YZ11:=Z122; FF:=A*Z1s; G:=(Y12+Z1s)^2-Y14-Z14s; H:=C*DD; II:=C^2; J:=Y12*(Y12+3*Z12); K:=DD*Z1s;
L:=C*Z1s; M:=A*DD; N:=Y12*DD;
F40 := -18*FF*(-9565938*YZ10+95659380*YZ9-101859525*YZ8+14880348*YZ7+57100383*YZ6-52396146*YZ5+14332383*YZ4-4578120*YZ3-513162*YZ2
+15732*YZ1+7*YZ0);
F30:=-12*G*(-14348907*YZ11+239148450*YZ10-643043610*YZ9+350928207*YZ8-60407127*YZ7-8575227*YZ6-7841853*YZ5 +12011247*YZ4
-3847095*YZ3-1325142*YZ2+56238*YZ1+35*YZ0);
F20:=-27*H*(-54206982*YZ10+157660830*YZ9-120282813*YZ8+50368797*YZ7
-25747551*YZ6+10693215*YZ5 -3826845*YZ4+777789*YZ3+35682*YZ2+4102*YZ1+7*YZ0+4782969*YZ11);
F10 := -18*II*(-4782969*YZ11+ 28697814*YZ10 -129317310*YZ9+130203045*YZ8-48479229*YZ7+11593287*YZ6-619407*YZ5+1432485*YZ4
-883197*YZ3+32814*YZ2-1318*YZ1+YZ0);
F00 :=2*J*(YZ0-6750*YZ1-524898*YZ2-2583657*YZ3 +23839029*YZ4-20484171*YZ5-136370385*YZ6+369351495*YZ7-669084219*YZ8+413461098*YZ9
-47829690*YZ10+14348907*YZ11);
F31 := 8*K*(2484*YZ1-915624*YZ3-90558*YZ2-28697814*YZ10+YZ0+95659380*YZ9- 61115715*YZ8+6377292*YZ7 +19033461*YZ6 - 14289858*YZ5
+3307473*YZ4);
F21 := 216*L*(YZ0+1960*YZ1-56610*YZ2-363744*YZ3+1192077*YZ4-3601260*YZ5 +5084775*YZ6 -3779136*YZ7 +4546773*YZ8 -7085880*YZ9
+3188646*YZ10);
F11 := 72*M*(8148*YZ1-112266*YZ2+434808*YZ3-2214945*YZ4 +11674206*YZ5-24360993*YZ6
+20549052*YZ7-11160261*YZ8+10628820*YZ9-9565938*YZ10+7*YZ0); F01 :=8*N*(-14348907*YZ11+28697814*YZ10-77590386*YZ9+208856313*YZ8
-152208639*YZ7+87333471*YZ6-19135521*YZ5+543105*YZ4-2329479*YZ3 +508302*YZ2-4138*YZ1+21*YZ0);
F:=F01*Sy+F11*SxSy+F21*Sx2Sy+F31*Sx3Sy+F00+F10*Sx+F20*Sx2+F30*Sx3+F40*Sx4; Y32:=Y3^2; Z3s:=Z3^2;
Z32:=b*Z3s; A:=X3^2; B:=3*Z32;
C:=(X3+Y3)^2-A-Y32; DD:=(Y3+Z3)^2-Y32-Z3s; E:=3*B; X3:=C*(Y32-E); Y3:=(Y32+E)^2-12*B^2; Z3:=4*Y32*DD; Y32:=Y3^2; Z3s:=Z3^2;
Z32:=b*Z3s; A:=X3^2; B:=3*Z32; C:=(X3+Y3)^2-A-Y32; DD:=(Y3+Z3)^2-Y32-Z3s; E:=3*B; X3:=C*(Y32-E); Y3:=(Y32+E)^2-12*B^2;
Z3:=4*Y32*DD;
return X3,Y3,Z3,F;
end function;

function Miller2nTuple(R, S, r, a, b, n, K)
Rx:=R[1]; Ry:=R[2]; Rz:=R[3];
Sx:=S[1]; Sy:=S[2]; Sx2:=Sx^2; Sx3:=Sx^3; Sx4:=Sx^4; SxSy:=Sx*Sy; Sx2Sy:=Sx2*Sy; Sx3Sy:=Sx3*Sy;
Rmultiplesmatrix:=[[Rx, Ry, Rz]];
for i:=2 to (2^n-1) by 1 do
    iR:=i*R;
    Rmultiplesmatrix:=Append(Rmultiplesmatrix, [iR[1], iR[2], iR[3]]);
end for;
fRaddvec:=[K!1]; addproduct:=fRaddvec[1];
ptx, pty, ptz, F := Dbl(Rx,Ry,Rz,Sx,Sy,b);
addproduct*:= F;
fRaddvec:=Append(fRaddvec, addproduct);
for i:=3 to (2^n-1) by 1 do
    ptx, pty, ptz, faddvalue := Add(ptx, pty, ptz, Rx, Ry, Rz, Sx, Sy);
    addproduct*:=faddvalue;
    fRaddvec:=Append(fRaddvec, addproduct);
end for;
Tx:=Rx; Ty:=Ry; Tz:=Rz;
f1 := 1; B := IntegerToSequence(r,2^n);
if B[#B] ne 1 then
    Tx, Ty, Tz, F:= Add(Tx, Ty, Tz, Rmultiplesmatrix[B[#B]][1], Rmultiplesmatrix[B[#B]][2], Rmultiplesmatrix[B[#B]][3], Sx, Sy);
    F:=F*fRaddvec[B[#B]];
    f1:=f1*F;
end if;
for i:=#B-1 to 1 by -1 do
Tx, Ty, Tz, F:=Quadruple(Tx, Ty, Tz, Sx, Sy, Sx2, SxSy, b);
    //Tx, Ty, Tz, F:=Octuple(Tx, Ty, Tz, Sx, Sy, Sx2, SxSy, Sx3, Sx4, Sx2Sy, Sx3Sy, b);
    f1:=f1^(2^n)*F;
    if B[i] ne 0 then
        Tx, Ty, Tz, F:= Add(Tx, Ty, Tz, Rmultiplesmatrix[B[i]][1], Rmultiplesmatrix[B[i]][2],
                Rmultiplesmatrix[B[i]][3], Sx, Sy);
        F:=F*fRaddvec[B[i]];
        f1:=f1*F;
    end if;
end for;
return f1;
end function;
```

# ECC2K-130 on Cell CPUs

Joppe W. Bos[1,⋆], Thorsten Kleinjung[1,⋆], Ruben Niederhagen[2,3,⋆], and Peter Schwabe[3,⋆]

[1] Laboratory for Cryptologic Algorithms
EPFL, Station 14, CH-1015 Lausanne, Switzerland
{joppe.bos,thorsten.kleinjung}@epfl.ch
[2] Department of Electrical Engineering
National Taiwan University, 1 Section 4 Roosevelt Road, Taipei 106-70, Taiwan
ruben@polycephaly.org
[3] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands
peter@cryptojedi.org

**Abstract.** This paper describes an implementation of Pollard's rho algorithm to compute the elliptic curve discrete logarithm for the Synergistic Processor Elements of the Cell Broadband Engine Architecture. Our implementation targets the elliptic curve discrete logarithm problem defined in the Certicom ECC2K-130 challenge. We compare a bitsliced implementation to a non-bitsliced implementation and describe several optimization techniques for both approaches. In particular, we address the question whether normal-basis or polynomial-basis representation of field elements leads to better performance. We show that using our software the ECC2K-130 challenge can be solved in one year using the Synergistic Processor Units of less than 2700 Sony Playstation 3 gaming consoles.

**Keywords:** Cell Broadband Engine Architecture, elliptic curve discrete logarithm problem, binary-field arithmetic, parallel Pollard rho.

## 1 Introduction

How long does it take to solve the elliptic curve discrete logarithm problem (ECDLP) on a given elliptic curve using given hardware? This question was addressed recently for the Koblitz curve defined in the Certicom challenge ECC2K-130 for a variety of hardware platforms [2]. This paper zooms into Section 6 of [2] and describes the implementation of the parallel Pollard rho algorithm [17] for the Synergistic Processor Elements of the Cell Broadband Engine Architecture (CBEA) in detail. We discuss our choice to use the technique of bitslicing [7] to accelerate the underlying binary-field arithmetic operations by comparing a bitsliced to a non-bitsliced implementation.

⋆ This work has been supported in part by the European Commission through the ICT Programme under Contract ICT–2007–216499 CACE, and through the ICT Programme under Contract ICT-2007-216646 ECRYPT II. Permanent ID of this document: bad46a78a56fdc3a44fcf725175fd253. Date: February 28, 2010.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 225–242, 2010.

Many optimization techniques for the non-bitsliced version do not require independent parallel computations (batching) and are therefore not only relevant in the context of cryptanalytical applications but can also be used to accelerate cryptographic schemes in practice.

To the best of our knowledge this is the first work to describe an implementation of high-speed binary-field arithmetic for the CBEA. We plan to put all code described in this paper into the public domain to maximize reusability of our results.

**Organization of the paper.** In Section 2 we describe the features of the CBEA which are relevant to this paper. To make this paper self contained we briefly recall the parallel version of Pollard's rho algorithm and summarize the choice of the iteration function in Section 3. Section 4 discusses different approaches to a high-speed implementation of the iteration function on the CBEA. In Sections 5 and 6 we describe the non-bitsliced and the bitsliced implementation, respectively. We summarize the results and conclude the paper in Section 7.

## 2 A Brief Description of the Cell Processor

The Cell Broadband Engine Architecture [12] was jointly developed by Sony, Toshiba and IBM. Currently, there are two implementations of this architecture, the Cell Broadband Engine (Cell/B.E.) and the PowerXCell 8i. The PowerXCell 8i is a derivative of the Cell/B.E. and offers enhanced double-precision floating-point capabilities and a different memory interface. Both implementations consist of a central *Power Processor Element* (PPE), based on the Power 5 architecture and 8 *Synergistic Processor Elements* (SPEs) which are optimized for high-throughput vector instructions. All units are linked by a high-bandwidth (204 GB/s) ring bus.

The Cell/B.E. can be found in the IBM blade servers of the QS20 and QS21 series, in the Sony Playstation 3, and several acceleration cards like the Cell Accelerator Board from Mercury Computer Systems. The PowerXCell 8i can be found in the IBM QS22 servers. Note that the Playstation 3 only makes 6 SPEs available to the programmer.

The code described in this paper runs on the SPEs directly and does not interact with the PPE or other SPEs during core computation. We do not take advantage of the extended capabilities of the PowerXCell 8i. In the remainder of this section we will describe only those features of the SPE which are of interest for our implementation and are common to both the Cell/B.E. and the PowerXCell 8i. Therefore we may address the Cell/B.E. and the PowerXCell 8i jointly as the *Cell processor* or *Cell CPU*. Further information on the current implementations of the Cell Broadband Engine Architecture can be found in [14].

Each SPE consists of a *Synergistic Processor Unit* (SPU) as its computation unit and a *Memory Flow Controller* (MFC) which grants access to the ring bus and therefore in particular to main memory.

### 2.1 SPU – Architecture and Instruction Set

The SPU is composed of three parts: The *Synergistic Execution Unit* (SXU) is the computational core of each SPE. It is fed with data either by the SPU *Register File Unit* (RFU) or by the *Local Storage* (LS) that also feeds the instructions into the SXU.

The RFU contains 128 general-purpose registers with a width of 128 bits each. The SXU has fast and direct access to the LS but the LS is limited to only 256 KB. The SXU does not have transparent access to main memory; all data must be transferred from main memory to LS and vice versa explicitly by instructing the DMA controller of the MFC. Due to the relatively small size of the LS and the lack of transparent access to main memory, the programmer has to ensure that instructions and the active data set fit into the LS and are transferred between main memory and LS accordingly.

**Dual-issuing.** The SXU has a pure RISC-like SIMD instruction set encoded into 32-bit instruction words; instructions are issued strictly in order to two pipelines called *odd* and *even pipeline*, which execute disjoint subsets of the instruction set. The even pipeline handles floating-point operations, integer arithmetic, logical instructions, and word SIMD shifts and rotates. The odd pipeline executes byte-granularity shift, rotate-mask, and shuffle operations on quadwords, and branches as well as loads and stores.

Up to two instructions can be issued each cycle, one in each pipeline, given that alignment rules are respected (i.e., the instruction for the even pipeline is aligned to a multiple of 8 bytes and the instruction for the odd pipeline is aligned to a multiple of 8 bytes plus an offset of 4 bytes), that there are no interdependencies to pending previous instructions for either of the two instructions, and that there are in fact at least two instructions available for execution. Therefore, a careful scheduling and alignment of instructions is necessary to achieve peak performance.

### 2.2 MFC – Accessing Main Memory

As mentioned before, the MFC is the gate for the SPU to reach main memory as well as other processor elements. Memory transfer is initiated by the SPU and afterwards executed by the DMA controller of the MFC in parallel to ongoing instruction execution by the SPU.

Since data transfers are executed in background by the DMA controller, the SPU needs feedback about when a previously initiated transfer has finished. Therefore, each transfer is tagged with one of 32 tags. Later on, the SPU can probe either in a blocking or non-blocking way if a subset of tags has any outstanding transactions. The programmer should avoid to read data buffers for incoming data or to write to buffers for outgoing data before checking the state of the corresponding tag to ensure deterministic program behaviour.

### 2.3 LS – Accessing Local Storage

The LS is single ported and has a *line interface* of 128 bytes width for DMA transfers and instruction fetch as well as a *quadword interface* of 16 bytes width

for SPU load and store. Since there is only one port, the access to the LS is arbitrated using the following priorities:

1. DMA transfers (at most every 8 cycles),
2. SPU load/store,
3. instruction fetch.

Instructions are fetched in lines of 128 bytes, i.e., 32 instructions. In the case that all instructions can be dual issued, new instructions need to be fetched every 16 cycles. Since SPU loads/stores have precedence over instruction fetch, in case of high memory access there should be an `lnop` instruction for the odd pipeline every 16 cycles to avoid instruction starvation. If there are ongoing DMA transfers an `hbrp` instruction should be used giving instruction fetch explicit precedence over DMA transfers.

### 2.4 Determining Performance

The CBEA offers two ways to determine performance of SPU code: performance can either be statically analysed or measured during runtime.

**Static analysis.** Since all instructions are executed in order and dual-issue rules only depend on latencies and alignment, it is possible to determine the performance of code through static analysis. The "IBM SDK for Multicore Acceleration" [13] contains the tool `spu_timing` which performs this static analysis and gives quite accurate cycle counts.

This tool has the disadvantage that it assumes fully linear execution and does not model instruction fetch. Therefore the results reported by `spu_timing` are overly optimistic for code that contains loops or a high number of memory accesses. Furthermore, `spu_timing` can only be used inside a function. Function calls—in particular calling overhead on the caller side—can not be analyzed.

**Measurement during runtime.** Another way to determine performance is through an integrated decrementer (see [14, Sec. 13.3.3]). Measuring cycles while running the code captures all effects on performance in contrast to static code analysis.

The disadvantage of the decrementer is that it is updated with the frequency of the so-called timebase of the processor. The timebase is usually much smaller than the processor frequency. The Cell/B.E. in the Playstation 3 (rev. 5.1) for example changes the decrementer only every 40 cycles, the Cell/B.E. in the QS21 blades even only every 120 cycles. Small sections of code can thus only be measured on average by running the code several times repeatedly.

For cycle counts we report in this paper we will always state whether the count was obtained using `spu_timing`, or measured by running the code.

## 3 Preliminaries

The main task on solving the Certicom challenge ECC2K-130 is to compute a specific discrete logarithm on a given elliptic curve. Up to now, the most

efficient algorithm known to solve this challenge is a parallel version of Pollard's rho algorithm running concurrently on a big number of machines.

In this section we give the mathematical and algorithmic background necessary to understand the implementations described in this paper. We will briefly explain the parallel version of Pollard's rho algorithm, review the iteration function described in [2], and introduce the general structure of our implementation.

### 3.1 The ECDLP and Parallel Pollard rho

The security of elliptic-curve cryptography relies on the believed hardness of the elliptic curve discrete logarithm problem (ECDLP): Given an elliptic curve $E$ over a finite field $\mathbb{F}_q$ and two points $P \in E(\mathbb{F}_q)$ and $Q \in \langle P \rangle$, find an integer $k$, such that $Q = [k]P$. Here, $[k]$ denotes scalar multiplication with $k$.

If the order of $\langle P \rangle$ is prime, the best-known algorithm to solve this problem (for most elliptic curves) is Pollard's rho algorithm [17]. In the following we describe the parallelized collision search as implemented in [2]. See [2] for credits and further discussion.

The algorithm uses a pseudo-random iteration function $f : \langle P \rangle \to \langle P \rangle$ and declares a subset of $\langle P \rangle$ as *distinguished points*. The parallelization is implemented in a client-server approach in which each client node generates an input point with known linear combination in $P$ and $Q$, i.e. $R_0 = [a_0]P + [b_0]Q$ with $a_0$ and $b_0$ generated from a random seed $s$. It then iteratively computes $R_{i+1} = f(R_i)$ until the iteration reaches a distinguished point $R_d$. The random seed $s$ and the distinguished point are then sent to a central server, the client continues by generating a new random input point.

The server searches for a *collision* in all distinguished points sent by the clients, i.e. two different input points reaching the same distinguished point $R_d$. The iteration function is constructed in such a way that the server can compute $a_d$ and $b_d$ such that $R_d = a_dP + b_dQ$ from $a_0$ and $b_0$ (which are derived from $s$). If two different input points yield a collision at a distinguished point $R_d$, the server computes the two (most probably different) linear combinations of the point $R_d$ in $P$ and $Q$: $R_d = [a_d]P + [b_d]Q$ and $R_d = [c_d]P + [d_d]Q$. The solution to the discrete logarithm of $Q$ to the base $P$ is then

$$Q = \left[\frac{a_d - c_d}{b_d - d_d}\right] P.$$

The expected number of distinguished points required to find a collision depends on the density of distinguished points in $\langle P \rangle$. The expected amount of iterations of $f$ on all nodes in total is approximately $\sqrt{\frac{\pi|\langle P \rangle|}{2}}$ assuming the iteration function $f$ is a random mapping of size $|\langle P \rangle|$ (see [11]).

### 3.2 ECC2K-130 and Our Choice of the Iteration Function

The specific ECDLP addressed in this paper is given in the Certicom challenge list [9] as challenge ECC2K-130. The given elliptic curve is a Koblitz curve

$E : y^2 + xy = x^3 + 1$ over the finite field $\mathbb{F}_{2^{131}}$; the two given points $P$ and $Q$ have order $l$, where $l$ is a 129-bit prime. The challenge is to find an integer $k$ such that $Q = [k]P$. Here we will only give the definition of distinguished points and the iteration function used in our implementation. For a detailed description please refer to [2], for a discussion and comparison to other possible choices also see [1]:

We define a point $R_i$ as *distinguished* if the Hamming weight of the $x$-coordinate in normal basis representation $\mathrm{HW}(x_{R_i})$ is smaller than or equal to 34. Our iteration function is defined as

$$R_{i+1} = f(R_i) = \sigma^j(R_i) + R_i,$$

where $\sigma$ is the Frobenius endomorphism and

$$j = ((\mathrm{HW}(x_{R_i})/2) \pmod 8) + 3.$$

The restriction of $\sigma$ to $\langle P \rangle$ corresponds to scalar multiplication with some scalar $r$. For an input $R_i = a_i P + b_i Q$ the output of $f$ will be $R_{i+1} = (r^j a_i + a_i)P + (r^j b_i + b_i)Q$. When a collision has been detected, it is possible to recompute the two according iterations and update the coefficients $a_i$ and $b_i$ following this rule. This gives the coefficients to compute the discrete logarithm.

### 3.3 Computing the Iteration Function

Computing the iteration function requires one application of $\sigma^j$ and one elliptic-curve addition. Furthermore we need to convert the $x$-coordinate of the resulting point to normal basis, if a polynomial-basis representation is used, and check whether it is a distinguished point.

Many applications use so-called inversion-free coordinate systems to represent points on elliptic curves (see, e.g., [10, Sec. 3.2]) to speed up the computation of point multiplications. These coordinate systems use a redundant representation for points. Identifying distinguished points requires a unique representation, this is why we use the affine Weierstrass representation to represent points on the elliptic curve. Elliptic-curve addition in affine Weierstrass coordinates on the given elliptic curve requires 2 multiplications, one squaring, 6 additions, and 1 inversion in $\mathbb{F}_{2^{131}}$ (see, e.g. [6]). Application of $\sigma^j$ means computing the $2^j$-th powers of the $x$- and the $y$-coordinate. In total, one iteration takes 2 multiplications, 1 squaring, 2 computations of the form $r^{2^m}$, with $3 \leq m \leq 10$, 1 inversion, 1 conversion to normal-basis, and one Hamming-weight computation. In the following we will refer to computations of the form $r^{2^m}$ as *m-squaring*.

**A note on the inversion.** To speed up the relatively costly inversion we can batch several inversions and use Montgomery's trick [16]: $m$ batched inversions can be computed with $3(m-1)$ multiplications and one inversion. For example, $m = 64$ batched elliptic curve additions take $2 \cdot 64 + 3 \cdot (64 - 1) = 317$ multiplications, 64 squarings and 1 inversion. This corresponds to 4.953 multiplications, 1 squaring and 0.016 inversions for a single elliptic-curve addition.

## 4 Approaches for Implementing the Iteration Function

In the following we discuss the two main design decisions for the implementation of the iteration function: 1.) Is it faster to use bitslicing or a standard approach and 2.) is it better to use normal-basis or polynomial-basis representation for elements of the finite field.

### 4.1 Bitsliced or Not Bitsliced?

Binary-field arithmetic was commonly believed to be more efficient than prime-field arithmetic for hardware but less efficient for software implementations. This is due to the fact that most common microprocessors spend high effort on accelerating integer- and floating-point multiplications. Prime-field arithmetic can benefit from those high-speed multiplication algorithms, binary-field arithmetic cannot. However, Bernstein showed recently that for *batched* multiplications, binary fields can provide better performance than prime fields also in software [3]. In his implementation of batched Edwards-curve arithmetic the bitslicing technique [7] is used to compute (at least) 128 binary-field multiplications in parallel on an Intel Core 2 processor.

Bitslicing is a matter of transposition: Instead of storing the coefficients of an element of $\mathbb{F}_{2^{131}}$ as sequence of 131 bits in 2 128-bit registers, we can use 131 registers to store the 131 coefficients of an element, one register per bit. Algorithms are then implemented by simulating a hardware implementation – gates become bit operations such as `AND` and `XOR`. For one element in 131 registers this is highly inefficient, it may become efficient if all 128 bits of the registers are used for 128 independent (batched) operations. The lack of registers—most architectures including the SPU do not support 131 registers—can easily be compensated for by *spills*, i.e. storing currently unused values on the stack and loading them when they are required.

The results of [3] show that for batched binary-field arithmetic on the Intel Core 2 processor bitsliced implementations are faster than non-bitsliced implementations. However, the question whether this is also the case on the SPU of the Cell processor is hard to answer a priori for several reasons:

- The Intel Core 2 can issue up to 3 bit operations on 128-bit registers per cycle, an obvious lower bound on the cycles per iteration is thus given as the number of bit operations per cycle divided by 3. The SPU can issue only one bit operation per cycle, the lower bound on the performance is thus three times as high.
- Bernstein in [3, Sec. 3] describes that the critical bottleneck for batched multiplications are in fact loads instead of bit operations. The Core 2 has only 16 architectural 128-bit registers and can do only 1 load per cycle, i.e. one load per 3 bit operations.

  The SPUs have 128 architectural 128-bit registers and can do one load per bit operation. However, unlike on the Core 2, the load operations have to compete with store operations. Due to the higher number of registers and the lower

arithmetic/load ratio it seems easier to come close to the lower bound of cycles imposed by the number of bit operations on the SPUs than on the Core 2. How much easier and how close exactly was very hard to foresee.
- Bitsliced operations rely heavily on parallelism and therefore require much more storage for inputs, outputs and intermediate values. As described in Section 2, all data of the active set of variables has to fit into 256 KB of storage alongside the code. In order to make code run fast on the SPU which executes all instructions in order, optimization techniques such as loop unrolling and function inlining are crucial; these techniques increase the code size and make it harder to fit all data into the local storage.
- Montgomery inversions require another level of parallelism, inverting for example 64 values in parallel requires $64 \cdot 128$ field elements (131 KB) of inputs when using bitsliced representation. This amount of data can only be handled using DMA transfers between the main memory and the local storage. In order to not suffer from performance penalties due to these transfers, they have to be carefully interleaved with computations.

We decided to evaluate which approach is best by implementing both, the bitsliced and the non-bitsliced version, independently by two groups in a friendly competition.

### 4.2 Polynomial or Normal Basis?

Another choice to make for both bitsliced and non-bitsliced implementations is the representation of elements of $\mathbb{F}_{2^{131}}$: Polynomial bases are of the form $(1, z, z^2, z^3, \ldots, z^{130})$, so the basis elements are increasing powers of some element $z \in \mathbb{F}_{2^{131}}$. Normal bases are of the form $(\alpha, \alpha^2, \alpha^4, \ldots, \alpha^{2^{130}})$, so each basis element is the square of the previous one.

Performing arithmetic in normal-basis representation has the advantage that squaring elements is just a rotation of coefficients. Furthermore we do not need any basis transformation before computing the Hamming weight in normal basis. On the other hand, implementations of multiplications in normal basis are widely believed to be much less efficient than those of multiplications in polynomial basis.

In [19], von zur Gathen, Shokrollahi and Shokrollahi proposed an efficient method to multiply elements in type-2 normal basis representation. Here we review the multiplier shown in [2]; see [2] for further discussion and history:

An element of $\mathbb{F}_{2^{131}}$ in type-2 normal basis representation is of the form

$$f_0(\zeta + \zeta^{-1}) + f_1(\zeta^2 + \zeta^{-2}) + f_2(\zeta^4 + \zeta^{-4}) + \cdots + f_{130}(\zeta^{2^{130}} + \zeta^{-2^{130}}),$$

where $\zeta$ is a 263rd root of unity in $\mathbb{F}_{2^{131}}$. This representation is first permuted to obtain coefficients of

$$\zeta + \zeta^{-1}, \zeta^2 + \zeta^{-2}, \zeta^3 + \zeta^{-3}, \ldots, \zeta^{131} + \zeta^{-131},$$

and then transformed to coefficients in polynomial basis

$$\zeta + \zeta^{-1}, (\zeta + \zeta^{-1})^2, (\zeta + \zeta^{-1})^3, \ldots, (\zeta + \zeta^{-1})^{131}.$$

Applying this transform to both inputs allows us to use a fast polynomial-basis multiplier to retrieve coefficients of

$$(\zeta+\zeta^{-1})^2, (\zeta+\zeta^{-1})^3, \ldots, (\zeta+\zeta^{-1})^{262}.$$

Applying the inverse of the input transformation yields coefficients of

$$\zeta^2+\zeta^{-2}, \zeta^3+\zeta^{-3}, \ldots, \zeta^{262}+\zeta^{-262}.$$

Conversion to permuted normal basis just requires adding appropriate coefficients, for example $\zeta^{200}$ is the same as $\zeta^{-63}$ and thus $\zeta^{200}+\zeta^{-200}$ is the same as $\zeta^{63}+\zeta^{-63}$. To obtain the normal-basis representation we only have to apply the inverse of the input permutation.

This multiplication still incurs overhead compared to modular multiplication in polynomial basis, but it needs careful analysis to understand whether this overhead is compensated for by the above-described benefits of normal-basis representation. Observe that all permutations involved in this method are free for hardware and bitsliced implementations while they are quite expensive in non-bitsliced software implementations.

## 5 The Non-bitsliced Implementation

For the non-bitsliced implementation, we decided not to implement arithmetic in a normal-basis representation. The main reason is that the required permutations, splitting and reversing of the bits, as required for the conversions in the Shokrollahi multiplication algorithm (see Section 4.2) are too expensive to outweigh the gain of having no basis change and faster $m$-squarings.

The non-bitsliced implementation uses a polynomial-basis representation of elements in $\mathbb{F}_{2^{131}} \cong \mathbb{F}_2[z]/(z^{131}+z^{13}+z^2+z+1)$. Field elements in this basis can be represented using 131 bits, on the SPE architecture this is achieved by using two 128-bit registers, one containing the three most significant bits. As described in Section 3 the functionality of addition, multiplication, squaring and inversion are required to implement the iteration function. Since the distinguished-point property is defined on points in normal basis, a basis change from polynomial to normal basis is required as well. In this section the various implementation decisions for the different (field-arithmetic) operations are explained.

The implementation of an addition is trivial and requires two `XOR` instructions. These are instructions going to the even pipeline; each of them can be dispatched together with one instruction going to the odd pipeline. The computation of the Hamming weight is implemented using the `CNTB` instruction, which counts the number of ones per byte for all 16 bytes of a 128-bit vector concurrently, and the `SUMB` instruction, which sums the four bytes of each of the four 32-bit parts of the 128-bit input. The computation of the Hamming weight requires four cycles (measured).

In order to eliminate (or reduce) stalls due to data dependencies we interleave different iterations. Our experiments show that interleaving a maximum of eight

**Algorithm 1.** The reduction algorithm for the ECC2K-130 challenge used in the non-bitsliced version. The algorithm is optimized for architectures with 128-bit registers.

**Input:** $C = A \cdot B = a + b \cdot z^{128} + c \cdot z^{256}$, such that $A, B \in \mathbb{F}_2[z]/(z^{131} + z^{13} + z^2 + z + 1)$ and $a, b, c$ are 128-bit strings representing polynomial values.
**Output:** $D = C \bmod (z^{131} + z^{13} + z^2 + z + 1)$.
1: $c \leftarrow (c \ll 109) + (b \gg 19)$
2: $b \leftarrow b$ `AND` $(2^{19} - 1)$
3: $c \leftarrow c + (c \ll 1) + (c \ll 2) + (c \ll 13)$
4: $a \leftarrow a + (c \ll 16)$
5: $b \leftarrow b + (c \gg 112)$
6: $x \leftarrow (b \gg 3)$
7: $b \leftarrow b$ `AND` 7
8: $a \leftarrow a + x + (x \ll 1) + (x \ll 2) + (x \ll 13)$
9: **return** $(D = a + b \cdot z^{128})$

iterations maximizes performance. We process 32 of such batches in parallel, computing on 256 iterations in order to reduce the cost of the inversion (see Section 3). All 256 points are converted to normal basis, we keep track of the lowest Hamming weight of the $x$-coordinate among these points. This can be done in a branch-free way eliminating the need for 256 expensive branches. Then, before performing the simultaneous inversion, only one branch is used to check if one of the points is distinguished. If one or more distinguished points are found, we have to process all 256 points again to determine and output the distinguished points. Note that this happens only very infrequently.

### 5.1 Multiplication

If two polynomials $A, B \in \mathbb{F}_2[z]/(z^{131} + z^{13} + z^2 + z + 1)$ are multiplied in a straight-forward way using 4-bit lookup tables, the table entries would be 134-bit wide. Storing and accumulating these entries would require operations (`SHIFT` and `XOR`) on two 128-bit limbs. In order to reduce the number of required operations we split $A$ as

$$A = A_l + A_h \cdot z^{128} = \widetilde{A}_l + \widetilde{A}_h \cdot z^{121}.$$

This allows us to build a 4-bit lookup table from $\widetilde{A}_l$ whose entries fit in 124 bits (a single 128-bit limb). Furthermore, the product of $\widetilde{A}_l$ and an 8-bit part of $B$ fits in a single 128-bit limb. While accumulating such intermediate results we only need byte-shift instructions. In this way we calculate the product $\widetilde{A}_l \cdot B$.

For calculating $\widetilde{A}_h \cdot B$ we split $B$ as

$$B = B_l + B_h \cdot z^{128} = \widetilde{B}_l + \widetilde{B}_h \cdot z^{15}.$$

Then we calculate $\widetilde{A}_h \cdot \widetilde{B}_l$ and $\widetilde{A}_h \cdot \widetilde{B}_h$ using two 2-bit lookup tables from $\widetilde{B}_l$ and $\widetilde{B}_h$. We choose to split 15 bits from $B$ in order to facilitate the accumulation of partial products in

$$C = A \cdot B = \widetilde{A}_l \cdot B_l + \widetilde{A}_l \cdot B_h \cdot z^{128} + \widetilde{A}_h \cdot \widetilde{B}_l \cdot z^{121} + \widetilde{A}_h \cdot \widetilde{B}_h \cdot z^{136}$$

since $121 + 15 = 136$ which is divisible by 8.

The reduction can be done efficiently by taking the form of the irreducible polynomial into account. Given the result $C$ from a multiplication or squaring, $C = A \cdot B = C_H \cdot z^{131} + C_L$, the reduction is calculated using the trivial observation that

$$C_H \cdot z^{131} + C_L \equiv C_L + (z^{13} + z^2 + z^1 + 1)C_H \bmod (z^{131} + z^{13} + z^2 + z + 1).$$

Algorithm 1 shows the reduction algorithm optimized for architectures which can operate on 128-bit operands. This reduction requires 10 `XOR`, 11 `SHIFT` and 2 `AND` instructions. On the SPU architecture the actual number of required `SHIFT` instructions is 15 since the bit-shifting instructions only support values up to 7. Larger bit-shifts are implemented combining both a byte- and a bit-shift instruction. When interleaving two independent modular multiplication computations, parts of the reduction and the multiplication of both calculations are interleaved to reduce latencies, save some instructions and take full advantage of the available two pipelines.

When doing more than one multiplication containing the same operand, we can save some operations. By doing the simultaneous inversion in a binary-tree style we often have to compute the products $A \cdot B$ and $A' \cdot B$. In this case, we can reuse the 2-bit lookup tables from $\tilde{B}_l$ and $\tilde{B}_h$. We can also save operations in the address generation of the products $\tilde{A}_l \cdot B_l$, $\tilde{A}_l \cdot B_h$, $\tilde{A'}_l \cdot B_l$ and $\tilde{A'}_l \cdot B_h$. Using these optimizations in the simultaneous inversion a single multiplication takes 149 cycles (`spu_timing`) averaged over the five multiplications required per iteration.

### 5.2 Squaring

The squaring is implemented by inserting a zero bit between each two consecutive bits of the binary representation of the input. This can be efficiently implemented using the `SHUFFLE` and `SHIFT` instructions. The reduction is performed according to Algorithm 1. Just as with the multiplication two squaring computations are interleaved to reduce latencies. A single squaring takes 34 cycles (measured).

### 5.3 Basis Conversion and $m$-Squaring

The repeated Frobenius map $\sigma^j$ requires at least 6 and at most 20 squarings, both the $x$- and $y$-coordinate of the current point need at most $3 + 7 = 10$ squarings each (see Section 3), when computed as a series of single squarings. This can be computed in at most $20 \times 34 = 680$ cycles ignoring loop overhead using our single squaring implementation.

To reduce this number a time-memory tradeoff technique is used. We precompute all values

$$T[k][j][i_0 + 2i_1 + 4i_2 + 8i_3] = (i_0 \cdot z^{4j} + i_1 \cdot z^{4j+1} + i_2 \cdot z^{4j+2} + i_3 \cdot z^{4j+3})^{2^{3+k}}$$

for $0 \leq k \leq 7$, $0 \leq j \leq 32$, $0 \leq i_0, i_1, i_2, i_3 \leq 1$. These precomputed values are stored in two tables, for both limbs needed to represent the number, of $8 \times 33 \times 16$ elements of 128-bit each. This table requires 132 KB which is more than half of the available space of the local store.

Given a coordinate $a$ of an elliptic-curve point and an integer $0 \leq m \leq 7$ the computation of the $m$-squaring $a^{2^{3+m}}$ can be computed as

$$\sum_{j=0}^{32} T[m][j][\lfloor (a/2^{4j}) \rfloor \bmod 2^4].$$

This requires $2 \times 33$ `LOAD` and $2 \times 32$ `XOR` instructions, due to the use of two tables, plus the calculation of the appropriate address to load from. Our assembly implementation of the $m$-squaring function requires 96 cycles (measured), this is 1.06 and 3.54 times faster compared to performing 3 and 10 sequential squarings respectively.

For the basis conversion we used a similar time-memory tradeoff technique. We enlarged the two tables by adding $1 \times 33 \times 16$ elements which enables us to reuse the $m$-squaring implementation to compute the basis conversion. For the computation of the basis conversion we proceed exactly the same as for the $m$-squarings, only the initialization of the corresponding table elements is different.

### 5.4 Modular Inversion

From Fermat's little theorem it follows that the modular inverse of $a \in \mathbb{F}_{2^{131}}$ can be obtained by computing $a^{2^{131}-2}$. This can be implemented using 8 multiplications, 6 $m$-squarings (using $m \in \{2, 4, 8, 16, 32, 65\}$) and 3 squarings. When processing many iterations in parallel the inversion cost per iteration is small compared to the other main operations such as multiplication. Considering this, and due to code-size considerations, we calculate the inversion using the fast routines we already have at our disposal: multiplication, squaring and $m$-squaring, for $3 \leq m \leq 10$. In total the inversion is implemented using 8 multiplications, 14 $m$-squarings and 7 squarings. All these operations depend on each other; hence, the interleaved (faster) implementations cannot be used. Our implementation of the inversion requires 3784 cycles (measured).

We also implemented the binary extended greatest common divisor [18] to compute the inverse. This latter approach turned out to be roughly 2.1 times slower.

## 6 The Bitsliced Implementation

This section describes implementation details for the speed-critical functions of the iteration function using bitsliced representation for all computations. As

explained in Section 4, there is no overhead from permutations when using bitslicing and therefore the overhead for normal-basis multiplication is much lower than for a non-bitsliced implementation. We implemented all finite-field operations in polynomial- and normal-basis representation and then compared the performance. The polynomial-basis implementation uses $\mathbb{F}_{2^{131}} \cong \mathbb{F}_2[z]/(z^{131} + z^{13} + z^2 + z + 1)$ just as the non-bitsliced implementation.

Due to the register width of 128, all operations in the bitsliced implementation processes 128 inputs in parallel. The cycle counts in this section are therefore for 128 parallel computations.

### 6.1 Multiplication

**Polynomial basis.** The smallest known number of bit operations required to multiply two degree-130 polynomials over $\mathbb{F}_2$ is 11961 [4]. However, converting the sequence of bit operations in [4] to C syntax and feeding it to spu-gcc does not compile because the size of the resulting function exceeds the size of the local storage. After reducing the number of variables for intermediate results and some more tweaks the compiler produced functioning code, which had a code size of more than 100 KB and required more than 20000 cycles to compute a multiplication.

We decided to sacrifice some bit operations for code size and better-scheduled code and composed the degree-130 multiplication of 9 degree-32 multiplications using two levels of the Karatsuba multiplication technique [15]. One of these multiplications is actually only a degree-31 multiplication; in order to keep code size small we use degree-32 multiplication with leading coefficient zero. We use improvements to classical Karatsuba described in [3] to combine the results of the 9 multiplications.

The smallest known number of bit operations for degree-32 binary polynomial multiplication is 1286 [4]. A self-written scheduler for the bit operation sequence from [4] generates code that takes 1303 cycles (`spu_timing`) for a degree-32 binary polynomial multiplication. In total our degree-130 multiplication takes 14503 cycles (measured). This includes 11727 cycles for 9 degree-32 multiplications, cycles required for combination of the results, and function-call overhead.

Reduction modulo the pentanomial $z^{131} + z^{13} + z^2 + z + 1$ takes 520 bit operations, our fully unrolled reduction function takes 590 cycles (measured), so multiplication in $\mathbb{F}_{2^{131}}$ takes $14503 + 590 = 15093$ cycles.

**Normal basis.** The normal-basis multiplication uses the conversion to polynomial basis as described in Section 4.2. For both, conversion of inputs to polynomial basis and conversion of the result to normal basis (including reduction) we use fully unrolled assembly functions. As for multiplication we implemented scripts to schedule the code optimally. One input conversion takes 434 cycles (measured), output conversion including reduction takes 1288 cycles (measured), one normal-basis multiplication including all conversions takes 16653 cycles (measured).

### 6.2 Squaring

**Polynomial basis.** In polynomial-basis representation, a squaring consists of inserting zero-bits between all bits of the input and modular reduction. The first part does not require any instructions in bitsliced representation because we do not have to store the zeros anywhere, we only have to respect the zeros during reduction. For squarings, the reduction is cheaper than for multiplications because we know that every second bit is zero. In total it needs 190 bit operations, hence, squaring is bottlenecked by loading 131 inputs and storing 131 outputs. One call to the squaring function takes 400 cycles (measured).

**Normal basis.** In normal basis a squaring is a cyclic shift of bits, so we only have to do 131 loads and 131 stores to cyclically shifted locations. A call to the squaring function in normal-basis representation takes 328 cycles (measured).

### 6.3 $m$-Squaring

**Polynomial basis.** In polynomial basis we decided to implement $m$-squarings as a sequence of squarings. A fully unrolled code can hide most of the 131 load and 131 store operations between the 190 bit operations of a squaring – implementing dedicated $m$-squaring functions for different values of $m$ would mostly remove the overhead of $m-1$ function calls but on the other hand significantly increase the overall code size.

**Normal basis.** For the normal-basis implementation we implemented $m$-squarings for all relevant values of $m$ as separate fully unrolled functions. The only difference between these functions is the shifting distance of the store locations. Each $m$-squaring therefore takes 328 cycles (measured), just like a single squaring.

**Conditional $m$-Squaring.** The computation of $\sigma^j$ cannot just simply be realized as a single $m$-squaring with $m = j$, because the value of $j$ is most likely different for the 128 bitsliced values in one batch. Therefore the computation of $r = \sigma^j(x_{R_i})$ is carried out using 3 conditional $m$-squarings as follows:
$r \leftarrow x^{2^3}$
**if** $x_{R_i}[1]$ **then** $r \leftarrow r^2$
**if** $x_{R_i}[2]$ **then** $r \leftarrow r^{2^2}$
**if** $x_{R_i}[3]$ **then** $r \leftarrow r^{2^4}$
**return** $r$,
where $x_{R_i}[k]$ denotes the bit at position $k$ of $x_{R_i}$. The computation of $\sigma^j(y_{R_i})$ is carried out in the same way.

When using bitsliced representation, conditional statements have to be replaced by equivalent arithmetic computations. We can compute the $k$-th bit of the result of a conditional $m$-squaring of $r$ depending on a bit $b$ as

$$r[k] \leftarrow (r[k] \texttt{ AND } \neg b) \texttt{ XOR } (r^{2^m}[k] \texttt{ AND } b).$$

The additional three bit operations per output bit can be interleaved with loads and stores needed for squaring. In particular when using normal-basis squaring

(which does not involve any bit operations) this speeds up the computation: A conditional $m$-squaring in normal-basis representation takes 453 cycles (measured).

For the polynomial-basis implementation we decided to first compute an $m$-squaring and then a separate conditional move. This `cmov` function requires 262 loads, 131 stores and 393 bit operations and thus balances instructions on the two pipelines. One call to the `cmov` function takes 518 cycles.

### 6.4 Addition

Addition is the same for normal-basis and polynomial-basis representation. It requires loading 262 inputs, 131 `XOR`s and storing of 131 outputs. Just as squaring, the function is bottlenecked by loads and stores rather than bit operations. One call to the addition function takes 492 cycles (measured).

### 6.5 Inversion

For both polynomial and normal basis the inversion is implemented using Fermat's little theorem. It involves 8 multiplications, 3 squarings and 6 $m$-squarings (with $m = 2, 4, 8, 16, 32, 65$). It takes 173325 cycles using polynomial basis and 136132 cycles using normal basis (both measured). Observe that with a sufficiently large batch size for Montgomery inversion this does not have big impact on the cycle count of one iteration.

### 6.6 Conversion to Normal Basis

**Polynomial basis.** For the polynomial-basis implementation we have to convert the $x$-coordinate to normal basis to check whether we found a distinguished point. This basis conversion is generated using the techniques described in [5] and uses 3380 bit operations. The carefully scheduled code takes 3748 cycles (measured).

### 6.7 Hamming-Weight Computation

The bitsliced Hamming-weight computation of a 131-bit number represented in normal basis can be done in a divide-and-conquer approach (producing bitsliced results) using 625 bit operations. We unrolled this algorithm to obtain a function that computes the Hamming weight using 844 cycles (measured).

### 6.8 Control Flow Overhead

For both, polynomial-basis and normal-basis representation there is additional overhead from additions, loop control, and reading new input points after a distinguished point has been found. This overhead accounts for only about 8 percent of the total computation time. Reading a new input point after a distinguished point has been found takes about 2,009,000 cycles. As an input point takes on average $2^{25.7} \approx 40,460,197$ iterations to reach a distinguished point, these costs are negligible and are ignored in our overall cycle counts for the iteration function.

### 6.9 Complete Iteration

To make the computation of the iteration function as fast as possible we used the largest batch size for Montgomery inversions that allows us to fit all data into the local storage. Our polynomial-basis implementation uses a batch size of 12 and needs 113844 cycles (measured) to compute the iteration function. The normal-basis implementation uses a batch size of 14 and requires 100944 cycles (measured). Clearly, the overhead caused by the conversions for multiplications in the normal-basis implementation is outweighed by the benefits in faster $m$-squarings, conditional $m$-squarings, and the saved basis conversion.

### 6.10 Using DMA Transfers to Increase the Batch Size

To be able to use larger numbers for the batch size we modified the normal-basis implementation to make use of main memory. The batches are stored in main memory and are fetched into LS temporarily for computation.

Since the access pattern to the batches is totally deterministic, it is possible to use multi-buffering to prefetch data while processing previously loaded data and to write back data to main memory during ongoing computations. Even though 3 slots—one for outgoing data, one for computation, and one for incoming data—are sufficient for the buffering logic, we use 8 slots in local memory as ringbuffer to hide indeterministic delays on the memory bus. We assign one DMA tag to each of these slots to monitor ongoing transactions.

Before computation, one slot is chosen for the first batch and the batch is loaded to LS. During one step of the iteration function, the SPU iterates multiple times over the batches. Each time, first the SPU checks whether the last write back from the next slot has finished using a blocking call to the MFC on the assigned tag. Then it initiates a prefetch for the next required batch into this next slot. Now—again in a blocking manner—it is checked whether the data for the current batch already has arrived. If so, data is processed and finally the SPU initiates a DMA transfer to write changed data back to main memory.

Due to this access pattern, all data transfers can be performed with minimal overhead and delay. Therefore it is possible to increase the batch size to 512 improving the runtime per iteration for the normal basis implementation by about 5 percent to 95428 cycles (measured). Measurements on IBM blade servers QS21 and QS22 showed that neither processor bus nor main memory are a bottleneck even if 8 SPEs are doing independent computations and DMA transfers in parallel.

## 7 Conclusions

To the best of our knowledge there were no previous attempts to implement fast binary-field arithmetic on the Cell. The closest work that we are aware of is [8], in which Bos, Kaihara and Montgomery solved an elliptic-curve discrete-logarithm problem over a 112-bit prime field using a PlayStation 3 cluster of 200 nodes. Too many aspects of both the iteration function and the underlying

**Table 1.** Cycle counts per input for all operations on one SPE of a 3192 MHz Cell Broadband Engine, rev. 5.1. For the bitsliced implementations, cycle counts for 128 inputs are divided by 128. The value $B$ in the last row denotes the batch size for Montgomery inversions.

| | Non-bitsliced, polynomial basis | Bitsliced, polynomial basis | Bitsliced, normal basis |
|---|---|---|---|
| Squaring | 34 | 3.164 | 2.563 |
| $m$-squaring | 96 | $m \times 3.164$ | 2.563 |
| Conditional $m$-squaring | — | $m \times 3.164 + 4.047$ | 3.539 |
| Multiplication | 149 | 117.914 | 130.102 |
| Addition | 2 | 3.844 | |
| Inversion | 3784 | 1354.102 | 1063.531 |
| Conversion to normal basis | 96 | 29.281 | — |
| Hamming-weight computation | 4 | 6.594 | |
| Pollard's rho iteration | 1148 ($B = 256$) | 889.406 ($B = 12$) | 788.625 ($B = 14$)<br>745.531 ($B = 512$) |

field arithmetic are different from the implementation in this paper to allow a meaningful comparison.

From the two implementations described in this paper it is clear that on the Cell processor bitsliced implementations of highly parallel binary-field arithmetic are more efficient than standard implementations. Furthermore we show that normal-basis representation of finite-field elements outperforms polynomial-basis representation when using a bitsliced implementation. For applications that do not process large batches of different independent computations the non-bitsliced approach remains of interest. The cycle counts for all field operations are summarized in Table 1 for both approaches.

Using the bitsliced normal-basis implementation—which uses DMA transfers to main memory to support a batch size of 512 for Montgomery inversions—on all 6 SPUs of a Sony Playstation 3 in parallel, we can compute 25.57 million iterations per second. The expected total number of iterations required to solve the ECDLP given in the ECC2K-130 challenge is $2^{60.9}$ (see [2]). Using the software described in this paper, this number of iterations can be computed in 2654 Playstation 3 years.

## References

1. Bailey, D.V., Baldwin, B., Batina, L., Bernstein, D.J., Birkner, P., Bos, J.W., van Damme, G., de Meulenaer, G., Fan, J., Güneysu, T., Gurkaynak, F., Kleinjung, T., Lange, T., Mentens, N., Paar, C., Regazzoni, F., Schwabe, P., Uhsadel, L.: The Certicom challenges ECC2-X. In: Workshop Record of SHARCS 2009: Special-purpose Hardware for Attacking Cryptographic Systems, pp. 51–82 (2009), http://www.hyperelliptic.org/tanja/SHARCS/record2.pdf

2. Bailey, D.V., Batina, L., Bernstein, D.J., Birkner, P., Bos, J.W., Chen, H.-C., Cheng, C.-M., Van Damme, G., de Meulenaer, G., Dominguez Perez, L.J., Fan, J., Güneysu, T., Gürkaynak, F., Kleinjung, T., Lange, T., Mentens, N., Niederhagen, R., Paar, C., Regazzoni, F., Schwabe, P., Uhsadel, L., Van Herrewege, A., Yang, B.-Y.: Breaking ECC2K-130 (2009), http://eprint.iacr.org/2009/541
3. Bernstein, D.J.: Batch binary Edwards. In: Halevi, S. (ed.) Advances in Cryptology – CRYPTO 2009. LNCS, vol. 5677, pp. 317–336. Springer, Heidelberg (2009)
4. Bernstein, D.J.: Minimum number of bit operations for multiplication (May 2009), http://binary.cr.yp.to/m.html (accessed 2009-12-07)
5. Bernstein, D.J.: Optimizing linear maps modulo 2. In: Workshop Record of SPEED-CC: Software Performance Enhancement for Encryption and Decryption and Cryptographic Compilers, pp. 3–18 (2009), http://www.hyperelliptic.org/SPEED/record09.pdf
6. Bernstein, D.J., Lange, T.: Explicit-formulas database, http://www.hyperelliptic.org/EFD/ (accessed 2010-01-05)
7. Biham, E.: A fast new DES implementation in software. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 260–272. Springer, Heidelberg (1997)
8. Bos, J.W., Kaihara, M.E., Montgomery, P.L.: Pollard rho on the PlayStation 3. In: Workshop Record of SHARCS 2009: Special-purpose Hardware for Attacking Cryptographic Systems, pp. 35–50 (2009), http://www.hyperelliptic.org/tanja/SHARCS/record2.pdf
9. Certicom. Certicom ECC Challenge (1997), http://www.certicom.com/images/pdfs/cert_ecc_challenge.pdf
10. Hankerson, D., Menezes, A., Vanstone, S.A.: Guide to Elliptic Curve Cryptography. Springer, New York (2004)
11. Harris, B.: Probability distributions related to random mappings. The Annals of Mathematical Statistics 31, 1045–1062 (1960)
12. Hofstee, H.P.: Power efficient processor architecture and the Cell processor. In: HPCA 2005, pp. 258–262. IEEE Computer Society, Los Alamitos (2005)
13. IBM. IBM SDK for multicore acceleration (version 3.1), http://www.ibm.com/developerworks/power/cell/downloads.html?S_TACT=105AGX16&S_CMP=LP
14. IBM DeveloperWorks. Cell Broadband Engine programming handbook (version 1.11), (May 2008), https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/1741C509C5F64B3300257460006FD68D
15. Karatsuba, A., Ofman, Y.: Multiplication of many-digital numbers by automatic computers. In: Proceedings of the USSR Academy of Science, vol. 145, pp. 293–294 (1962)
16. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation 48, 243–264 (1987)
17. Pollard, J.M.: Monte Carlo methods for index computation (mod $p$). Mathematics of Computation 32, 918–924 (1978)
18. Stein, J.: Computational problems associated with Racah algebra. Journal of Computational Physics 1(3), 397–405 (1967)
19. von zur Gathen, J., Shokrollahi, A., Shokrollahi, J.: Efficient multiplication using type 2 optimal normal bases. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 55–68. Springer, Heidelberg (2007)

# Practical Improvements of Profiled Side-Channel Attacks on a Hardware Crypto-Accelerator

M. Abdelaziz Elaabid[1] and Sylvain Guilley[2]

[1] Université de Paris 8, Équipe MTII, LAGA
2 rue de la liberté, 93526 Saint-Denis Cedex, France
[2] Institut TELECOM / TELECOM ParisTech, CNRS LTCI (UMR 5141)
Département COMELEC, 46 rue Barrault, 75 634 PARIS Cedex 13, France
{sylvain.guilley,elaabid}@TELECOM-ParisTech.fr

**Abstract.** This article investigates the relevance of the theoretical framework on profiled side-channel attacks presented by F.-X. Standaert et al. at Eurocrypt 2009. The analyses consist in a case-study based on side-channel measurements acquired experimentally from a hardwired cryptographic accelerator. Therefore, with respect to previous formal analyses carried out on software measurements or on simulated data, the investigations we describe are more complex, due to the underlying chip's architecture and to the large amount of algorithmic noise. In this difficult context, we show however that with an engineer's mindset, two techniques can greatly improve both the off-line profiling and the on-line attack. First, we explore the appropriateness of different choices for the sensitive variables. We show that a skilled attacker aware of the register transfers occurring during the cryptographic operations can select the most adequate distinguisher, thus increasing its success rate. Second, we introduce a method based on the thresholding of leakage data to accelerate the profiling or the matching stages. Indeed, leveraging on an engineer's common sense, it is possible to visually foresee the shape of some eigenvectors thereby anticipating their estimation towards their asymptotic value by authoritatively zeroing weak components containing mainly non-informational noise. This method empowers an attacker, in that it saves traces when converging towards correct values of the secret. Concretely, we demonstrate a 5 times speed-up in the on-line phase of the attack.

## 1 Introduction

Side-channel attacks are cryptanalytic techniques that exploit unintentional information leakage from cryptographic devices during their operation. In the case of symmetrical encryption or decryption, side-channel attacks aim at recovering the secret key. As a consequence, there is a strong interest in efficient implementation of countermeasures against such attacks. In the meantime, researchers have tackled the difficult task to formalize the study of attacks and countermeasures. A seminal theoretical study dealing with physical attacks is presented to the cryptographic community by S. Micali and L. Reyzin in [11]. In order to be practically

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 243–260, 2010.

usable, this paradigm requires a specialization, grounded on practical side-channel leakage simulations or measurements. This milestone is achieved by F.-X. Standaert *et al.* in [21], where the information theory is solicited to process the side-channel data. More precisely, in [21,20], F.-X. Standaert *et al.* discuss two metrics to measure two independent aspects: the first is the robustness evaluation of a target circuit and the second is the estimation of the power of an attack. For this purpose, two kinds of adversaries should be taken into consideration:

1. The adversary with an access to a certain amount of *a priori* information, such as the circuit leakage model and the algorithm being executed.
2. The adversary with an access to a clone of the attacked device. In this case, she would be free to manipulate it, in order to indirectly understand the behavior of the target device containing the secret information.

In this paper, we consider the second case. It unfolds in two stages. In a first stage of profiling, some estimations are carried out to extract information. In a second stage, a classification is carried out to perform the on-line attack.

The rest of the paper is organized as follows. Section 2 introduces the theoretical notions employed in the forthcoming concrete analyses. The first attack improvement we investigate is related to the adequate choice of the sensitive variable. This study is detailed for both evaluation and attack metrics in Sec. 3. The second improvement concerns an analysis of the dates with maximal leakage. In Sec. 4, we indeed observe that a thresholding that anticipates the irrelevance of some side-channel samples is beneficial to both the evaluation and the attack metrics. A discussion about the impact of these two findings on the interpretation of the *a priori* evaluation metric and of the *a posteriori* attack metrics is given in Sec. 5. Finally, conclusions and perspectives are in Sec. 6.

## 2 Theoretical Framework for the Practice-Oriented Side-Channel Study

### 2.1 Prerequisites

Based on some axioms, the theoretical analysis of the physical attacks detailed in [11] formalize the leakage by specifying "where, how, and when" the target circuit leaks information. Those axioms give a reliable description of the physical phenomena observable in practice. In this context, two technical definitions must be taken into consideration: the *leakage function* and the *adversary*.

**Leakage Function.** The leakage function $\mathcal{L}(C, M, R)$ is a function of three parameters. $C$ is the current internal configuration of the circuit / algorithm, which incorporates all the resources whose side-channels are measurable in principle; $M$ is a set of measures and $R$ is a random string which represents the noise.

This function is an important element in the rest of this article. Indeed, from this notion we will create distinguishers that will enable us to evaluate the circuit by quantifying its side-channel information, or to be used by an adversary to recover the key.

**Adversary.** In [11], the adversary is chosen so as to be the strongest possible and therefore can be different at each attack. In [20], the adversary uses a divide-and-conquer strategy to retrieve separately parts of the secret key. In other words, it defines a function $f : K \mapsto S_K$ which maps each key $k$ onto a class $s_k = f(k)$ such that $|S_K| \ll |K|$. The aim of a side-channel adversary is to guess a key class $s_k$ with non negligible probability.

For example, in the case where the considered encryption algorithm is DES (Data Encryption Standard), $k$ will be the 56-bit key for encryption, and $S_k$ is the 6-bit subkey generated by the key schedule from the master key $k$ and consumed at the substitution box (sbox) input.

According to the given definitions, the evaluation of the physical security of a device is based on the quality of the circuit and the strength of the adversary.

These two aspects urge to study the information quantity given by a leakage function. This will be taken advantage of to conduct a successful attack.

### 2.2 How to Quantify the Information Amount?

The concept of entropy was introduced by C. Shannon in [17]. It is a reliable measurement of uncertainty associated with a random variable. In our case, we use Shannon entropy to characterize the information leaked from a cryptographic device in order to extract its secret key. We denote by:

- $S_K$ the target key class discrete variable of a side-channel attack, and $s_K$ a realisation of this variable;
- $\mathbf{X}$ the discrete variable containing the inputs of the target cryptographic device, and $x$ the realisation of this variable;
- $\mathbf{L}$ a random variable denoting the side-channel observation generated with inputs of the target device, and $l$ a realisation of this random variable;
- $\Pr[s_K \mid \mathbf{L}]$ the conditional probability of a key class $s_K$ given a leakage $l$.

One reason to quantify the information is to measure the quality of the circuit for a given leakage function. For this purpose, we use the conditional entropy, defined in Sec. 2.2. The other goal is to measure how brightly this information (leakage) is used to successfully recover the key. In the sequel, we consider the success rate, defined in Sec. 2.2, to assess the strength of the adversary.

These two metrics enable us subsequently to ensure the circuit security (knowing its leakage) against a more or less strong adversary.

**The Conditional Entropy.** According to the definition of the conditional Shannon entropy, the conditional uncertainty of $S_K$ given $L$, denoted $\mathbf{H}(S_K \mid L)$, is defined by the following equations:

$$\begin{aligned}\mathbf{H}(S_K \mid \mathbf{L}) &\doteq \sum_{s_K}\sum_{l} -\Pr(s_K, l) \cdot \log_2 \Pr(s_K \mid l) \\ &= -\sum_{s_K}\Pr(s_K)\sum_{l}\Pr(l \mid s_K) \cdot \log_2 \Pr(s_K \mid l)\,. \end{aligned} \tag{1}$$

We the define conditional entropy matrix as:

$$\mathbf{H}_{s_K,s_{Kc}} \doteq -\sum_{l} \Pr(l \mid s_K) \cdot \log_2 \Pr(s_{Kc} \mid l)\,, \tag{2}$$

where $s_K$ and $s_{Kc}$ are respectively the correct subkey and the subkey candidate.

From (1) and (2), we derive:

$$\mathbf{H}(S_K \mid \mathbf{L}) = \sum_{s_K} \Pr(s_K)\ \mathbf{H}_{s_K,s_K}\,. \tag{3}$$

The value of diagonal elements from this matrix needs to be observed very carefully. Indeed, theorem 1 in [21] states that if they are minimum amongst all key classes $s_K$, then these key classes can be recovered by a Bayesian adversary.

**The Success Rate.** The adversary mentioned in 2.1 is an algorithm that aims at guessing a key class $s_K$ with high probability. Indeed with some queries, the adversary would estimate the success rate from the number of times for which the attack is successful. The success rate quantifies the strength of an adversary and thereafter evaluates the robustness of a cryptographic device in front of its attack.

### 2.3 Profiled Side-Channel Attacks in Practice

There exist many operational contexts in which an attack can be setup. Some attacks consist solely of an on-line phase. The earliest attacks of this kind exploit explicitly the linear dependency between the secret data and the leakage, as in the Differential Power Attack (DPA, [9]) and the Correlation Power Analysis (CPA, [3]). Recently, the Mutual Information Analysis (MIA, [6]) has extended those attacks to non-linear dependencies. Profiled attacks are attacks mounted by a rather strong adversary that can benefit from a long period of training (*i.e.* profiling) on a clone device, before launching the attack itself on the target device. Template attacks [4,2,7] and stochastic attacks [16,15] belong to this class, because they exploit all the information extracted during the training step. In this paper we concentrate on template attacks, as described in [2],for computing the success rates.

**Template Attacks.** During the *training phase* the attacker gathers a large number of traces, corresponding to random values of plaintexts and keys. As the clone system is in full control of the attacker, this number is limited only by time and available storage. The observed traces are then classified according to functions $\mathcal{L}$ that capture one modality of the circuit's leakage. A trace $t$ is considered as the realisation of a multivariate Gaussian random variable in $\mathbb{R}^N$. For each set $\mathcal{S}_k, k \in [0, N'[$ the attacker computes the average $\mu_k$ and the covariance matrix $\Sigma_k$. These are estimated by:

$$\mu_k = \frac{1}{|\mathcal{S}_k|} \sum_{t \in \mathcal{S}_k} t \qquad \text{and} \qquad \Sigma_k = \frac{1}{|\mathcal{S}_k| - 1} \sum_{t \in \mathcal{S}_k} (t - \mu_k)(t - \mu_k)^{\mathsf{T}}\,. \tag{4}$$

The ordered pair $(\mu_k, \Sigma_k)$ is called *the template associated with value $k$ of the subkey.*

The difficult part of this attack is the size of the $\Sigma_k$ matrices, namely $(N \times N)$, with $N \approx 10^4$. In our experiments, $N = 20,000$. To overcome this a few special indices can be chosen in $[0, N[$, called *interest points*, which contain most of the useful information. Various techniques are used to select these points: points with large differences between the average traces [4], points with maximal variance [7], and, more generally, principal component analysis (*aka* PCA [8].)

**Template Attacks in PCA.** The PCA consists in computing the eigenvectors $EV_i, i \in [0, N'[$ of empirical covariance matrix of all traces together, computed by:

$$S = \frac{1}{N'-1} \sum_{k=0}^{N'-1} TT^{\mathsf{T}} \quad \text{where} \quad \bar{\mu} = \frac{1}{N'} \sum_{k=0}^{N'-1} \mu_k \quad \text{and} \quad T = \begin{pmatrix} \mu_1^{\mathsf{T}} - \bar{\mu}^{\mathsf{T}} \\ \mu_2^{\mathsf{T}} - \bar{\mu}^{\mathsf{T}} \\ \vdots \\ \mu_N^{\mathsf{T}} - \bar{\mu}^{\mathsf{T}} \end{pmatrix}.$$

Let $EV$ be the matrix containing the most significant eigenvectors. In practice, only a few eigenvectors is sufficient to represent all data samples. In the case of our unprotected circuit, the leakage is well captured by one direction only, irrespective of the choice of the leakage function $\mathcal{L}$. The mean traces and covariance matrices defined in (4) are then expressed in this basis by:

$$\nu_k = (EV)^{\mathsf{T}} \mu_k \qquad \text{and} \qquad \Lambda_k = (EV)^{\mathsf{T}} \Sigma_k (EV)\,.$$

In addition to the $(EV)$ matrix, they constitute the templates of the PCA.

The *attack phase* consists then in acquiring the trace $\tau$ of an encipherement performed by the target system using the secret key $\kappa$, projecting it into this latter basis and matching it against the patterns using Bayes' rule. The attack is successful iff:

$$\kappa = \text{argmax}_k \left( \frac{1}{\sqrt{(2\pi)^{N'} |\Lambda_k|}} \exp\left( -\frac{1}{2} \cdot (EV(\tau - \mu_k))^{\mathsf{T}} \Lambda_k^{-1} (EV(\tau - \mu_k)) \right) \right).$$

**Applications of Templates in PCA to SecMat.** The investigations are done on an unprotected DES cryptoprocessor with an iterative architecture. We estimate the conditional entropy and the success rate for various leakage models: in fact, models are constructed from distinguishers for trace classification during the profiling attack. Altogether, this comparison will allow us characterize better the leakage.

We are facing the problem of choosing the relevant sensitive variable.

Actually, it has already been pointed out, for instance in Section 2 of [20], that there are variables more sensitive than others from the attacker point of view. Thus, the security of a cryptographic device should not be treated generally, but must depend on each sensitive variable.

We must choose a variable that is sensitive (*i.e.* linked to the key) and predictable (*i.e.* the size $N'$ of the subkey is manageable by an exhaustive search). This leads to several questions:

– where to attack? For instance at the input or at the output of an sbox?
– plain values or distance between two consecutive values? In the latter case, do we need the schematic of the algorithm implementation in the circuit?
– considering a transition as sensitive variable, is it worth considering the Hamming distance instead of the plain distance[1]?

To find the best compromise security / attacker, our experiments will attempt to respond to these issues. We study the following leakage models. **Model A** is the input of the first sbox, a 6-bit model. **Model B** is the output of the first sbox, a 4-bit model. **Model C** is the value of the first round corresponding to the fanout of the first sbox, a 4-bit model. **Model D** is the transition of model C. **Model E** is the Hamming weight of the model D.

We illustrate our study on the first round; the last round yields similar results. The mathematical definition of the models A to E is given in Tab. 1, using the notation of NIST FIPS 46 for the internal functions of DES and S as a shortcut for $S_1||S_2||\cdots||S_8$. For a better readability, we also provide with a dataflow illustration in Fig. 1.

**Table 1.** The five leakage models confronted in this paper

| Model index | Mathematical description | Abbre--viation | Nature | Dist--ance |
|---|---|---|---|---|
| A | $(R_0 \oplus K_1)[1:6]$ | R0+K1 | Combi. (shallow) | No |
| B | $S(R_0 \oplus K_1)[1:4]$ | S(R0+K1) | Combi. (deep) | No |
| C | $R_1\{9,17,23,31\} = P^{-1}(R_1)[1:4] = (S(R_0 \oplus K_1)) \oplus P^{-1}(L_0))[1:4]$ | R1 | Sequential | No |
| D | $(R_0 \oplus R_1)\{9,17,23,31\}$ | R0+P1 | Sequential | Yes |
| E | $\lvert(R_0 \oplus R_1)\{9,17,23,31\}\rvert$ | \|R0+R1\| | Sequential | Yes |

We use the term "model" to designate the choice of the variable that depends on the plaintext or the ciphertext, and whose values determine the number of templates. This vocabulary can be misleading since it has also been used in the past to represent other quantities. In the original paper about the DPA [10], the model is actually called a "selection function". In [14], the term "leakage model" qualifies the way a sensitive variable is concretely leaked; for instance, according to their terminology, the choice for the Hamming weight is a model of "reduction" of the sensitive variable. In this respect, they would say that E approximately models the leakage of D. We do not use this vocabulary, since we think that an attacker cannot tell the difference between an internal leakage and an externally observable one. This detail is rather a sophistication of the attack, best captured by the concepts behind the stochastic attacks [16]. The notion of "sensitive variable" is neither adequate, because it holds only if there

[1] Given two bitstrings $x_0$ and $x_1$ we call their plain distance the word $x_0 \oplus x_1$, as opposed to their Hamming distance defind as the integer $|x_0 \oplus x_1|$.

is a straightforward link in the leakage and a value present in the algorithm specification (like our model A, B or C), Now, it seems difficult to qualify of sensitive variable a Hamming distance between two internal values taken by a sensitive variable, because it is unrelated to the algorithm itself. So, to sum up, we recall that we employ the term model to define the mapping between a trace and its template.

We note that these models can be viewed as two families of distinguishers. The models A and B are concerned with moments during the encryption, while models B, C and D focus on registers. Each studied case corresponds to one number of templates. Model A uses $64 = 2^6$ templates because it predicts the activity of the sbox input, whereas model B applies to $16 = 2^4$ templates that are the sbox output values. Models C and D are also made up of 16 templates, and finally, model E focuses on 5 templates that are classifications of the different values of a 4 bit Hamming distance.

The acquisition campaign consists in a collection of $80,000$ traces. Half of these traces are used for profiling and the other half for the on-line attack. Templates as shown above are in practice a set of *means and covariances* for classes constructed from models. In addition PCA allows to construct eigenvectors that will be useful to reduce our data samples for online attack. In practice we use only the first eigenvector in order to project circuit consumption data. Indeed it is the most important vector since it relates to the greatest variance. We can also use the second or third eigenvectors if they are relevant, namely if they correspond to a large variance. Thus, we can increase the attacker's strength. On the other hand, using a bad eigenvector only contributes to damage our attack. In figure 2 we can see the difference between a good (the second one) and a bad (the thirteenth one) eigenvector. With an unprotected circuit, it appears clearly that the leakage is well localized in time. Therefore a good eigenvector (corresponding to a high eigenvalue) is also almost null everywhere but at the leaking dates. We will take advantage of this property in the second improvement presented later on in Sec. 4.

According to the eigenvectors represented in Fig. 3, we can see precisely when our circuit leaks information in the modality A, B, D, E or F. The highest the eigenvector at a sample, the largest the leakage there.

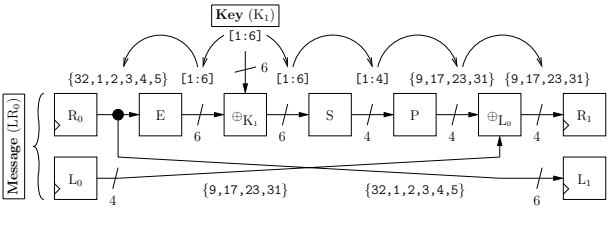
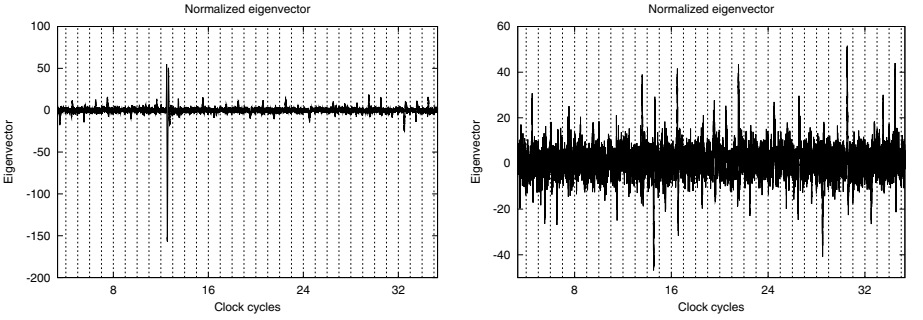



**Fig. 1.** Datapath of DES involved in the attack of the first round




(a) Good eigenvector. (b) Bad eigenvector.

**Fig. 2.** Difference between "good" and "bad" eigenvectors for model B

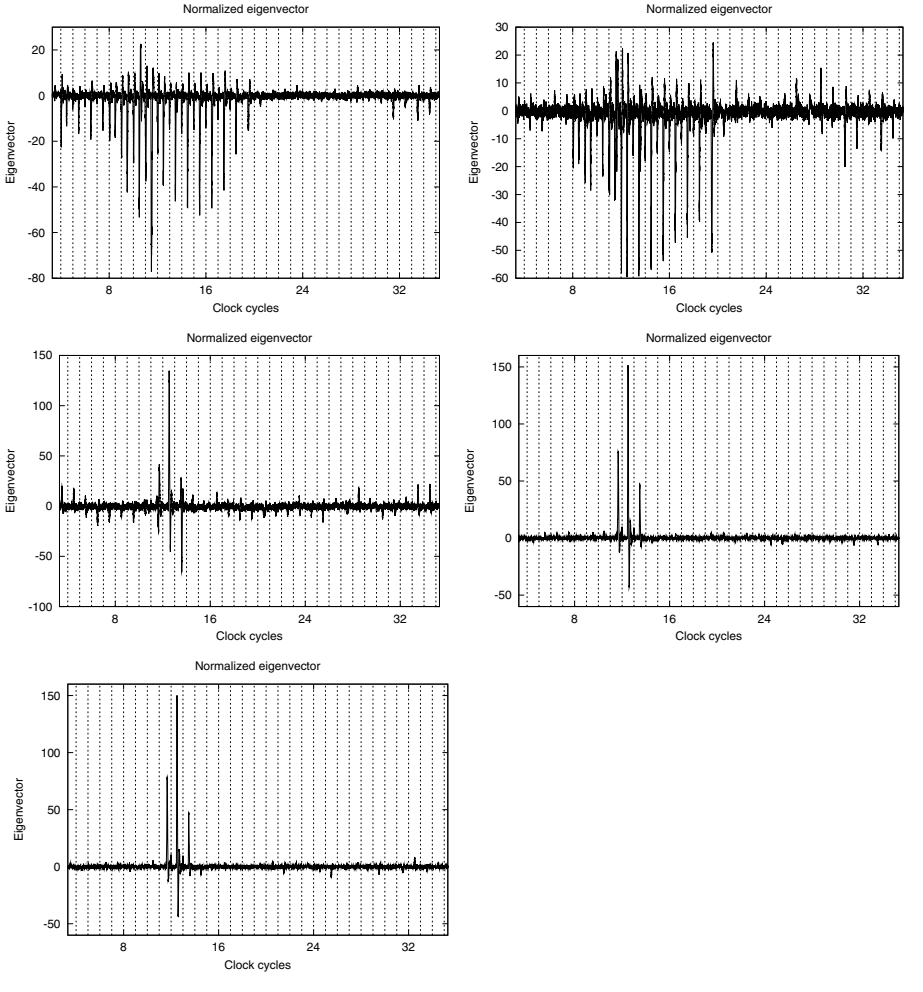



**Fig. 3.** Eigenvectors, for models A, B, C, D & E (left to righ, up to down)

# 3 Improvement of the Attacks Thanks to an Adequate Leakage Model

## 3.1 Success Rate

Success rates are depicted in Fig. 4 as a function of two parameters: the first one is the off-line profiling trace count and the second is the on-line attack trace count. For all curves, we notice that the success rate is an increasing function in terms of either trace count.

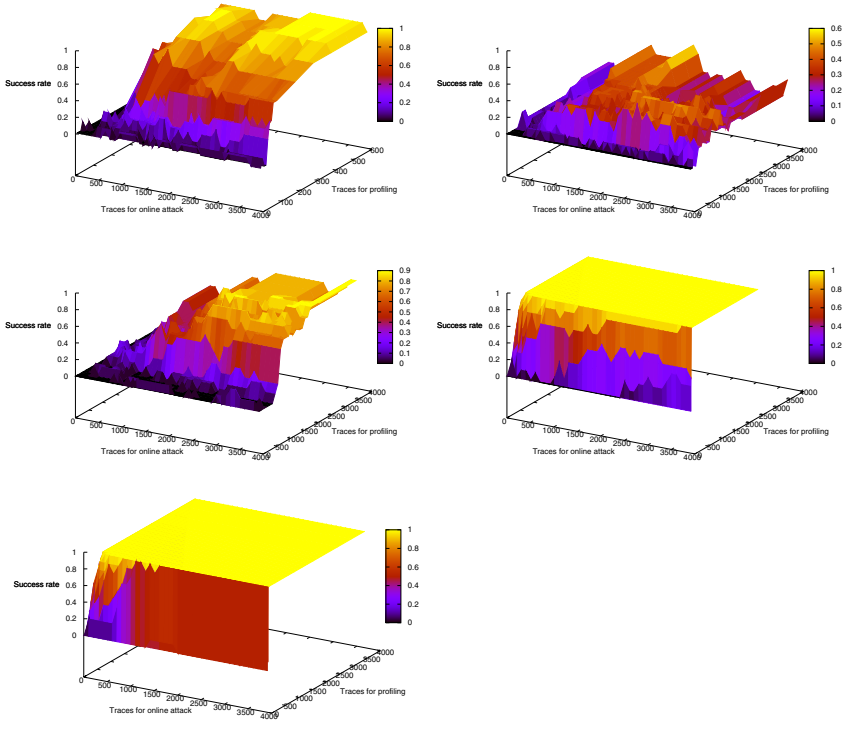


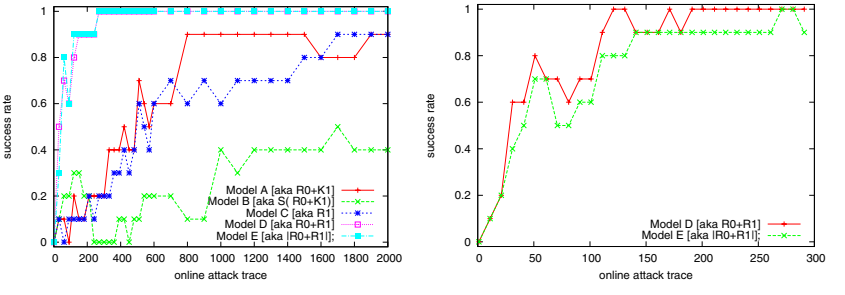

**Fig. 4.** Success rate, for models A, B, C, D & E (left to right, up to down)

Model A seems better than B. Intuitively, we would have expected a similar attack performance, as the traversal of the sbox is a quasi-bijective function. This is not exactly true on DES, since the sbox of DES has a fanin of 6 and a smaller fanout of 4. However, they are designed in such a way that if two input bits are fixed, the sbox becomes bijective (in answer for a balancing requirement).

Therefore, the reason for B to be slightly worse than A is related to the implementation. Indeed, in the round combinational logic, B addresses a deeper leakage than A. Thus B is more subject to intra-clocking cycle timing shifts in the transitions and to the spurious glitching activity that characterizes the CMOS combinational parts.

Following this discussion, we could also expect the model C to be better than A, since C models the activity of a register. However, this is false in practice. One explanation is that the eigenvector of A has more peaks than that of C, and thus collects more information scattered in the trace.

Figure 5(a) illustrates that the model D and E are by far the best to conduct an attack. Indeed, few traces are required for the attack success probability to reach one. Only about 200 traces, allow to have a 90% probability of success. With 250 traces, the probability rises to 100%. The conclusion is that the knowledge of the register transfers in a circuit can significantly enhance the attack. In an unprotected CMOS circuit, these models are unrivaled. These intuitions were already evoked in the literature (for instance in [3] or in [20]), but never demonstrated. Through our experiment, we indeed formally concur on this point: distances between consecutive states are leaking more than values in CMOS circuits.




(a) (b)

**Fig. 5.** Success rate comparison, (a) between all models for the same number of pre-characterization traces and (b) between models D and E for the same number of profiling traces per class

Eventually, we intend to compare models D (template-like [4]) and E (stochastic-like [16]). To be fair, we compute the success rate for a common number of traces used in each template; namely, 16/5 more traces are required to profile D. The result, given in Fig. 5(b) shows without doubt that D achieves better than E. The reason is that E is an approximate model. Figure 6 indeed shows that the degeneracy of identical Hamming weight classes is not perfect in practice; however, for a given number of measurements dedicated to profiling, E is favored.

### 3.2 Conditional Entropy

As shown in Figure 7(a), the conditional entropy is roughly decreasing with the number of traces used in the profiling phase. We do not comment on the very beginning of the curves, since for small numbers of traces in the profiling phase, the entropy estimation yields false results. However, we notice that the conditional

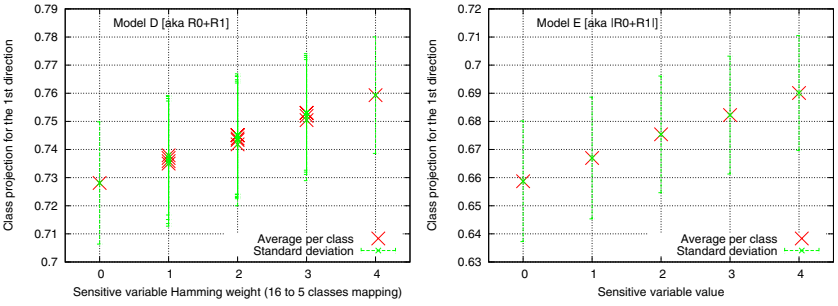
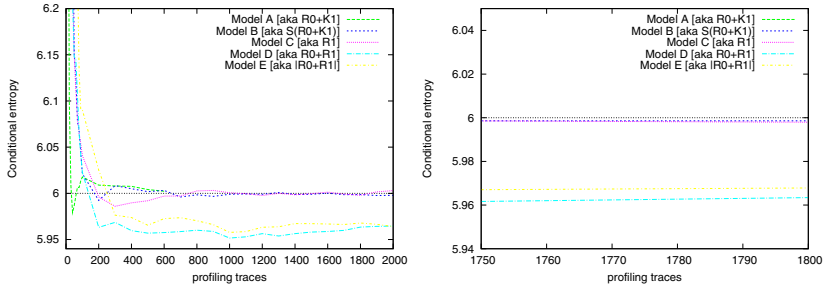



**Fig. 6.** Comparison between the averages of models D and E using the sole principal component




(a) (b)

**Fig. 7.** Conditional entropy comparison. (a) over 2000 traces; (b) zoom after the estimation has converged.

entropy tends towards a bounded value. As expected, this asymptotically value is less than or equal to 6 bits. Indeed, the keys we attack have 6 bits of entropy. The smaller the conditional entropy for a chosen model, the more vulnerable the circuit against one attack using this model.

In this comparison, we have tried to ensure having the same number of traces during the profiling phase. But this was not possible for model A that needs more measurements as it has more classes (64) than the others (16 or 5).

Therefore, we had the number of traces live in [0,600] for model A and in [0,2000] for the other models. By comparing the different models, it appears clearly that the models D and E are more favorable to side-channel leakage. Figure 7(b) confirms that asymptotically, the conditional entropy for models D and E is smaller than other models, and that D is better than E. Therefore, the choice of the sensitive variable is an important element to be taken into account in a security evaluation. Basically, we observe that the conditional entropy classifies the models in the same order as the success rate.

Now, we warn that in theory, the success rate and the conditional entropy are definitely not the same concept. However, in this study, it appears that there is necessarily a similarity between them. This is certainly due to the choice of

models, and also to the adequation in terms of relationship adversary/security. Indeed, on the contrary, we must be aware of some pitfalls of interpretation, like for instance the risk that a circuit may seem very well protected against a bad adversary.

But to sum up, this study of different models shows that the entropy is actually a good way to evaluate a circuit.

### 3.3 Hidden Models

We say that a leakage model $\mathcal{L}_1$ is more adequate than $\mathcal{L}_2$ if the success rate of an attack using $\mathcal{L}_1$ is greater than with $\mathcal{L}_2$. In the SecMat circuit we studied, the distance between the consecutive values of a sensitive variables held in registers corresponds to the leakage function E. Discovering this better model is always possible, provided however that the attacker has a thorough knowledge of the circuit's architecture. More specifically, the value of a sensitive variable is accessible from anyone in possession of the algorithm specifications; on the contrary, the knowledge of the variables sequence requires that of the chip's register transfer level (RTL) architecture, such as for instance its VHDL or Verilog description. Even in the absence of these pieces of information, an attacker can still attempt to retrieve the expression of the sensitive variables (or its sequence) as a function of the algorithm inputs or outputs, depending the attack is in known plaintext or known ciphertext context. Indeed, the attack acts as an oracle: if it fails to recover the key, it means that the leakage function is irrelevant. However, in hardware circuits, these functions are of high algebraic degree, especially when the next pipeline stage is a register that is one round inside the algorithm. This makes its prediction chancy, unless having some hints about a possible class of leakage functions. Refer to articles about SCARE [12,13] for a more complete overview of this problem.

We notice that the best leakage model (the most physical one) can be unknown willingly or as a backdoor. The algorithm pipeline can be indeed very complicated.

## 4 Improvement of the Attacks Thanks to Leakage Profiles Noise Removal by Thresholding

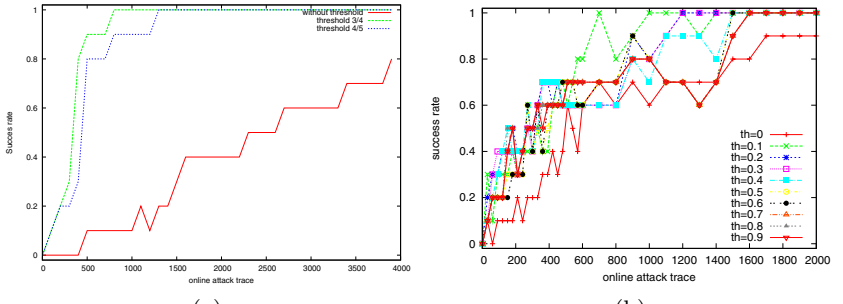
As already emphasized in Fig. 2, an adversary able to understand the shape of the eigenvectors is more likely to master the speed of the success rate of her attack. In a similar way, a security evaluator may require an ideal eigenvector to have a very clear idea of the expertized device security.

In the case of a noisy vector, we must seek the best moments of leakage and eliminate the noise. In this context we suggest to improve the success rate or degree of evaluation by creating a threshold $th \in [0, 1]$ on the eigenvectors. Generally speaking, in the case where we have infinite traces, the eigenvector will tend towards a denoised curve that reflects perfectly the temporal information for the leakage. In concrete evaluation and attack scenarios, we experience time or space constraints, and therefore we seek a better way to refine the eigenvectors.

### 4.1 Success Rate

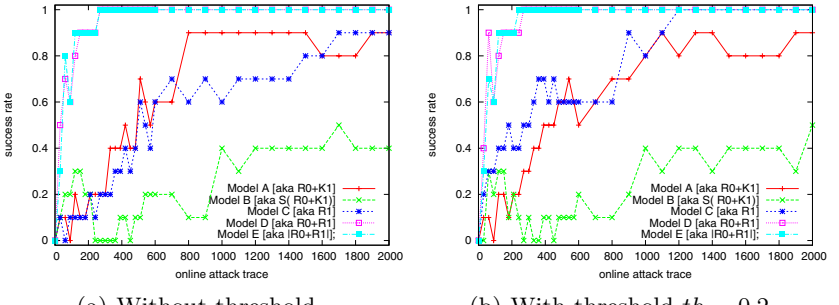
Figure 8(a) shows the significant gain which may be attributed to an adversary who perfectly knows how to exploit eigenvectors. This experiment was done with noisy traces different from those used in previous sections. The model chosen is the model A, which shows a significant increase in the rate of success (about $\times 5$). There, the optimal threshold $th_{\text{opt}}$ is either 3/4 or 4/5 of the maximum value of the eigenvector. Actually, we see that for an eigenvector whose noise is eliminated, the success rate becomes higher. Threshold choice is a compromise: when it is too high ($th \approx 1$), some noise remains, whereas when it is too low ($th \approx 0$), it eliminates informative parts provided by the profiling phase.

Figure 8(b) shows on the example of model C how the success rate improves depending on the chosen threshold. For models of the same family (based on the value instead of on a distance), such as A and C, it can be noted that an adversary may be mistaken about the success rate, believing that such model is better than the other. In our case, if we do not take a threshold it is believed that the model A is better than C. However, with a thresholding, the conclusions are inverse, as depicted in Fig. 9.




(a) (b)

**Fig. 8.** Success rate comparison (a) without and with threshold for model A and (b) with different thresholds for model C

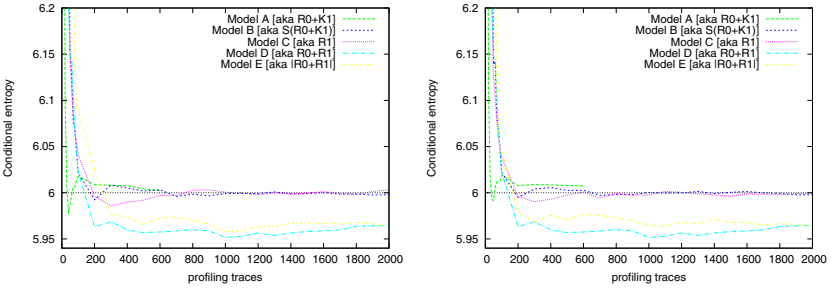



(a) Without threshold. (b) With threshold $th = 0.2$.

**Fig. 9.** Success rate improvements with threshold

### 4.2 Conditional Entropy

In the same way as the success rate, an evaluator may be mistaken about the security of a device. Any model may seem safer than another especially when models are related. The figure 10 shows that the non-use of thresholding leads us to err on the model C which appears equivalent or worse than A.




(a) Without threshold. (b) With threshold $th = 0.2$.

**Fig. 10.** Conditional entropy improvements with threshold

## 5 Discussion

The goal of this section is to discuss the impact of the two improvement techniques to the side-channel framework of [21]. Basically, we warn that those two techniques can make an evaluation unfair if they convey an advantage dissymmetry to the attacker at the expense of the evaluator. To illustrate our argumentation, we resort to the same diagram as presented in [18]. The evaluator (noted $\mathsf{E}$) computes the $x$ position whereas the attacker (noted $\mathsf{A}$) computes the $y$ position. This type of diagram can be seen as real world *a posteriori* (*i.e. post mortem* using forensic terminology) results as a function of forecast *a priori* predictions.

For instance, it seems reasonable that some links between evaluation and attack metric can be put forward if they are computed on the same circuit with the same signal processing toolbox. Let us take the example of a chip we intend to stress, in order to exacerbate its leakage. Typically increasing the die's temperature $T$ (or its supply voltage) will increase its consumption and thus the amount of side-channel. In these experiments, we expect the condition entropy and the attack success rate to evolve in parallel. And even more, this tendency should be independent on the chosen leakage model. These conclusions hold provided the temperature of the device during the evaluation is the same as during the attack. This is depicted in Fig. 11(a). Now, if the evaluator performs the evaluation at nominal temperature but that the attacker is able to attack at higher temperatures, he will be undoubtedly advantaged. This characterizes typically an asymmetric relationship between the attacker and the evaluator;

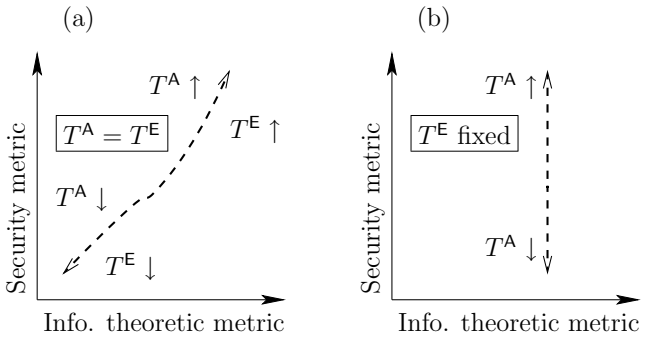



**Fig. 11.** Evaluation *vs* attack metrics diagram in a (a) symmetrical and (b) asymmetrical situation

we can also say that the attacker "cheats"[2] since she makes use of an extra degree of freedom unavailable to the attacker. This addition power changes the balance between the evaluation and the attack sides, as shown in Fig. 11(b). In concrete cases, for instance when the circuit to attack is a tamper-proof smartcard, the temperature, the voltage and all the operating conditions are monitored. Therefore, *a priori*, neither the evaluator nor the attacker can be advantaged.

The two contributions of this paper show however how to create a dissymmetry even when the evaluator and the attacker work in similar conditions (even better: on exactly the same traces [22]). The section 3 illustrates a situation of dissymmetry in *a priori* knowledge. It is sketched in Fig. 12(a). When the attacker knows that $\mathcal{L}_2$ is more adequate than $\mathcal{L}_1$, her attack will require in average less traces to be successful than that of the evaluator that is stuck at $\mathcal{L}_1$. The section 4 details the effect of a dissymmetry in expertise about the objects manipulated. Imagine a laboratory, such as an ITSEF, applying an attack "from the textbooks", as presented in Sec. 2. Then this ITSEF faces the risk of overestimating the security of its target of evaluation if it is not aware of the thresholding "trick".

In summary, we warn that the evaluator can be fooled into being too confident in a circuit's security, not having anticipated a weakness in the leakage model or in the attack processing stages. Thus routine in the evaluation can be harmful in the projected trust in the circuit. This fact is illustrated in Fig. 13; if an evaluator has evaluated many cryptographic components that were found to lay within the same region, *e.g.* corresponding to a common assurance evaluation level ("EAL" notion of the Common Criteria [5]), he will be tempted to limit its analysis to the measurement of an information theoretic metric. However, if an attacker comes up with either more knowledge about the device that the evaluator does or better signal processing techniques, the previsions of the ITSEF can turn out be too optimistic, since conservative (believing erroneously that "anyone attacks like I do").

[2] We mean that the attacker uses a strategy outside of the security model.

Our two attack improvement techniques do not invalidate the framework of [21], but simply warn that the notions it introduces shall be manipulated with care and not overseen. Otherwise, incorrect predictions can be done, thus ruining the confidence we can have in it.

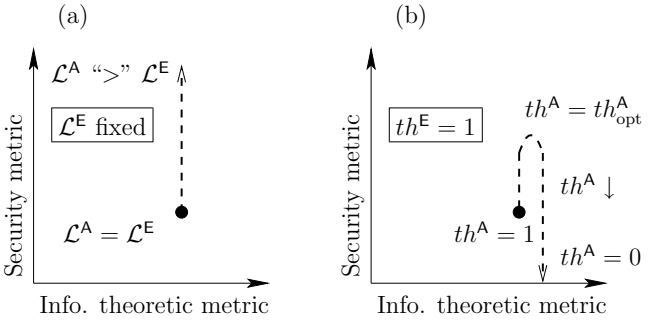



**Fig. 12.** Leakage *vs* attack diagram for our two attack optimizations: (a) adequate leakage models, (b) thresholding for non-informative samples

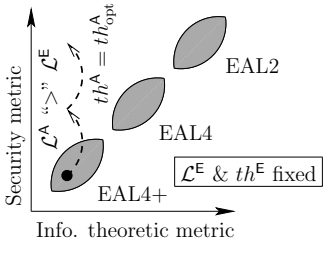



**Fig. 13.** Risk in trusting past experiences and relying on former evaluation *vs* attack unfounded empirical laws

## 6 Conclusions and Perspectives

In this article, we have put in practice the "unified framework for the analysis of side-channel key recovery attacks" [21], on the example of the "DPA contest" [22] real-world measurements. In our work, we place the evaluator, in charge of quantifying the amount of information leakage, and the attacker, in charge of exploiting the leakage to retrieve the secret, both on an equal footing regarding the experimental skill: both use the same acquisition campaign. We take advantage of these results to test another asymmetry in the evaluator / attacker couple, namely an asymmetry in initial knowledge. We concretely illustrate that any knowledge about the target's architecture or about the leakage structure in time can greater favor whichever actor takes advantage of it. From

the perspective of the attacker, we provide two improvements over straightforward attacks. First of all, we show that the care with which the sensitive variable is chosen influences qualitatively the attack outcome. In the studied hardware accelerator, the choice of the adequate sensitive variable is connected to its RTL architecture. Second, we illustrate that if the attacker knows that the leakage is localized in a few samples amongst the large number of sample making up the traces, she can intuit early (before the attack has converged to a unique key candidate) that some samples will mostly bring noise instead of useful information. Thanks to an innovative thresholding technique, we show that by ruling those samples out, the attacker can indeed speed-up the attack. Finally, we conclude about the usefulness of the framework [21] in terms of security predictions when both the evaluator and the attacker can play with the same degrees of freedom. However, we warn that an attacker, especially in cryptography, can always come up with "out-of-the-box" strategies that might empower her with an advantage unanticipated by the evaluator.

Further directions of research will consist notably in assessing the question of the existence of an optimal leakage model. Or is the best evaluation / attack obtained from an analysis involving concomitantly many leakage models? But in this case, what technique is the most suitable to combine coherently various leakage models? Techniques to exploit the side-channel information from multiple channels have already been presented. For instance, various EM are combined in [1]. Also, in [19], the simultaneous knowledge of power and EM leakage is taken advantage of to reduce the number of interactions with the cryptographic device under attack. However, in those two examples, the same leakage model is assumed. We endeavor in the future to get rid off this constraint. Additionally, we envision to port our techniques of leakage eigenvectors thresholding to attacks using an on-line profiling, such as the MIA [6], and to quantify the advantage it brings.

## References

1. Agrawal, D., Rao, J.R., Rohatgi, P.: Multi-channel attacks. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 2–16. Springer, Heidelberg (2003)
2. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template Attacks in Principal Subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006)
3. Brier, É., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
4. Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
5. Common Criteria consortium. Application of attack potential to smartcards v2-5 (April 2008), http://www.commoncriteriaportal.org/files/supdocs/CCDB-2008-04-001.pdf

6. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
7. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. Stochastic Methods. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 15–29. Springer, Heidelberg (2006)
8. Jolliffe, I.T.: Principal Component Analysis. Springer Series in Statistics (2002) ISBN: 0387954422
9. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
10. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
11. Micali, S., Reyzin, L.: Physically observable cryptography. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
12. Novak, R.: Side-channel attack on substitution blocks (Kunming, China). In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 307–318. Springer, Heidelberg (2003)
13. Novak, R.: Sign-based differential power analysis. In: Chae, K.-J., Yung, M. (eds.) WISA 2003. LNCS, vol. 2908, pp. 203–216. Springer, Heidelberg (2004)
14. Prouff, E., Rivain, M.: Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 499–518. Springer, Heidelberg (2009)
15. Schindler, W.: Advanced stochastic methods in side channel analysis on block ciphers in the presence of masking. Journal of Mathematical Cryptology 2(3), 291–310 (2008)
16. Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005)
17. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27 (1948)
18. Standaert, F.-X.: A Didactic Classification of Some Illustrative Leakage Functions. In: WISSEC, 1st Benelux Workshop on Information and System Security, Antwerpen, Belgium, November 8-9, p. 16 (2006)
19. Standaert, F.-X., Archambeau, C.: Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 411–425. Springer, Heidelberg (2008)
20. Standaert, F.-X., Koeune, F., Schindler, W.: How to Compare Profiled Side-Channel Attacks? In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 485–498. Springer, Heidelberg (2009)
21. Standaert, F.-X., Malkin, T., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
22. TELECOM ParisTech SEN research group. DPA Contest, 1st edn. (2008-2009), http://www.DPAcontest.org/

# Differential Fault Analysis of HC-128

Aleksandar Kircanski and Amr M. Youssef

Concordia Institute for Information Systems Engineering
Concordia University, Montreal, Quebec, Canada

**Abstract.** HC-128 is a high speed stream cipher with a 128-bit secret key and a 128-bit initialization vector. It has passed all the three stages of the ECRYPT stream cipher project and is a member of the eSTREAM software portfolio. In this paper, we present a differential fault analysis attack on HC-128. The fault model in which we analyze the cipher is the one in which the attacker is able to fault a random word of the inner state of the cipher but cannot control its exact location nor its new faulted value. To perform the attack, we exploit the fact that some of the inner state words in HC-128 may be utilized several times without being updated. Our attack requires about 7968 faults and recovers the complete internal state of HC-128 by solving a set of 32 systems of linear equations over $Z_2$ in 1024 variables.

## 1 Introduction

HC-128 [9] is a high speed stream cipher that has passed all the three phases of the ECRYPT eSTREAM competition and is currently a member of the eSTREAM software portfolio. The cipher design is suitable for modern super-scalar processors. It uses a 128-bit secret key and 128-bit initialization vector. At each step, it produces a 32-bit keystream output word. The inner state of the cipher is relatively large and amounts to 32768 bits, consisting of two arrays, $P$ and $Q$, of 512 32-bit words, each. HC-256 [10] is another cipher similar in structure to HC-128 but uses a 256-bit key and 256-bit IV. Only HC-128 participated in the eSTREAM competition. Along with the HC-128 proposal [9], an initial security analysis pointed out to a small bias in the least significant bit of the output words which allows a distinguisher based on $2^{151}$ outputs. Contrary to the claims of the cipher designer [9], in [6] it was shown that the distinguisher can be extended to other bits as well, due to the bias occurring in the operation of addition of three $n$-bit integers, which is utilized in HC-128. However, the initial security claim [9] that there exists no distinguisher for HC-128 that uses less than $2^{64}$ bits [9] has not been even nearly contradicted. In [11], Zenner presented a cache timing analysis of HC-256 but this attack is not directly applicable to HC-128.

In this paper, we present a differential fault analysis (DFA) attack on HC-128. Our attack requires around half the number of fault injections when compared to the attack [4] on RC4 in the equivalent fault model. In general, fault analysis attacks [2] fall under the category of implementation dependent attacks, which include side channel attacks such as timing analysis and power analysis. In fault

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 261–278, 2010.

analysis attacks, some kind of physical influence such as ionizing radiation is applied to the cryptographic device, resulting in a corruption of the internal memory or the computation process. The examination of the results under such faults often reveals some information about the cipher key or the secret inner state. The first fault analysis attack targeted the RSA cryptosystem in 1996 [2] and subsequently, fault analysis attacks were expanded to block ciphers (e.g., [1], [3]) and stream ciphers (e.g., [4], [5]). The threat of fault analysis attacks became more realistic after cheap and low-tech methods were found to induce faults.

## 2 HC-128 Specifications and Definitions

The following notation is used throughout the paper:

$+$ and $\boxminus$ : addition mod $2^{32}$ and subtraction mod 512.
$\oplus$: bit-wise XOR.
$\ll$, $\gg$: left and right shift, respectively, defined on 32 bit values.
$\lll$, $\ggg$: left and right rotation, respectively, defined on 32 bit values.
$x^b$: The $b^{th}$ bit of a word $x$.
$x^{c..b}$, where $c > b$: The word $x^c|x^{c-1}|..|x^b$.
$s'_i\langle P[f]\rangle, s'_i\langle Q[f]\rangle$: The faulty keystream, where the fault is inserted while the cipher is in state $i = 268$ and occurs at $P[f]$, $Q[f]$, respectively.

The secret inner state of HC-128 consists of the tables $P$ and $Q$, each containing 512 32-bit words. The execution of the cipher is governed by two public counters $i$ and $j$. The functions $g_1, g_2, h_1$ and $h_2$ in Fig. 1, are defined as follows:

$$g_1(x, y, z) = ((x \ggg 10) \oplus (z \ggg 23)) + (y \ggg 8),$$
$$g_2(x, y, z) = ((x \lll 10) \oplus (z \lll 23)) + (y \lll 8),$$
$$h_1(x) = Q[x^{7..0}] + Q[256 + x^{23..16}], \quad h_2(x) = P[x^{7..0}] + P[256 + x^{23..16}].$$

The key and IV initialization procedures are omitted since they are not relevant to our attack. We say that HC-128 is *in state* $i$, if $i$ steps have been executed,

**The HC-128 Keystream Generation Algorithm**

```
1: i = 0
2: repeat until enough keystream bits are generated
3:       j = i mod 512
4:       if (i mod 1024) < 512
5:            P[j] = P[j] + g1(P[j ⊟ 3], P[j ⊟ 10], P[j ⊟ 511])
6:            s_i = h1(P[j ⊟ 12]) ⊕ P[j]
7:       else
8:            Q[j] = Q[j] + g2(Q[j ⊟ 3], Q[j ⊟ 10], Q[j ⊟ 511])
9:            s_i = h2(Q[j ⊟ 12]) ⊕ Q[j]
10:      i = i + 1
```

**Fig. 1.** The HC-128 Keystream Generation Algorithm

counting from the initial inner state. We will denote the iteration in which the cipher goes from state $i$ to $i+1$ by *step* $i$.

**Definition 1.** *Let $P_s[j]$ denote the $P[j]$ value after it has been updated for $s$ times by the HC-128 KGA. Similarly, let $Q_s[j]$ denote the $Q[j]$ value after it has been updated for $s$ times, $j = 0, \ldots 511$.*

Definition 1 allows representing $P$ and $Q$ values at different cipher states as follows. If $s \in \{1, 2, \ldots\}$, $j \in \{0, \ldots 511\}$ and HC-128 is in state $i$, then

$$P[j] = \begin{cases} P_0[j], & i \in \{0, \ldots j\} \\ P_s[j], & i \in \{1024 \times (s-1) + j + 1, \ldots 1024 \times s + j\} \end{cases}$$

$$Q[j] = \begin{cases} Q_0[j], & i \in \{0, \ldots 512 + j\} \\ Q_s[j], & i \in \{1024 \times (s-1) + 512 + j + 1, \ldots \\ & \qquad 1024 \times s + 512 + j\} \end{cases}$$

To simplify the notation, regardless of whether $h_1$ or $h_2$ was called, the input value will be called the $h$ *input value*. Both functions take a 32-bit word on the input. However, only the least significant byte and third least significant byte of the input value are used. Let $x$ denote the input to the corresponding $h$ function called in step $i$. Define $A_i = x^{7..0}$ and $B_i = 256 + x^{23..16}$.

## 3 The Attack Overview

The fault model in which we analyze the cipher is the one in which the attacker is able to fault a random word of the inner state tables $P$ and $Q$ but cannot control its exact location nor its new faulted value. We also assume that the attacker is able to reset the cipher arbitrary number of times. To perform the attack, the faults are induced while the cipher is in state 268 instead of state 0. Such a choice reduces the number of required faults to perform the attack. Throughout the rest of the paper, whenever it is referred to a fault occurrence, it is assumed that the fault occurs when the cipher is in step $i = 268$. The aim of the attack is to recover the tables $P_1$ and $Q_1$, i.e. $P$ and $Q$ tables of the cipher in step $i = 1024$. Since the iteration function of HC-128 is $1-1$, the inner state can then be rewind to the initial state $i = 0$. The attack can now be summarized as follows. First, the faults are induced and the corresponding output is collected as follows:

- Repeat the following steps until all of the $P$, $Q$ words have been faulted at least once
  - Reset the cipher, iterate it for 268 steps and then induce the fault
  - Store the resulting faulty keystream words $s'_i$, $i = 268, \ldots 1535$

Then, the $h$ input values, as defined in the previous section, are recovered for certain steps as follows:

- Recover the $h$ input values in steps $512, \ldots 1023$ (details are provided in section 5.1)
- Recover a subset of the $h$ input values in steps $1024, \ldots 1535$ (the size of the recovered subset is quantified in section 5.2)

The inner state is recovered, bit by bit, in 32 phases. In phase $b = 0$, the bits $P_1^0[i], Q_1^0[i]$, $i = 0, \ldots 512$ are recovered. Then, in phases $b = 1, \ldots 30$, assuming the knowledge of $P_1^{b-1..0}[i], Q_1^{b-1..0}[i]$, $i = 0, \ldots 512$, the bits $P_1^b[i], Q_1^b[i]$ are recovered. In each phase, a system of linear equations over $Z_2$ in $P_1^b[i], Q_1^b[i]$ is generated as follows:

- Generate 512 equations of the form $(P_1^b[A_i] + P_1^b[B_i]) \oplus Q_1^b[i] = s_i^b$, $i = 512, \ldots 1023$ (section 5.3)
- Recover a subset of the $P_1^b[0], \ldots P_1^b[255]$ and a subset of $Q_1^b[0], \ldots Q_1^b[255]$ values and add the recovered information to the system (section 5.4)
- Generate more equations in $P_1^b[i], Q_1^b[i]$ values by considering the relations between faulty and non-faulty keystreams (section 5.5)
- Solve the obtained system of linear equations

Finally, the most significant bits of all the $P$ and $Q$ words are recovered by phase $b = 31$.

## 4 The Faulty Value Position and Difference

In this section, two algorithms are provided. The first one is used to recover the XOR difference between certain faulty and non-faulty inner state values after the fault has been induced and the cipher is iterated for certain number of steps. The algorithm is useful since the XOR differences between the non-faulty and the faulty inner state values is used to perform differential cryptanlaysis when the corresponding inner state values are *reused* in future cipher iterations. The second algorithm is used to recover the position of the induced fault. Before describing these two algorithms, an analysis of how the fault propagates as the cipher iterates is provided. Namely, we show that the position of the fault in the $P$ or the $Q$ tables uniquely determines the way by which the difference propagates through the corresponding table. This is due to the fact that, in HC-128, the update steps 5 and 8 in Fig. 1 use indices which are independent of the current state. Furthermore, although the indices used in the keystream output generation steps 6 and 9 depend on the inner state information, this does not impede the recovery of initial fault position, as will be shown below. To illustrate the above argument, assume that the fault occurred at $Q[f]$ while the cipher is in state $i = 268$. Since, according to line 5 of Fig. 1, the faulty value $Q'[f]$ is surely not referenced is during steps $i = 0, \ldots 511$, it follows that $P_1'[l] = P_1[l]$, $l = 0, \ldots 511$. Also, according to the update line 8 of Fig. 1, by which values $Q[j]$, $Q[j \boxminus 3]$, $Q[j \boxminus 10]$ and $Q[j \boxminus 511]$ are referenced, the first time in which $Q'[f]$ will be referenced is during the state in which $Q[f-1]$ is updated, i.e., in step $i = 512 + f - 1$. Thus, $Q_1[f-1] \neq Q_1'[f-1]$. More generally, define

$$\Delta Q_1[j] = \begin{cases} 0, & \text{if } Q_1[j] = Q_1'[j] \\ 1, & \text{if } Q_1[j] \neq Q_1'[j]. \end{cases} \quad (1)$$

Applying the same logic to follow the propagation until state 1024, for $1 \leq f \leq 501$, it is straightforward to check that

$$(\Delta Q_1[j])_{j=0}^{512} = \underbrace{00\ldots0}_{j=0,\ldots f-2}\ 110110110\ \underbrace{111\ldots11}_{j=f+8,\ldots 511}$$

The difference propagation in the inner state is also partially projected to the keystream. For instance, if the fault occurs at $Q[f]$, then $s_j = s'_j$ holds for $512 \le j < 512 + f - 1$. The first difference occurs at $i = 512 + j$, $j = f - 1$, after the value $Q[f-1]$ is affected and then referenced for the output in the same step. We define

$$\Delta s_i = \begin{cases} 0 & \text{if } s_i = s'_i \\ 1 & \text{if } s_i \neq s'_i \end{cases} \tag{2}$$

to track the difference propagation in the keystream output. In the presented reasoning, we implicitly assume that any difference in the right-hand side values of lines 5,6,8 or 9 of Fig. 1 always causes a difference in the corresponding left-hand sides. For 100,000 times, the inner state of HC-128 has been randomly initialized, iterated for 268 times and then faulted at random word. In all the 100,000 experiments, the correctness of our assumption was verified. The following Lemmas provide the complete difference propagation patterns for both the inner state and the keystream. The proofs are omitted since they are straightforward.

**Lemma 1.** *If the fault occurred in the $P$ table, its position $f$ uniquely determines the sequence $(\Delta P_1[j])_{j=0}^{512}$. Similarly, if the fault occurred in the $Q$ table, its position $f$ uniquely determines the sequence $(\Delta Q_1[j])_{j=0}^{512}$. The corresponding sequences, depending on the fault positions, are given in Table 1.*

**Table 1.** The effect of faults induced during state 268 on the $P$ and $Q$ tables

| Fault at $P[f]$ | $(\Delta P_1[j])_{j=0}^{512}$ |
|---|---|
| $f = 0$ | $1\ \underbrace{0\ldots0}_{j=1\ldots510}\ 1$ |
| $f \in \{1,\ldots 257\}$ | $\underbrace{0\ldots0}_{j=0\ldots f-1}\ 1\ \underbrace{0\ldots0}_{j=f+1\ldots511}$ |
| $f \in \{258,\ldots 264\}$ | $\underbrace{0\ldots0}_{j=0\ldots f-1}\ 10000000001001001001101101 10\ \underbrace{1\ldots1}_{j=f+28\ldots511}$ |
| $f \in \{265, 266, 267, 268\}$ | $\underbrace{0\ldots0}_{j=0\ldots f-1}\ 100100100110110110\ \underbrace{1\ldots1}_{j=f+18\ldots511}$ |
| $f \in \{269,\ldots 511\}$ | $\underbrace{0\ldots0}_{j=0\ldots f-2}\ 110110110\ \underbrace{1\ldots1}_{j=f+8\ldots511}$ |
| **Fault at $Q[f]$** | **$(\Delta Q_1[j])_{j=0}^{512}$** |
| $f = 0$ | $100100100110110110\ \underbrace{1\ldots1}_{j=18\ldots511}$ |
| $f \in \{1,\ldots 501\}$ | $\underbrace{0\ldots0}_{j=0\ldots f-2}\ 110110110\ \underbrace{1\ldots1}_{j=f+8\ldots511}$ |
| $f \in \{502,\ldots 508\}$ | $\underbrace{0\ldots0}_{j=0\ldots f-503}\ 100100100110110110\ \underbrace{1\ldots1}_{j=f-484\ldots511}$ |
| $f \in \{509, 510, 511\}$ | $\underbrace{0\ldots0}_{j=0\ldots f-510}\ 100100110110110\ \underbrace{1\ldots1}_{j=f-493\ldots511}$ |

**Lemma 2.** *If the fault occurred in the $P$ table, the fault position uniquely determines sequence $(\Delta s_i)_{i=256}^{511}|(\Delta s_i)_{i=1024}^{1279}$. Similarly, if the fault occurred in the $Q$ table, the fault position uniquely determines sequence $(\Delta s_i)_{i=512}^{1023}$. The corresponding sequences, depending on the fault position, are provided in Table 2.*

**Table 2.** The effect of faults induced during state 268 on the keystream

| Fault at $Q[f]$ | $(\Delta s_i)_{i=512}^{1023}$ |
|---|---|
| $f = 0$ | $100100100110110110\ \underbrace{1\ldots1}_{i=18\ldots511}$ |
| $f \in \{1,\ldots 499\}$ | $\underbrace{0\ldots0}_{i=0\ldots f-2}\ 110110110\ \underbrace{1\ldots1}_{i=f+8\ldots511}$ |
| $f = \{500, 501\}$ | $\underbrace{0\ldots0}_{i=0..f-501}\ 1\ \underbrace{0\ldots0}_{i=f-499\ldots498}\ 1101101101111$ |
| $f \in \{502,\ldots 508\}$ | $\underbrace{0\ldots0}_{i=0\ldots f-503}\ 101100100110110110\ \underbrace{1\ldots1}_{i=f-484\ldots511}$ |
| $f = \{509, 510, 511\}$ | $\underbrace{0\ldots0}_{0..f-510}\ 100100110110110\ \underbrace{1\ldots1}_{i=f-494\ldots511}$ |
| Fault at $P[f]$ | $(\Delta s_i)_{i=256}^{511}\|(\Delta s_i)_{i=1024}^{1279}$ |
| $f \in \{0,\ldots 247\}$ | $\underbrace{0\ldots0}_{i=0\ldots254+f}\ 110110110\ \underbrace{1\ldots1}_{i=263+f\ldots511}$ |
| $f \in \{248,\ldots 255\}$ | $\underbrace{0\ldots0}_{i=0\ldots254+f}\ \underbrace{110110110}_{i=255+f\ldots511}$ |
| $f = 256$ | $\underbrace{0\ldots0}_{i=0\ldots11}\ 1\ \underbrace{0\ldots0}_{i=13\ldots510}\ 1$ |
| $f = 257$ | $\underbrace{0\ldots0}_{i=0\ldots12}\ 1\ \underbrace{0\ldots0}_{i=14\ldots511}$ |
| $f \in \{258,\ldots 264\}$ | $\underbrace{0\ldots0}_{i=0\ldots f-247}\ 101100100110110110\ \underbrace{1\ldots1}_{i=f-228\ldots511}$ |
| $f \in \{265,\ldots 267\}$ | $\underbrace{0\ldots0}_{i=0\ldots f-254}\ 100100110110110\ \underbrace{1\ldots1}_{i=f-238\ldots511}$ |
| $f = 268$ | $\underbrace{0\ldots0}_{i=0\ldots11}\ 100100100110110110\ \underbrace{1\ldots1}_{i=30\ldots511}$ |
| $f \in \{269,\ldots 511\}$ | $\underbrace{0\ldots0}_{i=0\ldots f-258}\ 110110110\ \underbrace{1\ldots1}_{i=f-248\ldots511}$ |

### 4.1 Recovering the Differences between Faulty and Non-faulty Words

After a fault is introduced, other $P$ and $Q$ values are affected as the cipher iterates. In this section, we show how to derive the difference between these affected faulty values and their original counterparts. For illustration, assume that the fault occurred at $Q[f]$. In step $i = 512 + f - 1$, the faulty and non-faulty keystream words will be produced by

$$s_{512+f-1} = h_2(Q[f-13]) \oplus Q[f-1], \quad s'_{512+f-1} = h_2(Q'[f-13]) \oplus Q'[f-1].$$

However, since $Q'[f-13] = Q[f-13]$, it follows that $s_{512+f-1} \oplus s'_{512+f-1} = Q[f-1] \oplus Q'[f-1]$, which allows the recovery of $Q_1[f-1] \oplus Q'_1[f-1]$. For a fault position $f$, define the set $S(f)$ as follows:

$$l \in S(f) \Leftrightarrow 0 \leq l \leq 511 \text{ and } l \in \{f-1, f, f+2, f+3, f+5, f+6, f+8, f+9, f+10, f+13, f+16, f+19\} \quad (3)$$

where "+" and "-" denote addition and subtraction in the set of integers $Z$. In other words, given a fault at position $f$ in the $P$ or $Q$ tables, the set $S(f)$ defines the set of positions for which the difference from the original counterpart words can be recovered as given by the following two Lemmas.

**Lemma 3.** *Let HC-128 be in step* 268 *when a fault occurs in* $P[f]$, $269 \leq f \leq 511$. *Then, for* $l \in S(f)$, *we have*

$$P_1[l] \oplus P'_1[l] = s_l \oplus s'_l \quad (4)$$

*Proof.* The distribution of corrupted values in $P_1$ when $f \geq 269$ is provided in Table 1. If $l = f-1$, then

$$s_{f-1} = h_1(P_1[f-13]) \oplus P_1[f-1], \quad s'_{f-1} = h_1(P'_1[f-13]) \oplus P'_1[f-1]$$

According to Table 1, $P_1[f-13] = P'_1[f-13]$ and since there is no corrupted values in the $Q$ table, (4) follows. Similar proof follows for the other $l \in S(f)$ values, $269 \leq f \leq 511$.

**Lemma 4.** *Let HC-128 be in state* 268, *when a fault occurs in word* $Q[f]$, $0 \leq f \leq 501$. *Then, for* $l \in S(f)$, *we have*

$$Q_1[l] \oplus Q'_1[l] = s_{512+l} \oplus s'_{512+l} \quad (5)$$

The proof of Lemma 4 is analogous to the proof of Lemma 3. Note that the upper bound on $f$ in Lemma 4 allows a simplified treatment of recoverable differences. Namely, if the fault is on $Q[f]$ for $f > 501$, the propagation starts as early as in step $i = 512$ and the set of recoverable differences differs from $S(f)$.

Given the fault position $P[f]$ or $Q[f]$, the above two Lemmas establish that for $l \in S(f)$, $P[l] \oplus P'[l]$ or $Q[l] \oplus Q'[l]$ can be recovered. A converse question can also be posed: Given a position, say $Q[l]$, which fault positions in the $Q$ table will allow the recovery of $Q_1[l] \oplus Q'_1[l]$? For that purpose, it is convenient to define the set $S_Q^{-1}(l)$ for $0 \leq l \leq 511$ as follows

$$f \in S_Q^{-1}(l) \Leftrightarrow 0 \leq f \leq 501 \text{ and } f \in \{l+1, l, l-2, l-3, l-5, l-6, l-8, l-9, l-10, l-13, l-16, l-19\} \quad (6)$$

Now, given a position $Q[l]$, the set $S_Q^{-1}(l)$ provides all fault positions such that $Q_1[l] \oplus Q'_1[l] = s_{512+l} \oplus s'_{512+l}$ according to Lemma 4. Similarly, given a position $268 \leq l \leq 511$ in the $P$ table, the set $S_P^{-1}(l)$ defined by

$$f \in S_P^{-1}(l) \Leftrightarrow 269 \leq f \leq 511 \text{ and } f \in \{l+1, l, l-2, l-3, l-5, l-6, l-8, l-9, l-10, l-13, l-16, l-19\} \quad (7)$$

provides the fault positions $f$ such that $P_1[l] \oplus P'_1[l] = s_l \oplus s'_l$ can be recovered according to Lemma 3.

### 4.2 Recovering the Position of the Fault

In this section, we provide an algorithm to deduce the position where the fault occurred. Since, according to Lemmas 1 and 2, the fault position uniquely determines the corresponding sequences, the following functions can be defined:

$$\phi^P : f \mapsto (\Delta s_i)_{i=256}^{511} | (\Delta s_i)_{i=1024}^{1279}, \quad \phi^Q : f \mapsto (\Delta s_i)_{i=512}^{1023}$$

The functions are explicitly given in Table 2. By checking that no two right-hand side sequences in both parts of the Table 2 are equal, it follows that

**Lemma 5.** *The functions $\phi^P$ and $\phi^Q$ are 1-1.*

Let $\Delta^P = \phi^P(\{0, \dots 511\})$ and $\Delta^Q = \phi^Q(\{0, \dots 511\})$. If the fault does not cause $(\Delta s_i)_{512}^{1023} \in \Delta^P$ and $(\Delta s_i)_{i=256}^{511} | (\Delta s_i)_{i=1024}^{1279} \in \Delta^Q$ at the same time, which, as will be shown, happens with negligible probability, then Algorithm 1 returns the fault position.

**Algorithm 1. Fault Position Recovery**

**INPUT:** $(\Delta s_i)_{i=256}^{1279} = (s_i \oplus s'_i)_{i=256}^{1279}$
**OUTPUT:** The position where the fault occurred

1: If both $(\Delta s_i)_{512}^{1023} \in \Delta^P$ and $(\Delta s_i)_{i=256}^{511} | (\Delta s_i)_{i=1024}^{1279} \in \Delta^Q$, return *undefined*
2: If $(\Delta s_i)_{i=512}^{1023} \in \Delta^P$, return $\phi_P^{-1}((\Delta s_i)_{i=512}^{1023})$
3: If $(\Delta s_i)_{i=256}^{511} | (\Delta s_i)_{i=1024}^{1279} \in \Delta^Q$, return $\phi_Q^{-1}((\Delta s_i)_{i=256}^{511} | (\Delta s_i)_{i=1024}^{1279})$

From line 1 of Algorithm 1, if there is conflicting information on whether the fault occurred in the $P$ or the $Q$ table, the algorithm returns *undefined*. To estimate the probability of this unwanted event, let $F_{P[f]}$ and $F_{Q[f]}$ denote the event that the fault occurs at position $P[f]$ and $Q[f]$, respectively. Let $U$ denote the event that Algorithm 1 returns *undefined*. Then we have

$$\text{Prob}[U] = \sum_{f=0}^{511} \text{Prob}[U \cap F_{P[f]}] + \sum_{f=0}^{511} \text{Prob}[U \cap F_{Q[f]}] =$$

$$\frac{1}{1024}\left(\sum_{f=0}^{511} \text{Prob}[U|F_{P[f]}] + \sum_{f=0}^{511} \text{Prob}[U|F_{Q[f]}]\right) \tag{8}$$

where $\text{Prob}[U \cap F_{P[f]}] = \text{Prob}[F_{P[f]}]\text{Prob}[U|F_{P[f]}]$ and also $\text{Prob}[U \cap F_{Q[f]}] = \text{Prob}[F_{Q[f]}]\text{Prob}[U|F_{Q[f]}]$. To expand the probability $\text{Prob}[U|F_{P[f]}]$, let $n_0$ and $n_1$ denote the number of faulty values among the values $P[0], \dots P[255]$ and $P[256] \dots P[511]$, respectively, at state 512, given that the fault occurred at $P[f]$. Also, let $p = \frac{n_0+n_1}{256} - \frac{n_0 n_1}{256^2}$. If $n(\delta'_i)$ is the number of 1 values in a 512-element sequence $\delta'_i \in \Delta_Q$, then

$$\text{Prob}[U|F_{P[f]}] = \sum_{\delta'_i \in \Delta^Q} \text{Prob}[(\Delta s_i)_{i=512}^{1023} = \delta'_i | F_{P[f]}] = \sum_{\delta'_i \in \Delta^Q} p^{n(\delta'_i)}(1-p)^{512-n(\delta'_i)}$$

As for the probability $\text{Prob}[U|F_{Q[f]}]$, let $n_0$ and $n_1$ denote the number of faulty words among $Q[0], \ldots Q[255]$ and $Q[256], \ldots Q[511]$, respectively, at state 268, given that the fault occurred at $Q[f]$. Let $n_2$ and $n_3$ denote the number of faulty words among $Q[0], \ldots Q[255]$ and among $Q[256], \ldots Q[511]$, respectively, at state 1024, given that the fault occurred at $Q[f]$. Let $p_0 = \frac{n_0+n_1}{256} - \frac{n_0 n_1}{256^2}$ and $p_1 = \frac{n_2+n_3}{256} - \frac{n_2 n_3}{256^2}$. If $m_0(\delta'_i)$ and $m_1(\delta'_i)$, denote the number of 1 values among $\delta'_{12}, \ldots \delta'_{255}$ and $\delta'_{256}, \ldots \delta'_{511}$, respectively, where $\delta'_i \in \Delta_P$, then

$$\text{Prob}[U|F_{Q[f]}] = \sum_{\delta'_i \in \Delta^P} \text{Prob}[(\Delta s_i)_{i=256}^{511} | (\Delta s_i)_{i=1024}^{1279} = \delta'_i | F_{Q[f]}] =$$

$$= \sum_{\delta'_i \in \Delta^P} p_0^{m_0(\delta'_i)} (1-p_0)^{244-m_0(\delta'_i)} p_1^{m_1(\delta'_i)} (1-p_1)^{256-m_1(\delta'_i)}$$

Calculating the sets $\Delta_P$ and $\Delta_Q$ and substituting the corresponding values using Table 2 allows the computation of the sums in Eq. (8) as $\frac{1}{1024} \sum_{f=0}^{511} \text{Prob}[U|F_{P[f]}] = 2^{-66.293}$ and $\frac{1}{1024} \sum_{f=0}^{511} \text{Prob}[U|F_{Q[f]}] = 2^{-30.406}$. Thus, the probability that Algorithm 1 returns *undefined* as fault position is is $\text{Prob}[U] = 2^{-30.406}$.

## 5 Using DFA to Generate Equations

As described in section 3, the attack is performed by introducing faults until every $P$ and $Q$ word is faulted. Let $T$ be the number of fault injections required to fault each of the 1024 words in the $P$ and $Q$ tables at least once. The expected number of required faults, $E(T)$, is given by $E(T) = 7698.4$ (see the coupons' collector problem in [7].) After inducing that number of faults, the average number of faults at a particular word $P[i]$ or $Q[i]$ will be $7698.4/1024 \approx 7.52$. As stated in section 3, the attack proceeds in 32 phases. Each phase $b$ relies on the knowledge of $P_1^{b-1..0}[i]$, $Q_1^{b-1..0}[i]$, $i = 0, \ldots 511$, recovered in previous phases. Only the first phase, $b = 0$, does not require any previous bit knowledge. In each phase, a linear system of equations over $Z_2$ in $P_1^b[i]$, $Q_1^b[i]$, $i = 0, \ldots 511$ is generated and solved. Phase $b = 31$ proceeds with minor modifications compared to phases $0 \leq b \leq 30$, as explained below.

### 5.1 The Recovery of $h$ Input Values for Steps $512, \ldots 1023$

In every HC-128 step, one of the two $h$ functions is called, i.e., either $h_1$ or $h_2$. The input for the $h$ functions is a 32-bit value, out of which only 16 bits, $A_i, B_i$, play a role in the computation. In this section, we describe a method to recover all of the $A_i$, $B_i$ values, for $i = 512, \ldots 1023$.

To recover $A_i$, assume that the fault occurred at $P[f]$ while the cipher was in state 268. As can be seen from Table 1, if $1 \leq f \leq 255$, then as the cipher iterates through steps $i = 512, \ldots 1023$, no other $P$ values gets corrupted. Also, the $Q$ table does not get corrupted. Thus, in case $1 \leq f \leq 255$, the non-faulty and the faulty keystream words in step $512 \leq i \leq 1023$ are

$$s_i = (P_1[A_i] + P_1[B_i]) \oplus Q_1[j], \; s'_i\langle P[f]\rangle = (P'_1[A_i] + P_1[B_i]) \oplus Q_1[j]$$

Since $P_1'[A_i] \neq P_1[A_i]$ implies that $A_i = f$, then we have

$$s_i \neq s_i'\langle P_1[f]\rangle \Rightarrow A_i = f \tag{9}$$

In case $f = 0$, the fault does propagate to $P[511]$ and if $s_i \neq s_i'\langle P[0]\rangle$, then it is unclear whether $A_i = 0$ or $B_i = 511$, or both equalities hold. However, if there exists no faulty keystream for $1 \leq f \leq 255$ such that (9) is true, then $A_i = 0$. As for $B_i$, assume that a fault is inserted at word $P[f]$, $256 \leq f \leq 268$, while the cipher is in state 268. From Table 1, it is clear that at state 1024, none of the $P[0], \ldots P[f-1]$ values will be corrupted and the value $P[f]$ will necessarily be corrupted. Similarly, if the fault is inserted at $P[f]$ where $269 \leq f \leq 511$, none of the values $P[0], \ldots P[f-2]$ get corrupted and the value $P[f-1]$ will necessarily be corrupted. Thus, if $f_{max}$ denotes the maximal $f$ such that $s_i \neq s_i\langle P[f]\rangle$, then

$$B_i = \begin{cases} f_{max} & \text{if } f_{max} \in \{256, \ldots 268\} \\ f_{max} - 1 & \text{if } f_{max} \in \{269, \ldots 510\} \end{cases}$$

Finally, if $f_{max} = 511$, it is not clear whether $B_i = 510$ or $B_i = 511$. To differentiate between these two cases, it should be verified whether $s_i \neq s_i'$ also holds for any $f$ which does not corrupt $P[511]$, for instance for $f = 507$. The recovery of $A_i, B_i$, for all $512 \leq i \leq 1023$ is given by Algorithm 2.

**Algorithm 2. Recovery of $A_i$ and $B_i$, for some $i = 512, \ldots 1023$**

**INPUT:** Step $i \in \{512, \ldots 1023\}$
**OUTPUT:** $A_i$, $B_i$

1: If exists $1 \leq f \leq 255$ such that $s_i \neq s_i'\langle P[f]\rangle$: $A_i = f$
2: else $A_i = 0$
3: Find $f_{max}$, the maximum $f$ such that $s_i \neq s_i\langle P[f]\rangle$
4: If $256 \leq f_{max} \leq 268$: $B_i = f_{max}$
5: else if $269 \leq f_{max} \leq 510$: $B_i' = f_{max} - 1$
6: else if $s_i = s_i\langle P[507]\rangle$: $B_i = 510$
7: else $B_i = 511$
9: Return $A_i$, $B_i$

Given the definition of $h_2$ and by noting that $j = i \bmod 512$, the recovered $A_i$ and $B_i$ values are in fact

$$A_i = \begin{cases} Q_0^{7..0}[j \boxminus 12] & \text{if } i \in \{512..523\} \\ Q_1^{7..0}[j \boxminus 12] & \text{if } i \in \{524..1023\} \end{cases} \tag{10}$$

$$B_i = \begin{cases} Q_0^{23..16}[j \boxminus 12] + 256 & \text{if } i \in \{512..523\} \\ Q_1^{23..16}[j \boxminus 12] + 256 & \text{if } i \in \{524..1023\} \end{cases} \tag{11}$$

### 5.2 The Recovery of the $h$ Input Values for Steps $1024, \ldots 1535$

In this subsection, $A_i$, $B_i$ values for a subset of $i = 1024, \ldots 1535$ are recovered. While $B_i$ values will be recovered by a method similar to the one from the previous subsection, the same method is not applicable for $A_i$ recovery and we will utilize the *reuse* of inner state words to recover the $A_i$ values.

As for the recovery of $B_i$ for $i = 1024, \ldots 1535$, from Table 1 it can be observed that if for some $1 \leq f \leq 501$, $Q[f]$ is faulted at step 268, the value $Q[f+7]$ will remain unchanged and the values $Q[f+8], \ldots Q[511]$ will surely be corrupted. Thus, if $f_{min}$ denotes the minimal $249 \leq f \leq 501$ such that $s_i = s'_i\langle Q[f]\rangle$, then $B_i = f_{min}+7$. Also, since $Q[509]$, $Q[510]$ and $Q[511]$ will get corrupted regardless of the fault position in $Q$, it is not possible to distinguish which of values 509, 510 or 511 $B_i$ was equal to.

Thus, if for given step $i$, $B_i < 509$ holds, $B_i$ will be recovered. Moreover, if $B_i < 500$, $Q_1^{7..0}[B_i]$ will be recovered by (11). Assuming that $B_i < 500$ for step $i$, then $A_i$ can be recovered as follows. Consider the faulty keystream $s'_i\langle Q[f]\rangle$ where $f \in S_Q^{-1}(B_i)$. According to Lemma 4 and (6)

$$Q_1[B_i] \oplus Q'_1[B_i] = s_{512+B_i} \oplus s'_{512+B_i}$$

Thus, $Q_1^{'7..0}[B_i]$ can be recovered by $Q_1^{'7..0}[B_i] = Q_1^{7..0}[B_i] \oplus s_{512+B_i}^{7..0} \oplus s_{512+B_i}^{'7..0}$. After being used in step $512 + B_i$, the value $Q[B_i]$ is *reused* in step $i$ as follows

$$s_i = (Q_1[A_i] + Q_1[B_i]) \oplus P_2[j],\ s'_i\langle Q[f]\rangle = (Q'_1[A_i] + Q'_1[B_i]) \oplus P'_2[j]$$

If $257 \leq f \leq 501$, $Q_1[A_i] = Q'_1[A_i]$ holds according to Table 1. Also, the $P$ table remains uncorrupted and thus $P_2[j] = P'_2[j]$. Thus, focusing on the least significant byte and XORing the previous two values yields

$$s_i^{7..0} \oplus s_i^{'7..0}\langle Q[f]\rangle = (Q_1^{7..0}[A_i] + Q_1^{7..0}[B_i]) \oplus (Q_1^{7..0}[A_i] + Q_1^{'7..0}[B_i]) \qquad (12)$$

Since $s_i^{7..0} \oplus s_i^{'7..0}$, $Q_1^{7..0}[B_i]$ and $Q_1^{'7..0}[B_i]$ are known, (12) represents a test that allows eliminating some wrong candidates for $Q_1^{7..0}[A_i]$ value. One test of the form (12) will be generated for each faulty instance for which the fault position is $Q[f]$, where $f \in S_Q^{-1}(B_i)$. Consequently, an $0 \leq A_i \leq 255$ can be discarded if the corresponding $Q_1^{7..0}[A_i]$ recovered by (11) does not satisfy (12).

The test (12) can be reformulated so that the third least significant byte is used as follows

$$s_i^{23..16} \oplus s_i^{'23..16} =$$
$$(Q_1^{23..16}[A_i] + Q_1^{23..16}[B_i] + \sigma_i) \oplus (Q_1^{23..16}[A_i] + Q_1^{'23..16}[B_i] + \sigma'_i) \qquad (13)$$

where $\sigma_i$ is a carry corrector defined to be 1 if $Q_1^{15..0}[A_i] + Q_1^{15..0}[B_i] \geq 2^{16}$ and 0 otherwise. Another carry corrector, $\sigma'_i$, is defined analogously. The value $Q_1^{'23..16}[B_i]$ is obtained in the same way as the value $Q_1^{'7..0}[B_i]$ above. If $0 \leq A_i \leq 255$ and the corresponding $Q_1^{23..16}[A_i]$ are substituted in (13) and none of $\sigma_i, \sigma'_i \in \{0, 1\}$ satisfy the test, then $A_i$ is discarded.

**Algorithm 3. Recovery of $A_i$ and $B_i$, for some $i = 1024, \ldots 1535$**

**INPUT:** Step $i \in \{1024, \ldots 1535\}$
**OUTPUT:** $A_i$ or $undef$, $B_i$ or $undef$

1: Calculate $F = \{257 \leq f \leq 501 | s_i = s'_i \langle Q[f] \rangle\}$
2: If $|F| = 0$: $B_i = undef$
3: Else $B_i = min(F) + 7$
4: If $B_i > 500$ $A_i = undef$; Return $A_i$, $B_i$
5: Else: let $Cand(A_i) = \{0, 1, ..255\}$
6: For each $f \in S_Q^{-1}(B_i)$
7: Deduce $Q_1'^{7..0}[B_i] = Q_1^{7..0}[B_i] \oplus s_{512+B_i}^{7..0} \oplus s_{512+B_i}'^{7..0}$
8: For $A_i = 0, \ldots 255$
9: If $s_i^{7..0} \oplus s_i'^{7..0} \neq (Q_1^{7..0}[A_i] + Q_1^{7..0}[B_i]) \oplus (Q_1^{7..0}[A_i] + Q_1'^{7..0}[B_i])$
10: Eliminate $A_i$ from $Cand(A_i)$
11: If for every $\sigma_i, \sigma'_i \in \{0, 1\}$, $s_i^{23..16} \oplus s_i'^{23..16} \neq d$, where
12: $d = (Q_1^{23..16}[A_i] + Q_1^{23..16}[B_i] + \sigma_i) \oplus (Q_1^{23..16}[A_i] + Q_1'^{23..16}[B_i] + \sigma'_i)$
13: Eliminate $A_i$ from $Cand(A_i)$
14: If $|cand(A_i)| = 1$, let $A_i$ be the unique $cand(A_i)$ member
15: Else: $A_i = undef$
16: Return $A_i$, $B_i$

In what follows, we estimate the expected number of steps for which both the $A_i$ and $B_i$ values are recovered by the presented method. Let $1024 \leq i \leq 1535$ be a step of HC-128. If, for example, for step $i$, $257 \leq B_i \leq 492$, then the $B_i$ value will surely be recovered as provided by the above method. Furthermore, for such a particular value $B_i$, $|S_Q^{-1}(B_i)| = 12$ will hold. Since for each $f \in S_Q^{-1}(B_i)$ around 7.52 faults occur at $Q[f]$, as shown at the beginning of section 5, around $7.52 \times 12 = 90.24$ tests given by Eq. (12) and the same number of tests given by Eq. (13) will be applied to the set of candidates for $A_i$. According to our experimental results, such a number of tests is sufficient to discard all the false candidates for $A_i$. In particular, an experiment in which Algorithm 3 was executed for all 512 steps $i \in \{1024, \ldots 1535\}$ for 10,000 times, with random HC-128 instantiations, was conducted. On average, in 472.7 out of the 512 steps, both $A_i$ and $B_i$ values were recovered.

### 5.3 Equations of the Form $P_1^b[A_i] \oplus P_1^b[B_i] \oplus Q_1^b[j] = s_i^b \oplus c_{i,b}$

After the steps given by subsections 5.1 and 5.2 have been executed, the attack proceeds in 32 phases, each consisting of 3 parts, as presented by the attack overview in section 3. In this subsection, the first part of $b$-th attack phase is presented.

The first part of $b$-th phase, in which starting 512 equations are generated, proceeds as follows. In steps $i \in \{512, \ldots 1023\}$, the keystream output word is generated as $(P_1[A_i] + P_1[B_i]) \oplus Q_1[j] = s_i$ where $j = i \bmod 512$. Since $A_i$ and $B_i$ for $i \in \{512, \ldots 1023\}$ have been recovered in subsection (5.1), focusing on the $b$-th bit yields 512 bits equations of the form

$$P_1^b[A_i] \oplus P_1^b[B_i] \oplus Q_1^b[j] = s_i^b \oplus c_{i,b}, i = 512, \ldots 1023 \tag{14}$$

where $c_{i,b}$ is a known carry corrector which is equal to 1 if there is carry in $(P_1^{b-1..0}[A_i] + P_1^{b-1..0}[B_i])$ and 0 otherwise. In case $b \in \{0, \ldots 7\}$ or $b \in \{16, \ldots 23\}$, relying on the knowledge obtained by (10) and (11), the system can be extended by adding information $Q_1^b[w] = a_w, w = 0, \ldots 499$, regarded as equations. However, for $b \notin \{0, \ldots 7, 16, \ldots 23\}$ such equations are unavailable. Hence, a method to systematically add more equations to the system (14) that works for all $b = 0, \ldots 31$, i.e., that makes the corresponding system of rank 1024, is necessary. In order to provide a generic treatment for all $b$ values, in what follows, equations derived from information given by (10) and (11) will not be utilized.

### 5.4 Recovering Bits $P_1^b[0], \ldots P_1^b[255]$ and $Q_1^b[0], \ldots Q_1^b[255]$

In the second part of the $b$-th phase of the attack, the system of equations given by (14) is expanded. Note that in steps $512 \leq i \leq 1023$, the output is generated by $s_i = (P_1[A_i] + P_1[B_i]) \oplus Q_1[j]$, whereas in steps $1024 \leq i \leq 1535$, the output is generated by $s_i = (Q_1[A_i] + Q_1[B_i]) \oplus P_2[j]$. The idea is to corrupt $P_1[B_i]$ and $Q_1[B_i]$ in the previous two relations and recover $P_1[A_i]$ and $Q_1[A_i]$ by observing how these values react to addition of different values. The difference of the corrupted values is controlled by utilizing the *reuse* of $P_1[B_i]$ and $Q_1[B_i]$ over different states of the cipher. The analysis results in the recovery of a subset of the $P_1^b[0], \ldots P_1^b[255]$ and also a subset of the $Q_1^b[0], \ldots Q_1^b[255]$ values.

As for recovering $P_1^b[0], \ldots P_1^b[255]$, let $512 \leq i \leq 1023$ and $268 \leq B_i \leq 511$. Consider a fault at position $P[f]$, so that $f \in S_P^{-1}(B_i)$. Using Lemma 3 and (7), define $\delta = s_{B_i} \oplus s'_{B_i} = P_1[B_i] \oplus P'_1[B_i]$. Assume that for the faulty cipher instance in question, $\delta^b = 1$. Consider the difference

$$\Delta = s_i \oplus s'_i = (P_1[A_i] + P_1[B_i]) \oplus (P_1[A_i] + P'_1[B_i]),$$

and denote by $c_b$ and $c'_b$ the carry from the $b-1$ to $b$-th bit in the sums $P_1[A_i] + P_1[B_i]$ and $P_1[A_i] + P'_1[B_i]$, respectively. If $c_b = c'_b$, then the bit $P^b[A_i]$ is recovered as follows

$$P_1^b[A_i] = \begin{cases} \delta^{b+1} \oplus \Delta^{b+1}, & \text{if } c_b = c'_b = 0 \\ \delta^{b+1} \oplus \Delta^{b+1} \oplus 1, & \text{if } c_b = c'_b = 1 \end{cases} \tag{15}$$

If $c_b \neq c'_b$, the bit $P_1^b[B_i]$ is not uniquely determined and will not be recovered.

To recover $P_1^b[A_i]$, the explained procedure is repeatedly applied using each fault occurring at $P[f]$, $f \in S_P^{-1}(B_i)$. Let $p_1 = Prob[\delta^b = 1]$ and $p_2 = Prob[c_b = c'_b]$, then the probability of success can be lower bounded as follows:

$$Prob[P_1^b[A_i] \text{ recovery succeeds}] \geq \sum_{B_i=256}^{511} \frac{1}{256} \left(1 - (1 - p_1 p_2)^{|S_P^{-1}(B_i)|}\right) \tag{16}$$

The values $|S_P^{-1}(B_i)|$ are given by Table 4 and $Prob[\delta^b = 1] = \frac{1}{2}$. As for $Prob[c_b = c'_b]$, it can be modelled as the probability that there exists a carry at bit $b$ in two random sums [8]. It achieves a minimum for $b = 31$ and thus the lower bound for the success probability over possible bit positions is given by $Prob[\text{Recovery of } P_1^b[A_i] \text{ succeeds}] \geq 0.908$.

Now, the probability that for some particular $k \in \{0, \ldots 255\}$, the value $P_1^b[k]$ will not be recovered in some particular step $i$ is then less than $1 - \frac{1}{256} \times 0.908$. Let $Z_k = 1$ if $P_1^b[k]$ has not been recovered after applying the algorithm for all steps $i = 512, \ldots 1023$. Otherwise, let $Z_k = 0$. The number of $P_1^b[k]$, $0 \leq k \leq 255$ values not recovered can then be estimated as

$$E(\sum_{k=0}^{255} Z_k) = \sum_{k=0}^{255} E(Z_k) \leq 256 \times (1 - \frac{1}{256} \times 0.908)^{512} = 41.511 \qquad (17)$$

Thus, the method presented in this section when applied on steps $i = 512, \ldots 1023$, is expected to recover more than $256 - 41.511 = 214.49$ of the $P_1^b[0], \ldots P_1^b[255]$ values. The exact procedure is presented by Algorithm 4.

**Algorithm 4. Recovery of $P_1^b[A_i]$, for some $i = 512, \ldots 1023$**

**INPUT:** Step $i \in \{512, \ldots 1023\}$
**OUTPUT:** Bit $P_1^b[A_i]$, or *undef*

1: For every faulty keystream, where the fault occurred at $P[f]$, $f \in S_P^{-1}(B_i)$
2: Calculate $\delta = s_{B_i} \oplus s'_{B_i}$ and $\Delta = s_i \oplus s'_i$
3: If $P_1^{b-1..0}[x] + P_1^{b-1..0}[B_i] < 2^b$, set $c_b = 0$, else set $c_b = 1$
4: If $P_1^{b-1..0}[x] + P_1'^{b-1..0}[B_i] < 2^b$, set $c'_b = 0$, else set $c'_b = 1$
5: If $c_b = c'_b$:
6: Return $P_1^b[A_i]$ calculated according to (15)
7: Return *undef*

As for recovering $Q_1^b[0], \ldots Q_1^b[255]$, an analogous technique, applied on steps $i = 1024, \ldots 1535$, is used. The exact procedure is presented by Algorithm 5. The expected number of recovered values is calculated analogously to (16), whereas it needs to be taken into account that $A_i$ and $B_i$, $i \in \{1024, \ldots 1535\}$, need to be successfully recovered by subsection 5.2. The $|S_Q^{-1}(B_i)|$ values, given at Table 3, are more favorable than the corresponding $|S_P^{-1}(B_i)|$ values in (16). The expected number of $Q_1^b[0], \ldots Q_1^b[255]$ values to be recovered is 218.01.

### 5.5 Utilizing Equations in Faulty Bits

In this subsection, the system constructed in subsections 5.3 and 5.4 is expanded further for the purpose of attaining the full rank of the system. Consider the faulty output word in steps $512, \ldots 1023$, $s'_i = h_2(Q'_1[j \boxminus 12]) \oplus Q'_1[j]$. Evidently, regarding the previous relation as an equation is useless since it includes faulty inner state bits. Below, a method to transform the faulty inner state bits participating in the equation to original inner state bits is provided. Again, the *reuse* of inner state words is utilized.

**Algorithm 5. Recovery of $Q_1^b[A_i]$, for some $i = 1024, \ldots 1535$**

**INPUT:** Step $i \in \{1024, \ldots 1535\}$
**OUTPUT:** Bit $Q_1^b[A_i]$, or *undef*

1: If $A_i$ or $B_i$ are unknown, return *undef*
2: For every faulty keystream, where the fault occurred at $Q[f]$, $Q \in S_Q^{-1}(B_i)$
3: Calculate $\delta = s_{512+B_i} \oplus s'_{512+B_i}$ and $\Delta = s_i \oplus s'_i$
4: If $Q_1^{b-1..0}[x] + Q_1^{b-1..0}[B_i] < 2^b$, set $c_b = 0$, else set $c_b = 1$
5: If $Q_1^{b-1..0}[x] + Q_1'^{b-1..0}[B_i] < 2^b$, set $c'_b = 0$, else set $c'_b = 1$
6: If $c_b = c'_b = 0$: Return: $\delta^{b+1} \oplus \Delta^{b+1}$
6: If $c_b = c'_b = 1$: Return $\delta^{b+1} \oplus \Delta^{b+1} \oplus 1$
8: Return *undef*

**Table 3.** The number of fault positions which allow the recovery of $Q_1[l] \oplus Q'_1[l]$

| $l$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\lvert S_Q^{-1}(l)\rvert$ | 2 | 2 | 3 | 4 | 4 | 5 | 6 | 6 | 7 | 8 |
| $l$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | $19, \ldots 500$ |
| $\lvert S_Q^{-1}(l)\rvert$ | 9 | 9 | 9 | 10 | 10 | 10 | 11 | 11 | 11 | 12 |
| $l$ | 501 | 502 | 503 | 504 | 505 | 506 | 507 | 508 | 509 | 510 | 511 |
| $\lvert S_Q^{-1}(l)\rvert$ | 11 | 10 | 10 | 9 | 8 | 8 | 7 | 6 | 6 | 5 | 4 |

**Table 4.** The number of fault positions which allow the recovery of $P_1[l] \oplus P'_1[l]$

| $l$ | 268 | 269 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | 277 | 278 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\lvert S_P^{-1}(l)\rvert$ | 1 | 2 | 2 | 3 | 4 | 4 | 5 | 6 | 6 | 7 | 8 |
| $l$ | 279 | 280 | 281 | 282 | 283 | 284 | 285 | 286 | 287 | $288, \ldots 510$ | 511 |
| $\lvert S_P^{-1}(l)\rvert$ | 9 | 9 | 9 | 10 | 10 | 10 | 11 | 11 | 11 | 12 | 11 |

Let the fault position be $Q[f]$, where $f \in S_Q^{-1}(l)$ and $244 \leq l \leq 499$. The non-faulty and the faulty instances of the cipher in step $i_0 = 512 + l + 12$ are

$$s_{i_0} = h_2(Q_1[l]) \oplus Q_1[l+12],\ s'_{i_0} = h_2(Q'_1[l]) \oplus Q'_1[l+12] \tag{18}$$

Note that $Q_1^{7..0}[l] = A_{i_0}$ and $Q_1^{23..16}[l] = B_{i_0} - 256$ are known according to subsection 5.1 and that the difference $Q'_1[l] \oplus Q_1[l]$ can be calculated as $\delta = Q'_1[l] \oplus Q_1[l] = s_{512+l} \oplus s'_{512+l}$, according to Lemma 4. Thus, $A'_{i_0}$ and $B'_{i_0}$ can be recovered as

$$A'_{i_0} = Q_1^{7..0}[l] \oplus \delta^{7..0},\ B'_{i_0} = Q^{23..16}[l] \oplus \delta^{23..16} + 256 \tag{19}$$

So, the second equation in line (18), considering only bit $b$, can be rewritten as

$$s'^b_{i_0} \oplus c_{i_0,b} = P_1^b[A'_{i_0}] \oplus P_1^b[B'_{i_0}] \oplus Q'^b_1[l+12] \tag{20}$$

where $A'_{i_0}$ and $B'_{i_0}$ are known and $c_{i_0,b}$ is an indicator of the carry in $P_1^{b-1..0}[A'_{i_0}] + P_1^{b-1..0}[B'_{i_0}]$ which is also known due to the assumption that bits $b-1, \ldots 0$ of

all the $P$ and $Q$ words are known. Finally, to add equation (20) to the system constructed in the previous sections, the variable $Q_1'^b[l+12]$ needs to be eliminated. Once again, to reexpress $Q_1'^b[l+12]$, the idea is to wait for this value to be *reused* once more during steps $1024, \ldots 1535$.

Due to the assumed lower bound $244 \leq l$, it follows that $l+12 \geq 256$. Hence, it is possible for the $B_i$ index in some step $1024 \leq i \leq 1535$ to take the value $l+12$ which was used in step $i_0$. If such a step exists, denote it by $i_1$. Also, assume that $A_{i_1} < f-1$, so that $Q_1[A_{i_1}] = Q_1'[A_{i_1}]$. Finally, assume that both $A_{i_1}$ and $B_{i_1}$ have been successfully recovered by the procedure given in subsection 5.2. Then, if $j_1 = i_1 \bmod 512$, the non-faulty and faulty keystream words are $s_{i_1} = (Q_1[A_{i_1}] + Q_1[B_{i_1}]) \oplus P_2[j_1]$ and $s'_{i_1} = (Q_1[A_{i_1}] + Q_1'[B_{i_1}]) \oplus P_2[j_1]$ and the difference can be computed as

$$s_{i_1} \oplus s'_{i_1} = (Q_1[A_{i_1}] + Q_1[B_{i_1}]) \oplus (Q_1[A_{i_1}] + Q_1'[B_{i_1}]) \tag{21}$$

Extracting bit $b$ from (21) and cancelling out $Q_1[A_{i_1}]$ yields

$$s_{i_1}^b \oplus s_{i_1}'^b = Q_1^b[B_{i_1}] \oplus c_{i_1,b} \oplus Q_1'^b[B_{i_1}] \oplus c'_{i_1,b} \tag{22}$$

where $c_{i_1,b}$ and $c'_{i_1,b}$ are carry indicators for $Q_1^{b-1..0}[A_{i_1}] + Q_1^{b-1..0}[B_{i_1}]$ and $Q_1^{b-1..0}[A_{i_1}] + Q_1'^{b-1..0}[B_{i_1}]$, respectively. The carry indicator $c_{i_1,b}$ is calculated trivially and as for $c'_{i_1,b}$, it is necessary to find $Q_1'^{b-1..0}[B_{i_1}]$. For that, it suffices to focus on the bits $b-1, \ldots 0$ in equation (21), since all values except $Q_1'^{b-1..0}[B_{i_1}]$ are known and the required value can be calculated as

$$Q_1'^{b-1..0}[B_{i_1}] = ((s_{i_1}^{b-1..0} \oplus s_{i_1}'^{b-1..0}) \oplus (Q_1^{b-1..0}[A_{i_1}] + Q_1^{b-1..0}[B_{i_1}])) - Q_1^{b-1..0}[A_{i_1}] \tag{23}$$

After finding $c_{i_1,b}$ and $c'_{i_1,b}$, from (22) and since $B_{i_1} = l+12$, $Q_1'[l+12]$ can be expressed in terms of $Q_1$ bits as

$$Q_1'^b[l+12] = s_{i_1}^b \oplus s_{i_1}'^b \oplus Q_1^b[B_{i_1}] \oplus c_{i_1,b} \oplus c'_{i_1,b} \tag{24}$$

Substituting (24) in (20) yields

$$s_{i_0}'^b \oplus c_{i_0,b} = P_1^b[A'_{i_0}] \oplus P_1^b[B'_{i_0}] \oplus s_{i_1}^b \oplus s_{i_1}'^b \oplus Q_1^b[B_{i_1}] \oplus c_{i_1,b} \oplus c'_{i_1,b} \tag{25}$$

which is added to the system of equations without introducing any new variables. The described procedure is summarized by Algorithm 6.

Let $N$ denote the number of equations generated by repeating the procedure above for all $f \in S_Q^{-1}(l)$ and $244 \leq l \leq 499$. To estimate $E(N)$, let $\rho(l+12)$ be the step number $i$, $1024 \leq i \leq 1535$, for which $B_i = l+12$, if such a step exists. Also, let $I$ denote the indicator function, returning 1 if the condition in question is true and returning 0 otherwise. Finally, let $FLT_Q[f]$ be the number of faults that occurr at position $Q[f]$. Then

$$N = \sum_{l=244}^{499} \sum_{f \in S_Q^{-1}(l)} FLT_Q[f] \times I[\rho(l+12) \text{ exists}] \times I[A_{\rho(l+12)} < f-1] \times I[A_{\rho(l+12)}, B_{\rho(l+12)} \text{ known}]$$

**Algorithm 6. Add equations by expressing the faulty with non-faulty bits**

**INPUT:** Faulty keystream for a fault occuring at $Q[f]$, $f \in S_P^{-1}(l)$, $244 \leq l \leq 499$
**OUTPUT:** An equation of form (25)

1: Let $\delta = s_{512+l} \oplus s'_{512+l}$ and $i_0 = 512 + l + 12$
2: Calculate $A'_{i_0}$ and $B'_{i_0}$ according to (19)
3: If $P_1^{b-1..0}[A'_{i_0}] + P_1^{b-1..0}[B'_{i_0}] < 2^b$ set $c_{i_0,b} = 0$, else $c_{i_0,b} = 1$
4: For $1024 \leq i_1 \leq 1535$ such that $A_{i_0}$ and $B_{i_0}$ are known
5: If $B_{i_0} = l + 12$ and $A_{i_0} < f - 1$
6: If $Q_1^{b-1..0}[A_{i_1}] + Q_1^{b-1..0}[B_{i_1}] < 2^b$, let $c_{i_1,b} = 0$, else $c_{i_1,b} = 1$
7: Calculate $Q_1'^{b-1..0}[B_{i_1}]$ according to (23)
8: If $Q_1^{b-1..0}[A_{i_1}] + Q_1'^{b-1..0}[B_{i_1}] < 2^b$, let $c'_{i_1,b} = 0$, else $c'_{i_1,b} = 1$
9: Return equation (25)

Recall that $E(FLT_Q[f]) = 7.52$. Also, $E(I[\rho(l + 12) \text{ exists}]) \approx 1 - (\frac{255}{256})^{512}$. If $f > 257$, $E(I[A_{\rho(l+12)} < f - 1]) = 1$ and otherwise $\frac{f-2}{256}$. Finally, according to subsection 5.2, $E(I[A_{\rho(l+12)}, B_{\rho(l+12)} \text{ known}]) \geq \frac{472.7}{512}$. Substituting the values above and using additivity of $E(\cdot)$ yields that $E(N) \geq 18380.1$.

## 6 Attack Complexity and Experimental Results

Adding the number of equations generated by the algorithms presented in subsections 5.3, 5.4 and 5.5 gives a lower bound of $512 + 214.49 + 218.01 + 18380.1 = 19324.6$ equations expected to be in the final system for bits $b \in \{0, \ldots 30\}$. The correctness of the system and the uniqueness of the solution have been verified experimentally as follows. For 100 times HC-128 was randomly initialized and the faults have been simulated as specified by the attack. The procedures specified by by subsections 5.3, 5.4 and 5.5 have been executed and the rank of the resulting system of equations for bits $b \in \{0, \ldots 30\}$ was verified to be 1024 in all the 100 times. As for bit $b = 31$, the procedures from subsection 5.4 are not applicable, leaving out the system to be generated only by subsections 5.3 and 5.5, yielding about $512+18380.1 = 18892.1$ equations. Again, throughout the 100 experiments, the rank of resulting system for bit $b = 31$ was 1022 each time. Thus, to yield a complete HC-128 inner state, the missing two bits need to be guessed. The correctness of the guessed bits is easily verified by running the cipher and comparing the resulting key stream with the observed one. As for the attack complexity, around 7698.4 faults at random inner state words are required, as given by the beginning of section 5. The most expensive computational factor in the attack is solving the linear system of equations in 1024 bit variables for 32 times.

## 7 Conclusion

In this paper, a DFA attack on HC-128 was presented. The adopted attack model assumes that the attacker is able to fault a random word of the inner state of the

cipher but cannot control its exact location nor its new faulted value. The attack operates by constructing 32 systems of linear equations over $Z_2$, each of 1024 bit variables representing the inner state values. It also utilizes what we called the *reuse* of inner state words in different states of the cipher in order to facilitate the differential fault analysis.

## References

1. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
2. Boneh, D., Demillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
3. Dusart, P., Letourneux, G., Vivolo, O.: Differential fault analysis on AES. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 293–306. Springer, Heidelberg (2003)
4. Hoch, J., Shamir, A.: Fault Analysis of Stream Ciphers. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 240–253. Springer, Heidelberg (2004)
5. Kircanski, A., Youssef, M.A.: Differential Fault Analysis of Rabbit. In: Rijmen, V. (ed.) SAC 2009. LNCS, vol. 5867, pp. 197–214. Springer, Heidelberg (2009)
6. Maitra, S., Paul, G., Raizada, S.: Some observations on HC-128. In: Proceedings of the International Workshop on Coding and Cryptography, WCC, Ullensvang, Norway, May 10-15, pp. 527–539 (2009)
7. Mitzenmacher, M., Upfal, E.: Probability and Computing. Cambridge University Press, Cambridge, ISBN-10: 0521835402
8. Staffelbach, O., Meier, W.: Cryptographic Significance of the Carry for Ciphers Based on Integer Addition. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 601–613. Springer, Heidelberg (1991)
9. Wu, H.: The Stream Cipher HC-128. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 39–47. Springer, Heidelberg (2008)
10. Wu, H.: A new stream cipher HC-256. In: Roy, K.B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 226–244. Springer, Heidelberg (2004)
11. Zenner, E.: A Cache Timing Analysis of HC-256. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 199–213. Springer, Heidelberg (2009)

# Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices

Marcel Medwed[1], François-Xavier Standaert[2], Johann Großschädl[3], and Francesco Regazzoni[2]

[1] Graz University of Technology, Austria
[2] Université catholique de Louvain, Belgium
[3] University of Luxembourg, Luxembourg

**Abstract.** The market for RFID technology has grown rapidly over the past few years. Going along with the proliferation of RFID technology is an increasing demand for secure and privacy-preserving applications. In this context, RFID tags need to be protected against physical attacks such as Differential Power Analysis (DPA) and fault attacks. The main obstacles towards secure RFID are the extreme constraints of passive tags in terms of power consumption and silicon area, which makes the integration of countermeasures against physical attacks even more difficult than for other types of embedded systems. In this paper we propose a fresh re-keying scheme that is especially suited for challenge-response protocols such as used to authenticate tags. We evaluate the resistance of our scheme against fault and side-channel analysis, and introduce a simple architecture for VLSI implementation. In addition, we estimate the cost of our scheme in terms of area and execution time for various security/performance trade-offs. Our experimental results show that the proposed re-keying scheme provides better security (and does so at less cost) than state-of-the-art countermeasures.

## 1 Introduction

Radio-Frequency Identification (RFID) is an emerging technology that enables identification of non-line-of-sight objects or subjects. Based on cheap RF microcircuits—called tags—apposed on or incorporated in the items to identify, the RFID technology is now widely deployed in our everyday life. Considering the plethora of applications that are targeted, it is little surprising that security and privacy issues related to RFID technology have become an important concern in recent years. Solving these issues has created a need for cryptography-enabled RFID tags that fulfill stringent constraints on area and power consumption.

Unfortunately, while designing secret-key or public-key cryptographic hardware on an area budget of a few thousand logic gates is challenging enough, the cost criteria is not the only one to be met by secure RFID tags. In particular, as these tags often operate in insecure or even hostile environments, they can be subject to different types of implementation (i.e. physical) attacks. This means that, instead of targeting the cryptographic algorithms and/or protocols at the mathematical level, an adversary may directly attack their implementation in

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 279–296, 2010.

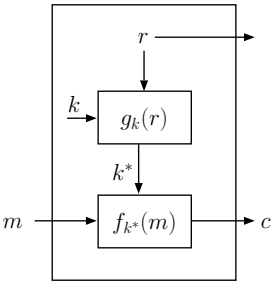



**Fig. 1.** Fresh re-keying: basic principle

order to break a system. A considerable number of experiments reported in the literature during the past ten years demonstrate the low-cost nature of certain classes of physical attacks, most prominently side-channel attacks (e.g. Simple Power Analysis, Differential Power Analysis [14]) and fault attacks [1]. The very same attacks that can break unprotected smart cards also apply to RFID tags (just slight adaptations of the measurement setup are necessary [10]), which is no surprise given that both types of devices are manufactured using the same or similar VLSI technology. As a consequence, the need for low-cost protections against such physical attacks emerges. The requirement of physical security is a particularly challenging issue since most published approaches to increase the resistance against side-channel or fault attacks are rather expensive [19]. Even worse, there exist only very few combined countermeasures against both types of attacks. On the other hand, developing and implementing security protocols with large mathematical security margins is not very worthwhile if the devices executing these protocols do not provide a similar (or at least reasonable) level of physical security.

In order to solve these problems, we introduce a simple re-keying scheme to protect RFID tags and other low-cost devices against Differential Fault Attacks and a large category of side-channel attacks (namely the "standard" SPA and DPA attacks as described in Section 2). This scheme, depicted in Figure 1, can be used in a challenge-response authentication protocol for RFID tags or, more generally, for physically secure encryption. It contains an encryption function $f$ (typically, $f$ is a block cipher; we will use the AES as example throughout this paper) to encrypt every challenge (or, more generally, every message block $m$) with a fresh session key $k^*$. This session key $k^*$ is obtained with the help of a function $g$ that uses a master key $k$ and an on-tag generated public nonce $r$ as input. That is, the tag computes the session key $k^* = g_k(r)$ first and then the ciphertext $c = f_{k^*}(m)$. At a first glance, it may seem that such a scheme just shifts the problem of protecting the block cipher $f$ against physical attacks to the problem of protecting the function $g$ against the same attacks. However, we argue in this paper that fresh re-keying provides significant advantages, both in terms of security and performance, over existing countermeasures. Firstly, and quite simply, it makes attacks based on Differential Fault Analysis unpractical because the same key is never used twice to encrypt a challenge or message. Sec-

ondly, it allows for separating the requirements on the two functions. On the one hand, $g$ has to be low-cost and easy to protect against side-channel attacks, but does not have to be cryptographically strong. On the other hand, $f$ only needs to be secure against side-channel attacks with a data complexity bounded to one single query (i.e. SPA, essentially).

In the following sections of this paper, we define a number of desired properties for the function $g$ and propose a concrete implementation that we analyze in detail. We then explore the design space for $g$ and discuss different trade-offs between performance and resistance against side-channel attacks. Relying on previous results of evaluations of protected devices, our implementation figures show that a significant level of physical security can be obtained at reasonable cost. In particular, we assess the computational difficulty of performing attacks based on the traditional "divide-and-conquer" strategy in which different parts of the master key $k$ are recovered separately. We show that the cost of such an attack is prohibitive for the targeted applications. Regarding fault analysis, we discuss the protection against differential fault attacks, but do not consider any of the simple fault attacks that reduce the number of rounds of a block cipher or output the key instead of the ciphertext. These attack scenarios represent a more general, scheme-independent threat and are commonly prevented by other means, e.g. loop invariants or code signatures. Finally, we briefly discuss the resistance of our scheme against the recently introduced algebraic side-channel attacks [29]. While the exact evaluation of such advanced techniques is left as a topic for further research, we sketch solutions to prevent them.

### 1.1 Related Work

There exists a significant literature on countermeasures to prevent side-channel attacks. In this section, we summarize the most common countermeasures and discuss their advantages and limitations. We then argue how our proposal can be seen as a new trade-off between security and performance.

First, masking (e.g. [7,31]) is a commonly applied and thus well understood technique to protect a device against side-channel attacks. Its main drawback is that the performance penalty can be significant because of the need to compute a correction term on-the-fly, during the encryption process [7]. Masking can be defeated by higher-order attacks [21] or due to hardware issues such as glitches [18]. Nonetheless, it is usually considered as an effective ingredient for the protection of cryptographic hardware. The permutation tables analyzed in [3] have quite similar properties, both in terms of performance and security [27].

Next to masking, hiding is another frequently applied countermeasure. Many hiding schemes have been proposed in the literature recently, e.g. different time randomization tools or side-channel resistant logic-styles featuring an (almost) data-independent power consumption profile. Such logic-styles can be based on standard CMOS cell libraries (e.g. WDDL [36]), or require full-custom design (e.g. SABL [35]). Again, there is a security vs. performance/cost trade-off since designing full-custom hardware is more expensive than using standard cells, but provides also more options for fine-tuning and, hence, better security [16].

Closer to our proposed solution are different protocol-level countermeasures such as regular key updates. The idea of regular key updates was first described in [13] and has recently attracted significant attention, see e.g. [22,23]. These re-keying schemes have the advantage of being formally analyzed, which allows for a good evaluation of the security level they provide. On the other hand, they still rely on certain physical assumptions that must be fulfilled by the hardware and they can be quite inefficient when a chip has to be re-initialized regularly (which is typically the case in a challenge-response setting). More precisely, as detailed in [33], a secure initialization process for such constructions using e.g. the AES would require a tree-based structure with up to $n$ applications of the AES, where $n$ is the size of the initialization vector. This hardly fits into the RFID realm.

The use of all-or-nothing transforms to prevent certain types of side-channel attacks is discussed in [15]. Here the idea is to transform the plain- and ciphertexts with a low-cost mapping that is easy to protect against physical adversaries and makes the guessing strategy, exploited in most standard DPA attacks, hard to apply. As this proposal is quite recent, its careful security analysis is still an open problem. Interestingly, it also shifts the problem of protecting a complete cipher to the problem of protecting a simpler transform. But as for re-keying schemes, the initialization and synchronization of an encryption protected with such all-or-nothing transforms can be expensive. Another drawback is the need of an additional secret shared between the two parties.

Our fresh re-keying is related to the idea of all-or-nothing transforms and the standard re-keying schemes. Its big advantage is to provide a low-cost solution to the initialization problem because a fresh session key is used to encrypt every block of plaintext. In addition, since we also apply a transform on the key, we avoid the need of sharing an additional secret as is the case with all-or-nothing transforms. Finally, the proposed solution is low-cost because we only need one transform to protect the key rather than two transforms to protect the plaintext and ciphertext. On the other hand, we share the advantages of these protocol-level countermeasures. In particular, we do not need to compute correction terms during the encryption process and can take advantage of masking and hiding to protect our re-keying scheme, as will be detailed in the following sections. Due to its high regularity, we can even consider a full-custom implementation.

Note that, because we focus on RFID applications, this paper primarily deals with implementation efficiency, as will be detailed in Section 4 and 5. In particular, we demonstrate that for the same or similar gate count, our scheme allows for implementing more masking and shuffling than if one directly attempts to protect a block cipher design with similar countermeasures. We believe that this is an important first step in order to motivate further research on fresh re-keying schemes. Our security analysis considers an important class of practical attacks but generalizing it towards more abstract (or general) models of computation and leakage (e.g. the ones summarized in [24]) and evaluating the performance penalties that this would imply is an interesting open research question.

## 2 Background

As detailed in the introduction, the countermeasure proposed in this paper splits the problem of physical security into different subproblems. Some parts of our design are only required to be protected against SPA, whereas other parts also require DPA-resistance. Since these are all standard notions in the field of cryptographic hardware, we only summarize them and point to references for a more formal treatment. In addition, we describe DPA attacks exploiting the standard divide-and-conquer strategy that we consider in our security analysis. Finally, we discuss how our re-keying scheme can be used in an authentication protocol.

### 2.1 SPA and DPA

In terms of side-channel resistance, the main requirement for our protocol to be secure can be summarized as follows:

1. The function $f$ needs to be secure against SPA.
2. The function $g$ needs to be secure against both SPA and DPA.

SPA stands for Simple Power Analysis and corresponds to an attack in which an adversary directly recovers key material from the inspection of a single measurement trace (i.e. power consumption or EM radiation, typically). DPA stands for Differential Power Analysis and corresponds to more sophisticated attacks in which the leakage corresponding to different measurement traces (i.e. different plaintexts encrypted under the same key) is combined. As a matter of fact, in absence of an efficient solution to guess the session key $k^*$ from the master key $k$, such attacks can only be applied to the function $g$ in the scheme depicted in Figure 1. Indeed, for the block cipher $f$, every plaintext will be encrypted using a different $k^*$. For more details about such attacks, we refer to [19].

### 2.2 Divide-and-Conquer Strategies

Divide-and-conquer attacks, such as the standard DPA described in [20], are attacks in which the adversary recovers small parts of a master key (also called subkeys) one by one. Most side-channel attacks published in the open literature fall into this category. In such a setting, an important skill of the adversary is the ability to predict some (key-dependent) intermediate computations during the encryption process (e.g. the first round S-box outputs in a block cipher). As will be detailed in Section 6.3, this is typically what is made difficult by our fresh re-keying scheme. If $g$ has good enough diffusion, it should be hard to guess the intermediate computations of $f$ depending on the master key $k$. Therefore, only SPA attacks can be performed against the session key $k^*$.

### 2.3 Challenge-Response Protocol

In a challenge-response authentication protocol, one entity sends a challenge and the other party responds with the encrypted challenge together with some additional information. Then, the response is checked. Depending on the use

case, this process can be repeated with swapped roles. As our re-keying scheme is designed for physically secure encryption, it can be straightforwardly used in any symmetric-key challenge-response authentication. The tag simply has to implement the fresh-rekeying as shown in Figure 1. The reader implements the same scheme, except that the nonce $r$ is provided from outside by the tag.

Regarding the communication overhead, the transmission of the $r$ values does not necessarily imply that the number of passes in the protocol increases. In a three-pass mutual authentication protocol (such as described in, e.g., ISO/IEC 9798-2 [11]) the $r$ values can be included in data transported during the passes. Thus, the number of passes increases at most by one, depending on who starts the protocol. Note that an important property of our fresh re-keying is that the adversary should not gain an advantage when resetting the device. That is, after each reset the tag should compute a fresh nonce $r$ and session key (in a passive RFID scenario, the tag is reset any time it is taken out of the reader field).

## 3 Choice of the Function $g$

In order to investigate the security of the fresh re-keying scheme in Figure 1, one first needs to determine the two functions $f$ and $g$. As previously mentioned, a natural choice for the function $f$ is a block cipher, e.g. AES. Hence, it remains the choice of the function $g$, which is, in fact, the most critical both for security and performance. In this section, we specify the required properties for $g$ and select an appropriate candidate according to those properties.

### 3.1 Desired Properties

The following properties for $g$ are motivated by a combination of side-channel security aspects and hardware implementation aspects.

*P1: Diffusion.* One bit of the session key $k^*$ should depend on many bits of the master key $k$. In other words, guessing one bit of the session key must be computationally difficult. This property ensures that the divide-and-conquer technique, usually applied in DPA, cannot be easily carried out.

*P2: No need for synchronization.* The function $g$ should not have a variable inner state which needs to be kept synchronous among the parties. The only inner state should be the static portion $k$ (contrary to [22,23]).

*P3: No additional key material.* The symmetric key material, which needs to be distributed in advance among the parties and stored within the devices, should not be larger than that of classical block encryption. That is, the master key $k$ should suffice to evaluate both functions $f$ and $g$ (contrary to [15]).

*P4: Little hardware overhead.* Deriving the session key $k^*$ in hardware must be cheaper than protecting the "original" circuit (i.e. the function $f$) by means of secure logic-styles and other countermeasures.

*P5: Easy to protect against SCA.* $g$ should have a suitable algebraic structure that makes its protection against SCA easier than e.g. block ciphers. Combined

with the previous property, this means that deriving the session key $k^*$ with a protected $g$ should also be lower in cost than protecting $f$.

*P6: Regularity.* If possible, the function $g$ should have a high regularity in order to facilitate its implementation in a full-custom design. This is motivated by the good security properties that the fine-tuning of such designs allows.

### 3.2 Candidate

From a cryptographic point of view, the most obvious choice for $g$ would either be a hash function or an encryption function. However, they would not be useful in the present context since they are just as complex to implement and protect as the original block cipher $f$. In contrast, from an engineering point of view, a bitwise XOR function would be best. In fact, an XOR fulfills many of the above properties, but the diffusion remains very weak. Combining these two extremes led us to select $g$ as the following modular multiplication:

$$g : \left(\mathrm{GF}(2^8)[y]/p(y)\right)^2 \rightarrow \mathrm{GF}(2^8)[y]/p(y) : (k, r) \rightarrow k * r.$$

In the later sections of this paper, the polynomial $p(y)$ will be defined as $y^d + 1$ with $d \in \{4, 8, 16\}$. The actual choice of $d$ will be used as parameter to improve the diffusion (i.e. *P1*), as will be discussed in Section 6.3. Regarding the other properties, *P2* is fulfilled because the function only depends on the public but random nonce $r$ and the secret key $k$; *P3* is fulfilled because only one master key $k$ is needed to evaluate $g$. *P4*-*P6* are discussed in the next section.

Note that the diffusion property of this modular multiplication significantly depends on the choice of $r$. Since $r$ is randomly generated on-chip by the tag, it allows arguing about the tag's physical security by showing that the diffusion is high enough on average. By contrast, on the reader side, the nonce might be generated by an adversary. Hence, the re-keying will not ensure diffusion (and physical security) on that side. As a consequence, the (more expensive) reader is expected to be protected against implementation attacks by other means.

## 4 Implementation of the Function $g$

In this section we elaborate on the implementation of $g$. We start from a general description of the multiplication algorithm, extend it to a blinded version, and eventually discuss the use of secure logic for a hardware implementation.

### 4.1 Unprotected Implementation

The unprotected implementation of the multiplication follows Algorithm 1, the complexity of which mainly depends on $p(y)$. Thus, the degree of this polynomial can be used to trade off performance for diffusion. For example, if $d = 16$ (resp. $d = 8$), every bit of the session key $k^*$ will depend on 64 (resp. 32) bits of the master key on average (refer to Section 6.3 for details). Note that if $d < 16$, the

**Algorithm 1.** Product-scan algorithm for multiplication.

**Require:** $a, b \in \mathrm{GF}(2^8)[y]/y^d + 1$
**Ensure:** $c = a * b \in \mathrm{GF}(2^8)[y]/y^d + 1$
1: $\rho \leftarrow \mathrm{rand}(),\ i \leftarrow 0,\ j \leftarrow \rho,\ k \leftarrow \rho,\ l \leftarrow 0$
2: **while** $k \neq \rho - 1 \bmod d$ **do**
3: ACCU $\leftarrow 0$
4: **for** $l = 0$ to $d - 1$ **do**
5: ACCU $\leftarrow$ ACCU $+ a_i \cdot b_j$
6: **if** $l < d$ **then**
7: $i \leftarrow i - 1 \bmod d$
8: **end if**
9: $j \leftarrow j + 1 \bmod d$
10: **end for**
11: $c_k \leftarrow$ ACCU
12: $k \leftarrow k + 1 \bmod d$
13: **end while**
14: **return** $c$

multiplication is simpler, but needs to be applied several times to cover all key bytes (e.g. twice if $d = 8$, four times if $d = 4$).

We opted for a product-scan algorithm [8], in which the result is calculated digit-wise. That is, in each iteration of the outer loop (lines 4–10), all partial products which add to the same digit of the final product are computed and accumulated. The main disadvantage of this algorithm is the out-of-order processing of the operands. However, the special choice of $p(y)$ allows to overcome this problem; this choice will be justified in Section 4.4.

### 4.2 Improving $g$'s SPA/DPA Resistance with Shuffling

Due to the structure of the ring we are operating on, the individual digits of the product are independent (i.e. "carry-free"), which allows one to randomize the order in which these digits are accumulated. Therefore, *shuffling* can be applied as a side-channel countermeasure [19]. Shuffling has the effect that an adversary who observes a side-channel trace can not directly infer the operations carried out in different samples (i.e. in our case: which part of the product is processed at what time), which makes SPA difficult. In addition, shuffling also increases the data complexity of a DPA attack by $d^2$ [9]. Note that this countermeasure comes for free in our case because only the starting index of the outer loop has to be initialized with a random value (line 1 in Algorithm 1).

### 4.3 Improving $g$'s SPA/DPA Resistance with Blinding

DPA attacks against a multiplication algorithm usually target the partial products. This is simply because a partial product depends only on one digit of each operand, which allows for applying a divide-and-conquer strategy. A common

**Algorithm 2.** Blinded session key generation.

**Require:** $k, r, b_i$ with $i = 1$ to $m$, the masking order
**Ensure:** $k^* = k * r$
1: $bk \leftarrow k$
2: **for** $i = 1$ to $m$ **do**
3: $\quad bk \leftarrow bk + b_i$
4: **end for**
5: $k^* \leftarrow bk * r$
6: **for** $i = 1$ to $m$ **do**
7: $\quad k^* \leftarrow k^* + b_i * r$
8: **end for**
9: **return** $k^*$

side-channel countermeasure, which is also applicable in our context, is to use a redundant representation for the variables. Sharing a variable over $(m+1)$ variables is referred to as $m^{\text{th}}$-order *blinding* (also called masking in the symmetric setting [31]). Blinding is a powerful countermeasure, but only efficient when the computational overhead due to operating on such a redundant representation is small. Since addition and multiplication are distributive in our algebra, this condition is nicely satisfied.

Algorithm 2 shows an $m^{\text{th}}$-order blinded version of the function $g$. In line 3, $m$ random blinds $b_i$ are added to $k$ before the multiplication is carried out in line 5; afterwards, each product $b_i * r$ has to be removed again from the result in line 7. It can be easily verified that this does not change the result. However, it ensures that any adversary who wants to mount a DPA on $g$ needs to exploit the joint information of $m$ partial products, thus perform an $m^{\text{th}}$-order DPA. The number $m$ presents the second parameter in our design space for $g$. Both the time and space complexity of $g$ increases linearly with $m$, while the complexity of a DPA on $g$ grows exponentially with $m$ [2].

### 4.4 Improving $g$'s SPA/DPA Resistance with Protected Logic-Styles

Finally, the main reason why we opted for the product-scan algorithm is because it facilitates the use of secure logic-styles. More precisely, the part of the memory that holds sensitive data is small when multiplying operands according to the product-scan method. While a DPA can be performed on the partial products during the execution of the multiplication, it is difficult to attack a byte of the final result directly, as they depend on many bytes of both operands. Since the product-scan algorithm operates only on one product-byte at a time, only this byte and the corresponding multiplication in $\text{GF}(2^8)$ have to be protected. In general, secure logic is expensive compared to standard CMOS; thus it should be used sparingly. However, this is exactly what our proposed re-keying scheme allows a designer to achieve. A third parameter in our design space will be the use of such protected hardware and the resulting impact on area.

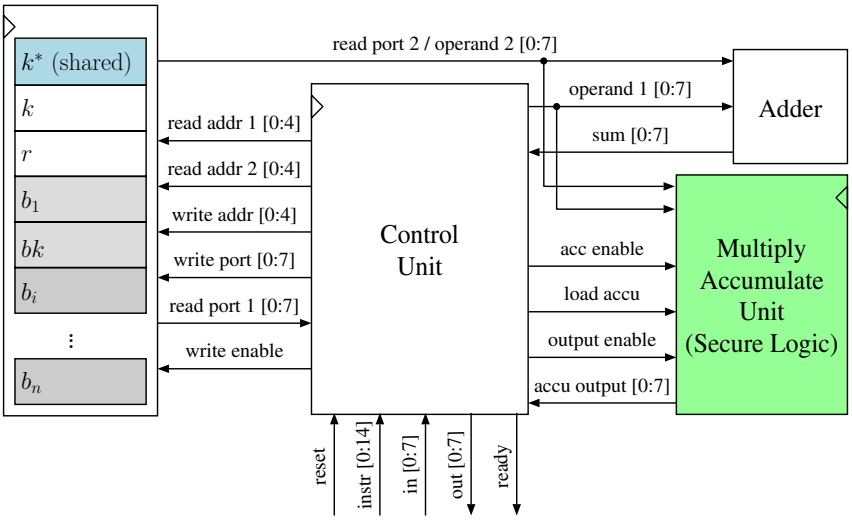



**Fig. 2.** Block diagram of the random transformation circuit

## 5 Global Architecture

Following the algorithmic description in the previous section, we now focus on concrete hardware cost and performance issues. We investigate the design space of $g$ and present the results of different hardware implementations.

### 5.1 Block Diagram and Design Space for the Function $g$

Figure 2 depicts a hardware architecture for $g$. The diagram contains all components necessary to generate $k^*$, except a random number generator, which we can reasonably assume to be available on the tag. The three main components of the architecture are the controller, the memory and the multiply-accumulate (MAC) unit. Determined by the use of the AES for the function $f$, the memory consists of 128-bit registers. Note that the register holding $k^*$ would be shared with $f$, hence it does not contribute to the cost of $g$'s circuit. The actual size of the memory is directly related to the second design parameter: the blinding order. If the blinding order is zero, only two registers are needed. If $m^{\text{th}}$-order blinding is implemented, $(m+1)$ additional registers are required.

The control unit is basically invariant across the design space. Changing the degree of the polynomial only changes loop constants within the controller and affects the clock cycles needed to carry out an operation. Also the successive executions of an operation (such as needed for $d \in \{4, 8\}$) are managed by the controller. A similar statement holds when changing the blinding order. The core of the architecture is the MAC unit featuring a GF($2^8$) multiplier, a GF($2^8$) adder, and an 8-bit register. Since the MAC unit targeted for implementation in secure logic, its size is crucial, as will be analyzed in the next section.

**Table 1.** Post synthesis results of an ASIC implementation of $g$, $g$-pMAC and the full re-keying scheme. The protected MAC unit is estimated on basis of iMDPL logic.

| Implementation | w/o blinding | 1st-order | 2nd-order | 3rd-order |
|---|---|---|---|---|
| function $g$ | 4.5 kG | 7.3 kG | 8.7 kG | 10.2 kG |
| $g$-pMAC | 11.7 kG | 14.6 kG | 16.0 kG | 17.5 kG |
| $g$ + AES[1] | 7.9 kG | 10.7 kG | 12.1 kG | 13.6 kG |
| $g$-pMAC + AES[1] | 15.1 kG | 18.0 kG | 19.4 kG | 20.9 kG |

[1]AES implementation taken from [5].

### 5.2 Implementation Results and Performance Evaluation

We evaluated the post-synthesis silicon area of an ASIC implementation of the function $g$. The synthesis tool we used was the Design Compiler 2008.09 from Synopsys, and the library was the Free Standard Cell library from Faraday Technology for the UMC L180 CMOS technology. Our evaluation is based on typical corner values and reasonable assumptions for the constraints. Additionally, we varied the frequency between 1 and 20 MHz, which are reasonably boundaries for the considered application (typical frequencies are 6 MHz for HF tags and 1 MHz for UHF tags). After several synthesis runs with different constraints and optimization options, we selected the results with the smallest area. Our results are summarized in Table 1, including the following information:

1. The area estimation (gate equivalents) for $g$ for different blinding orders.
2. The cost of a MAC unit in a DPA-resistant logic-style ($g$-pMAC). We used iMDPL [26] and, hence, a scaling factor of 18 [12] for our estimations.
3. The total area ($g$ + AES) needed for protecting the AES core of Feldhofer et al. [5] using our fresh re-keying scheme, with the same parameters.

We compared our design to the protected AES circuit presented by Feldhofer et al. in [6]. Their implementation has an area of approximately 19.5 kG and is, to the best of our knowledge, the smallest protected AES core that targets RFID applications. These results show that our fresh re-keying scheme requires less area than a direct protection of its underlying block cipher, i.e. properties *P4*, *P5* and *P6* in Section 3.1 are indeed fulfilled. More precisely, the implementation in [6] has a level of security comparable to the one of $g$ featuring either 1st-order blinding or a protected MAC. This is because their implementation protects parts with masking and other parts with secure logic. In the first case, our implementation requires approximately 8.8 kG less than theirs, and in the second case, the difference is approximately 4.4 kG. A combination of the two countermeasures would still be around 1.5 kG smaller. Note that we could also implement a blinding order of up to 5 at the same cost.

Table 2 presents the number of clock cycles needed for the generation of a fresh session key $k^*$. Contrary to the area requirements, this number is strongly determined by the polynomial selected for the diffusion. For example, when this polynomial equals $y^{16} + 1$, the time required to complete the computation is

**Table 2.** Cycle count for re-keying with different diffusion levels and blinding orders

| Blinding order | $y^{16}+1$ | $y^{8}+1$ | $y^{4}+1$ |
|---|---|---|---|
| w/o blinding | 290 | 162 | 98 |
| $1^{st}$-order | 562 | 360 | 178 |
| $2^{nd}$-order | 834 | 504 | 258 |
| $3^{rd}$-order | 1160 | 648 | 338 |

almost three times longer than when using $y^4 + 1$. The corresponding security levels will be discussed in the next section. As a consequence, the designer can easily tune the desired diffusion level towards the needs of the application (in RFID the throughput is generally not a strict constraint). For comparison, the performance overhead of the implementation of Feldhofer et al. [6] is between 32 and 1005 clock cycles, depending on the number of dummy instructions.

## 6 Security Analysis

In this section, we provide a security analysis of our re-keying scheme. We start with a note on the choice of $k$. Next, we discuss the security of the complete scheme against Differential Fault Analysis. Then, we investigate its resistance against side-channel attacks in three parts. First, we argue about the security of the function $g$ against SPA and DPA. Second, we discuss the security of the function $f$ against SPA only. Finally, and most importantly, we aim to analyze the difficulty of mounting divide-and-conquer attacks against the complete re-keying scheme. We will conclude the section with some open questions related to advanced attack techniques exploiting algebraic cryptanalysis.

### 6.1 The Choice of $k$

Due to the structure of the ring we use, there exist zero divisors. If $k$ takes the value of a zero divisor, there exist several $r$ which lead to the same $k^*$. To avoid such collisions, we have to reduce the key space $K$ to elements $k \in K$ that are co-prime to the polynomial $y^d + 1$. The resulting loss of entropy can be stated as $\Delta H = 128 - H(K)$. For $d = 16$, we get $\Delta H = 128 - \log_2(255 * 256^{15}) \approx 0.0056$ bits. For $d = 8$, $\Delta H$ doubles, while for $d = 4$ it becomes four times as large. In any of the three cases, the reduction of $K$ can be neglected.

### 6.2 Resistance against Fault Attacks

Even in its most powerful variants, Differential Fault Analysis requires at least one pair of correct and erroneous outputs to attack cryptographic algorithms [25]. From such a pair, information about the secret key can be recovered. This means that an adversary requires to encrypt the same plaintext (at least) twice with the same secret key, which is prevented by our scheme. In other words, the combination of re-keying with an initialization process using a fresh $r$ for every plaintext block provides a solid protection against Differential Fault Attacks.

### 6.3 Resistance against Standard Side-Channel Attacks

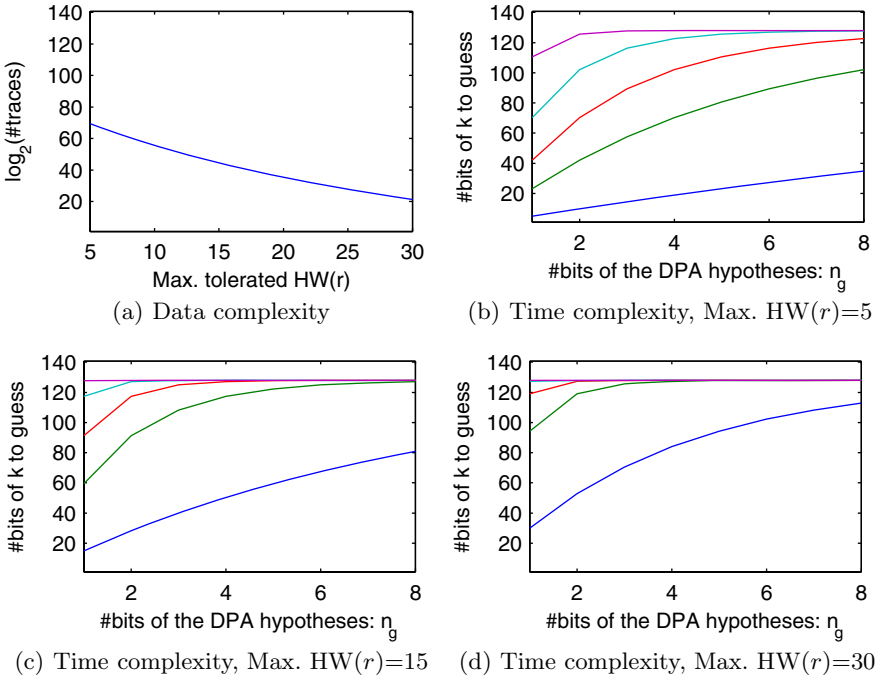
As mentioned in Section 2.1, the security of our fresh re-keying scheme is based on two requirements: (1) the function $f$ needs to be secure against SPA; (2) the function $g$ needs to be secure against both SPA and DPA. In this section, we elaborate on how our architectural choices allow for fulfilling these requirements (depending on different security parameters). Thereafter, we analyze in detail the security of the complete scheme against divide-and-conquer attacks, i.e. we show that, if the two previous conditions are met, it is computationally hard to mount such DPA attacks against the functions $f$ and $g$ taken as a whole.

**Security of $g$ against SPA and DPA.** The proposed architecture for $g$ allows us to deploy three well-studied countermeasures against DPA attacks, namely shuffling, blinding and protection via secure logic. For an extensive discussion of these countermeasures we refer to e.g. [19]; a more theoretical treatment can be found in [30]. We note that, in addition to the large design space, our scheme has some specific advantages compared to the straightforward protection of a block cipher. For example, designing a masking scheme for a software implementation of a block cipher that has an order higher than 3 is an open problem, as pointed out in [30]. In our case, thanks to the algebraic structure of the function $g$, a generalization to higher orders is as easy as for asymmetric encryption. In addition, as detailed in the previous section, the low-cost nature of $g$ allows one to combine several types of countermeasures against side-channel attacks at a lower cost than if they would be directly applied to the original AES.

**Security of the AES against SPA.** Although not as difficult to achieve as DPA resistance, security against SPA is crucial for the AES. In particular, and since our re-keying scheme implies to run the key scheduling algorithm for any new encryption, it is important to avoid attacks like the one in [17]. In order to limit cost overheads, our strategy is to apply the same shuffling that is described in Section 4.2 to the 16 state bytes of our AES implementation, as well as to the key expansion. As detailed in [6], this can be done with little overhead (we do not even need additional memory as we do not make use of dummy cycles).

**Security of the Overall Scheme against Divide-and-Conquer Attacks.** The previous paragraphs described solutions for achieving SPA/DPA resistance for $g$ and SPA resistance for $f$. They show that the level of security against these attacks can be easily tuned at the cost of some performance overheads that are at least lower than those of protecting a stand-alone AES. It now remains to argue about the security of the combined functions, i.e. can an attacker directly perform a DPA on the function $f$ by guessing the master key $k$?

In order to evidence that such attacks are computationally hard, we argue in the following way. According to [20], a DPA attack against the AES requires guessing some intermediate computation during the encryption process. In the simplest case, one bit may be guessed (e.g. one bit after the first key addition layer). In this context, the number of master key bits on which each bit of the




(a) Data complexity

(b) Time complexity, Max. HW($r$)=5

(c) Time complexity, Max. HW($r$)=15

(d) Time complexity, Max. HW($r$)=30

**Fig. 3.** Data vs. time complexity of a standard DPA against the combined $f$ and $g$

session key $k^*$ depends only depends on the Hamming Weight (HW) of $r$. This is because every bit of $k^*$ is a sum of all bits of $k$ weighted by the bits of $r$. Since all $n$ bits of $r$ are uniformly distributed, the probability that $\mathrm{HW}(r) \leq X$ is

$$P = \Pr[\mathrm{HW}(r) \leq X] = \sum_{i=0}^{X} \frac{\binom{n}{i}}{2^n}.$$

This probability can be directly related to the data complexity of a concrete attack. That is, a small multiple of $1/P$ traces have to be collected to observe an $r$ with the desired properties. Figure 3(a) illustrates this relationship between the Hamming weight of $r$ and the number of traces that have to be collected to observe such an $r$. It can be seen that even for a Hamming weight as large as 30 the data complexity is significant as one million traces would be required.

In a typical DPA, there are two effects that improve the diffusion. First, the adversary will usually not predict the output of the key addition, but rather the output of the first S-box layer (or even MixColumns), in order to better distinguish between the different key candidates. This requires to guess 8 bits of the session key (or 32 for MixColumns); we denote the number of session key bits to guess as $n_g$. Second, several traces corresponding to several plaintexts will generally be combined in a DPA, each one giving rise to a new random $r$. We denote this number of traces as $n_t$. Overall, the percentage of bits on which a

DPA attack depends can be described as a function of the maximally tolerated Hamming weight $X$ as follows:

$$1 - \left(\frac{n - X}{n}\right)^{n_t \cdot n_g}.$$

Figures 3(b)-3(d) show the number of bits of $k$ to guess as a function of the hypothesis size $n_g$. The different curves show the complexity for $n_t = 1$ (lowest curve), 5, 10, 20, and 50 (topmost curve). Since between 10 and 50 traces are usually required to recover an AES key byte with reasonable confidence in unprotected devices [32], it directly implies that the diffusion and, hence, time complexity will generally be sufficient to protect RFID tags.

We end this subsection with two basic examples to illustrate how our countermeasure influences the data and time complexities of a divide-and-conquer attack. First we consider an AES implementation for which the attacker needs to predict $n_g = 8$ bits of the session key (a usual quantity). Furthermore, we assume that he needs $n_t = 10$ traces to mount a successful DPA. Even if the attacker waits for $r$ values with a Hamming weight of 5 (as in Figure 3(b)), he needs to guess almost 128 bits of the master key to predict 10 times those 8 bits of the session key. Thus the time complexity of such an attack would be close to $2^{128}$. To assess the data complexity, we additionally look at the probability of observing such $r$ values. In Figure 3(a), it can be seen that they occur every $2^{70}$ traces on average. For the second example we assume that guessing $n_g = 1$ bit for $n_t = 5$ traces is enough. Even in this (unlikely) case, the data and time complexities are still prohibitive. In order to arrive at a more reasonable data complexity, we wait for $r$ values with Hamming weight 15. This means that we have to observe (and acquire the traces for) $5 \cdot 2^{44}$ encryptions on average. The time complexity in this case would be $2^{60}$. Note that large data complexities may be hard to cope with in practical side-channel attacks as even an effective measurement setup is limited to approximately 20 traces per second.

### 6.4 Resistance against Algebraic Side-Channel Attacks

By clearly separating the properties of the functions $f$ and $g$, our proposed re-keying scheme has pushed the security against side-channel attacks towards an extreme direction. On the one hand, standard side-channel attacks are thwarted (as discussed in this paper). On the other hand, the function $g$ that is protected against such attacks is not very strong from a cryptographic point of view. As a consequence, it appears to be an interesting target for the recently introduced algebraic side-channel attacks [28,29]. These attacks are not based on a divide-and-conquer strategy; they rather interpret the encryption of a plaintext into a ciphertext as a big system of low-degree boolean equations in which the key bits are unknown variables. Then, the information leakage corresponding to this encryption is added to the system, also in the form of low-degree equations. As demonstrated in [29], leaking information about the Hamming weights may be sufficient to solve the system in practical time limits (minutes, typically). Quite

naturally, the complexity of solving such a system of equations strongly depends on the algebraic structure of the target algorithm. For example, the AES is more robust than the low-cost cipher PRESENT in this context [28].

Looking at our proposal for $g$, the situation is even worse since this function is linear. Taking an analogy with stream ciphers, one could see the side-channel leakage as a filtering function at the output of a linear number generator $g$. However, thanks to our flexible architecture, we can also offer positive arguments to prevent such attacks. Most importantly, algebraic attacks can hardly deal with erroneous information. Hence, the shuffling that we can perform for free on the implementation of both $f$ and $g$ will most likely make mounting these attacks much harder. Because of their recent nature, we leave the exact quantification of algebraic side-channel attacks as a scope for further research.

## 7 Conclusions

In this paper, we discussed a new approach for re-keying and explored its use as a countermeasure against physical attacks. The proposed scheme is tailored to the security requirements and resource constraints of RFID applications. We evaluated the architecture and security of the scheme, including its robustness against DFA, SPA, and DPA. The flexibility and configurability of the proposed architecture allows for reaching a high level of security at an area cost that is close to the most efficient solutions available in the literature.

Open problems are in two main directions. First, it would be useful to extend the present proposal in order to protect the reader side (which is needed to be protected against physical attacks by other means than the fresh re-keying in the present proposal). Second, our analysis relies on a simple candidate for the function $g$. Investigating alternative ones, possibly trading some performance overhead for security, is a promising topic for future research. In this context, it is worth noting that there exist simple ways to improve the diffusion properties of our scheme. As an illustration, one can generate two random nonces $r_1$ and $r_2$, and then compress the resulting $k * r_1$ and $k * r_2$ (e.g. by XOR-ing the two halves together, producing $n/2$ bits twice), and use the concatenation of the two compressed strings as $k^*$. Pushing such a diffusion/performance trade-off even further, one could also consider randomness extractors as function $g$ (which are of independent interest in leakage-resilient cryptography [4,34]).

Summarizing, we hope that our new countermeasure and instantiation for $g$ makes an interesting case compared to traditional approaches to prevent physical attacks and raises interesting (theoretical and practical) research questions.

**Acknowledgements.** This work was funded in part by the European Commission's ECRYPT II Network of Excellence, by the Belgian State IAP program P6/26 BCRYPT, by the Walloon region E.USER project, and by the Austrian Government funded project ARTEUS. François-Xavier Standaert is a Research Associate of the Belgian Fund for Scientific Research (FNRS – F.R.S.).

## References


1. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
2. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
3. Coron, J.-S.: A New DPA Countermeasure Based on Permutation Tables. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 278–292. Springer, Heidelberg (2008)
4. Dziembowski, S., Pietrzak, K.: Leakage-Resilient Cryptography. In: Proceedings of FOCS 2008, Washington, DC, USA, October 2008, pp. 293–302 (2008)
5. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES Implementation on a Grain of Sand. IEE Proceedings on Information Security 152(1), 13–20 (2005)
6. Feldhofer, M., Popp, T.: Power Analysis Resistant AES Implementation for Passive RFID Tags. In: Proceedings of Austrochip 2008, Linz, Austria, October 8, 2007, pp. 1–6 (October 2008), ISBN 978-3-200-01330-8
7. Goubin, L., Patarin, J.: DES and Differential Power Analysis: the Duplication Method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
8. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, Berlin (2004)
9. Herbst, C., Oswald, E., Mangard, S.: An AES Smart Card Implementation Resistant to Power Analysis Attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252. Springer, Heidelberg (2006)
10. Hutter, M., Medwed, M., Hein, D., Wolkerstorfer, J.: Attacking ECDSA-enabled RFID Devices. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 519–534. Springer, Heidelberg (2009)
11. International Organisation for Standardization (ISO), ISO/IEC 9798-2: Information technology – Security techniques – Entity authentication – Mechanisms using symmetric encipherment algorithms (1999)
12. Kirschbaum, M., Popp, T.: Private Communication (2009)
13. Kocher, P.: Leak Resistant Cryptographic Indexed Key Update, US Patent 6539092
14. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
15. McEvoy, R.P., Tunstall, M., Whelan, C., Murphy, C.C., Marnane, W.P.: All-or-Nothing Transforms as a Countermeasure to Differential Side-Channel Analysis, Cryptology ePrint Archive, Report 2009/185, http://eprint.iacr.org/2009/185
16. Macé, F., Standaert, F.-X., Quisquater, J.-J.: Information Theoretic Evaluation of Side-Channel Resistant Logic Styles. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 427–442. Springer, Heidelberg (2007)
17. Mangard, S.: A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 343–358. Springer, Heidelberg (2003)
18. Mangard, S., Popp, T., Gammel, B.M.: Side-Channel Leakage of Masked CMOS Gates. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 351–365. Springer, Heidelberg (2005)
19. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks. Springer, Heidelberg (2007)

20. Mangard, S., Oswald, E., Standaert, F.-X.: One for All, All for One: Unifying Standard DPA Attacks, Cryptology ePrint Archive, Report 2009/449 (2009)
21. Messerges, T.S.: Using Second-Order Power Analysis to Attack DPA Resistant Software. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000)
22. Petit, C., Standaert, F.-X., Pereira, O., Malkin, T.G., Yung, M.: A Block Cipher based PRNG Secure Against Side-Channel Key Recovery. In: The Proceedings of ASIACCS 2008, Tokyo, Japan, March 2008, pp. 56–65 (2008)
23. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
24. Pietrzak, K.: Provable Security for Physical Cryptography. In: The Proceedings of WEWORC 2009, Graz, Austria (July 2009) (invited talk)
25. Piret, G., Quisquater, J.-J.: A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003)
26. Popp, T., Kirschbaum, M., Zefferer, T., Mangard, S.: Evaluation of the Masked Logic Style MDPL on a Prototype Chip. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 81–94. Springer, Heidelberg (2007)
27. Prouff, E., McEvoy, R.P.: First-Order Side-Channel Attacks on the Permutation Tables Countermeasure. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 81–96. Springer, Heidelberg (2009)
28. Renauld, M., Standaert, F.-X.: Algebraic Side-Channel Attacks, Cryptology ePrint Archive, Report 2009/279, http://eprint.iacr.org/2009/279
29. Renauld, M., Standaert, F.-X., Veyrat-Charvillon, N.: Algebraic Attacks on the AES: Why Time also Matters in DPA. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 97–111. Springer, Heidelberg (2009)
30. Rivain, M., Prouff, E., Doget, J.: Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 171–188. Springer, Heidelberg (2009)
31. Schramm, K., Paar, C.: Higher Order Masking of the AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 208–225. Springer, Heidelberg (2006)
32. Standaert, F.-X., Gierlichs, B., Verbauwhede, I.: Partition vs. Comparison Side-Channel Distinguishers: an Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 253–267. Springer, Heidelberg (2009)
33. Standaert, F.-X., Pereira, O., Yu, Y., Quisquater, J.-J., Yung, M., Oswlad, E.: Leakage Resilient Cryptography in Practice, Cryptology ePrint Archive, Report 2009/341 (2009), http://eprint.iacr.org/2009/341
34. Standaert, F.-X.: How Leaky is and Extractor? In: Workshop on Provable Security against Side-Channel Attacks, Leiden, The Netherlands (February 2010)
35. Tiri, K., Akmal, M., Verbauwhede, I.: Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand DPA on Smart Cards. In: The Proceedings of ESSCIRC 2002, Florence, Italy, September 2002, pp. 403–406 (2002)
36. Tiri, K., Verbauwhede, I.: A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In: The Proceedings of DATE 2004, Paris, France, February 2004, vol. 1, pp. 10246–10251 (2004)

# Strong Cryptography from Weak Secrets

## Building Efficient PKE and IBE from Distributed Passwords

Xavier Boyen[1], Céline Chevalier[2], Georg Fuchsbauer[3], and David Pointcheval[3]

[1] Université de Liège, Belgium
[2] Telecom ParisTech, Paris, France
[3] École Normale Supérieure, CNRS-INRIA, Paris, France

**Abstract.** Distributed-password public-key cryptography (DPwPKC) allows the members of a group of people, each one holding a small secret password only, to help a leader to perform the private operation, associated to a public-key cryptosystem. Abdalla *et al.* recently defined this tool [1], with a practical construction. Unfortunately, the latter applied to the ElGamal decryption only, and relied on the DDH assumption, excluding any recent pairing-based cryptosystems. In this paper, we extend their techniques to support, and exploit, pairing-based properties: we take advantage of pairing-friendly groups to obtain efficient (simulation-sound) zero-knowledge proofs, whose security relies on the Decisional Linear assumption. As a consequence, we provide efficient protocols, secure in the standard model, for ElGamal decryption as in [1], but also for Linear decryption, as well as extraction of several identity-based cryptosystems [6,4]. Furthermore, we strenghten their security model by suppressing the useless testPwd queries in the functionality.

## 1 Introduction

Recently, Abdalla *et al.* [1] proposed the notion of distributed-password public-key cryptography (DPwPKC), which allows the members of a group of people, each one holding a small independent secret password, to act collectively (for the benefit of one of them, who "owns" the group) as the custodian of a private key in some ordinary public-key cryptosystem — without relying on any secure (secret and/or authentic) storage — as long as each member remembers his or her password. Precisely, in DPwPKC, the members initially create a "virtual" key pair $(\mathsf{sk}, \mathsf{pk})$, by engaging in some distributed protocol over adversarial channels, where only $\mathsf{pk}$ is revealed, while $\mathsf{sk}$ is implicitly determined by the collection of passwords. Third parties can perform the public-key operation(s) of the underlying system using $\mathsf{pk}$. Members can help the leader of the group perform private-key operation(s) in a distributed manner, by engaging in some protocol using only their knowledge of their respective passwords.

Password-based public-key cryptography is generally considered infeasible because password-based secret-key spaces are easy to enumerate, and the knowledge of the public key makes it possible to test the correct key from that space,

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 297–315, 2010.

without interacting with anyone (offline dictionary attack). In DPwPKC, there are as many passwords as participants, and (unlike in virtually all applications of passwords) the passwords are not meant to be shared: they are chosen independently by each player. Since the passwords need not be related, they will likely be diverse, and the min-entropy of their *combination* ought to grow linearly with the number of participants, even if every single password is itself minuscule. For instance, with ten players each holding a random 20-bit password, the virtual secret key will be a random 200-bit string, which is more than enough to build a secure public-key system for usual values of the security parameter. This is what makes $\mathsf{sk}$ in DPwPKC resistant to brute-force off-line dictionary attacks, even though the corresponding $\mathsf{pk}$ is public.

The main contribution of [1] was to define general functionalities for distributed password-based key generation and private computation in the UC model, and to give a construction for ElGamal decryption as a proof of concept. However, the construction proposed in [1] was merely illustrative because it required generic simulation-sound non-interactive zero-knowledge ($\mathsf{SSNIZK}$) proofs for NP languages, which can only be performed efficiently in the random oracle model [3]. Furthermore, their distributed private computation protocol could only perform the task of computing $c^{\mathsf{sk}}$ from the implicit secret key $\mathsf{sk}$, and the security of their protocol relied on the DDH assumption. Together, these restrictions limited its applicability to ElGamal decryption.

In this work, we first improve and strengthen the ideal functionalities defined in [1], by further restricting the information that the adversary can gain from an attack. This will make any protocol that we can prove to realize those functionalities stronger, since the simulation will have to work without this information. (Recall that in the UC model, the functionalities are supposed to capture everything that we allow the adversary (and thus the simulator) to learn.)

Then, we extend the techniques from [1] to support a much broader class of private-key operations in discrete-log-hard groups, including operations involving random ephemerals and/or operations in bilinear groups. More precisely, our construction still targets the distributed computation of $c^{\mathsf{sk}}$, but under the Decision Linear assumption, which makes the proof more intricate since the DDH is now verifiable: we had to change the workings of the protocol to introduce secret values. Furthermore, the construction works for several values of $c$ at once, and now allows to share random ephemerals in the exponent. It thus allows a much greater variety of public-key cryptosystems to be converted to distributed password-based cryptosystems, including extraction of identity-based private keys — thus giving us the new interesting notion of "password-based distributed identity-based encryption" (DPwIBE). Contrarily to regular IBE, the "central" key extraction authority is now distributed among a group of people (sufficiently many of them trusted), with the "master key" being implicitly contained in the collections of short independent passwords held by those users.

In the process of strengthening and generalizing the protocols, we also make them much more efficient. To do so, we develop special-purpose *simulation-sound* non-interactive zero-knowledge proofs ($\mathsf{SSNIZK}$) for our languages of interest,

*in the standard model*, and show how to use them instead of the inefficient general SSNIZK considered in [1]. We do this using bilinear maps, in the CRS model, relying on a classic decisional hardness assumption for bilinear groups. The SSNIZK proofs we construct revisit the techniques of [12] and use efficient proofs inspired by the recent Groth-(Ostrovsky)-Sahai sequence of efficient NIZK construction in bilinear groups [14], but do not trivially follow from them.

A number of new technical challenges had to be solved. We specifically mention the following: 1) the use of pairings not only helps us make efficient zero-knowledge proofs for various languages, it would also help the adversary verify the result of the private computation $c^{\mathsf{sk}}$ in the basic DPwPKC protocol from [1]. Since the UC model requires that the simulation be carried out until the end on both correct and incorrect inputs, this will make our new security reduction somewhat more intricate since the result sent at the end of the simulation is random and we do not want the adversary to become aware of it. 2) In connection with the stronger and simpler functionality definitions we propose, the adversary is no longer allowed to conduct *explicit* password compatibility tests prior to the private-key operation. This should intuitively further complicate the simulation, though we remarkably note that these queries were indeed useless in the proofs and thus getting rid of them has no negative impact. 3) Generally speaking, we achieved much of our security and efficiency gains over [1], by succeeding to make our protocols being *fully robust* by the use of public verifications (computations of pairings) rather than intermediate validity tests (SSNIZK proofs, relying on the random oracle model in [1]). This is generally both more efficient (no more SSNIZK proofs) and more secure than testing, but it can lead to significantly more complex simulations owing to the ideal functionality being less "helpful".

## 2 Security Model

**Split Functionalities.** Throughout this paper, we assume basic familiarity with the universal composability framework [9]. Without any strong authentication mechanisms, the adversary can always partition the players into disjoint subgroups and execute independent sessions of the protocol with each subgroup, playing the role of the other players. Such an attack is unavoidable since players cannot distinguish the case in which they interact with each other from the case where they interact with the adversary. The authors of [2] addressed this issue by proposing a new model based on *split functionalities* which guarantees that this attack is the only one available to the adversary.

The split functionality is a generic construction based upon an ideal functionality. In the initialization stage, the adversary $\mathcal{A}$ adaptively chooses disjoint subsets of the honest parties (with a unique session identifier that is fixed for the duration of the protocol). During the computation, each subset $H$ activates a separate instance of the functionality $\mathcal{F}$. All these functionality instances are independent: The executions of the protocol for each subset $H$ can only be related in the way $\mathcal{A}$ chooses the inputs of the players it controls. The parties $P_i \in H$ provide their own inputs and receive their own outputs, whereas $\mathcal{A}$ plays the role of all the parties $P_j \notin H$.

Note that the use of these split functionalities already allows the adversary to try some passwords for users by choosing subgroups of size 1 and trying a password for each of them while impersonating the other players. They are thus enough to model on-line dictionary attacks. In [1], additional TestPwd queries were available to the adversary, thus allowing additional password trials. In this paper, we limit the adversary against the ideal functionality (i.e. the simulator), to the unavoidable on-line dictionary attack but in the strict sense, and thus without any additional TestPwd queries. This means that we give less power to the simulator. Both the constructions in [1] and ours do not need them in the security proofs, which means that a stronger security level is reached.

In the sequel, as we describe our two general functionalities $\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$ and $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$ (the complete descriptions can be found in the full version [8]), one has to keep in mind that an attacker controlling the communication channels can always choose to view them as the split functionalities $s\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$ and $s\mathcal{F}_{\mathsf{pwDistPrivateComp}}$, which implicitly consist of multiple instances of $\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$ and $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$ for non-overlapping subsets of the original players. Furthermore, one cannot prevent $\mathcal{A}$ from keeping some flows, which will never arrive. This is modelled in our functionalities by a bit $\mathsf{b}$, which specifies whether the flow is really sent or not.

**The Players and the Group Leader.** We denote by $n$ the number of users involved in a given execution of the protocol. All the computation is done for the benefit of only one of them, denoted as the *group leader*. The role of all the other ones, the *players*, is to help it in its use of the group's virtual key. A group is thus formed arbitrarily and is defined by its composition, which cannot be changed: a leader, which is the only one to receive the result of a private computation in the end, and a (ordered or not, according to the secret key computation from the passwords) set of players to assist it.

**The Aim of the Functionalities.** The functionalities are intended to capture distributed-password protocols for (the key-generation and private-key operation of) an arbitrary public-key primitive, but taking into consideration the unavoidable on-line dictionary attacks. More precisely, the aim of the distributed key generation functionality $\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$ is to provide a public key to the users, computed according to their passwords with respect to a function PublicKeyGen given as parameter. Moreover, it ensures that the group leader never receives an incorrect key in the end, whatever the adversary does.

In the distributed private computation functionality $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$, the aim is to perform a private computation for the sole benefit of the group leader, which is responsible for the correctness of the computation; in addition, it is the only user to receive the end result. This functionality will thus compute a function of some supplied input $in$, depending on a set of passwords that must define a secret key corresponding to a given public key. More precisely, it will be able to check the compatibility of the passwords with the public key thanks to a verification function PublicKeyVer, and if it is correct it will then compute the secret key sk from the passwords with the help of a function SecretKeyGen, and from there evaluate PrivateComp(sk, $in$) and give the result to the leader.

The function PrivateComp could be the decryption function Dec of a public-key encryption scheme, or the signing function Sign in a signature scheme, or the identity-based key extraction function Extract in an IBE system.

Note that SecretKeyGen and PublicKeyVer are naturally related to the function PublicKeyGen called by the former functionality. In all generality, unless SecretKeyGen and PublicKeyGen are both assumed to be deterministic, we need the predicate PublicKeyVer in order to verify that a public key is "correct" without necessarily being "equal" (to some canonical public key). Also note that the function SecretKeyGen is not assumed to be injective, lest it unduly restrict the number of users and the total size of their passwords. The distributed computations should not reveal more information than the non-distributed ones, and thus the ideal functionalities can make use of these functions as black-boxes.

**The Functionalities.** We only recall here the main points of the functionalities, referring the interested reader to [1] for details. But, importantly, as in [10], the functionalities are not in charge of providing the passwords to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where an honest user executes the protocol with an incorrect password: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where related passwords are used in different protocols.

The private-computation functionality fails directly at the end of the initialization phase if the users do not share the same (public) inputs. In principle, after the initialization stage (the NewSession queries) is over, the eligible users are ready to receive the result. However the functionality waits for the adversary $\mathcal{S}$ to send a compute message before proceeding. This allows $\mathcal{S}$ to decide the exact moment when the result should be sent to the users and, in particular, it allows $\mathcal{S}$ to choose the exact moment when corruptions should occur (for instance $\mathcal{S}$ may decide to corrupt some party $P_i$ before the result is sent but after $P_i$ decided to participate to a given session of the protocol; see [15]). Also, although in the key generation functionality all users are normally eligible to receive the public key, in the private computation functionality it is important that only the group leader receives the output (though he may choose to reveal it afterwards to others, outside of the protocol, depending on the application). In both cases, after the result is computed, $\mathcal{S}$ can choose whether the group leader indeed receives it. If delivery is denied ($\mathsf{b} = 0$), then nobody gets it, and it is as if it was never computed. Otherwise, in the first functionality, the other players may be allowed to receive it too, according to a schedule chosen by $\mathcal{S}$.

Note that given the public key, if the adversary knows/controls sufficiently many passwords so that the combined entropy of the remaining passwords is low enough, he will be able to recover these remaining passwords by brute force attack. This is unavoidable and has nothing to do with the fact that the system is distributed: off-line attacks are always possible in principle in public-key systems, and become feasible as soon as a sufficient portion of the private key is known.

## 3 Notations and Building Blocks

The authors of [1] propose a protocol that deals with a particular case of unauthenticated distributed private computation [2], as captured by their functionalities recalled in the former section. Informally, assuming $s$ to be a secret key, the aim of the protocol is to compute a value $c^s$ given an element $c$ of the group. They claim that this computation can be used to perform distributed BLS signatures [7], ElGamal decryptions [11], linear decryptions [5], and BF or BB1 identity-based key extraction [6,4] but they only focus on ElGamal decryptions, relying on the DDH assumption.

Here, we show how to really achieve such results, by constructing a protocol relying on the Decision Linear assumption [5] for compatibility with bilinear groups. This protocol will easily enable "password-based" Boneh-Franklin IBE scheme [6]. In the following section, we show how to modify the protocol to obtain "password-based" Boneh-Boyen ($BB_1$) IBE scheme [4] and linear decryptions [5].

**Notations.** Let $\mathbb{G}$ be a multiplicative cyclic group of prime order $p$ and $g_3$ a generator of $\mathbb{G}$. The linear encryption works as follows: The private key is a pair of scalars, $\mathsf{sk}_{\text{lin}} = (x_1, x_2)$, and the public key, $\mathsf{pk}_{\text{lin}} = (g_1, g_2, g_3)$, where $g_1 = {g_3}^{1/x_1}$, $g_2 = {g_3}^{1/x_2}$. In order to encrypt $M \in \mathbb{G}$, one chooses $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$, and the ciphertext consists of $C = \mathcal{E}_{\mathsf{pk}_{\text{lin}}}(M; r_1, r_2) = (C_1, C_2, C_3) = ({g_1}^{r_1}, {g_2}^{r_2}, M{g_3}^{r_1+r_2})$. The decryption process consists of $M = \mathcal{D}_{\mathsf{pk}_{\text{lin}}}(C) = C_3/({C_1}^{x_1}{C_2}^{x_2})$.

This encryption scheme is secure under the Decisional Linear ($\mathsf{DLin}$) assumption, first presented in [5] and stated here for completeness: For random $x, y, r, s \in \mathbb{Z}_p^*$ and $(g, f = g^x, h = g^y, f^r, h^s) \in \mathbb{G}^5$, it is computationally intractable given $g^d$ to distinguish between the case where $d = r + s$ or $d$ is random. More precisely, a triple $(f^r, h^s, g^d)$ is named a *linear triple* in basis $(f, h, g)$ if $d = r + s$. We also consider a one-time signature scheme consisting of the three algorithms $(\mathsf{SKG}, \mathsf{Sign}, \mathsf{Ver})$.

**Passwords, Public Key and Private Key.** Each user $P_i$ owns a privately selected password $\mathsf{pw}_i$, to act as the $i$-th share of the secret key $\mathsf{sk}$ (see below). For convenience, we write $\mathsf{pw}_i = \mathsf{pw}_{i,1} \ldots \mathsf{pw}_{i,\ell} \in \{0, \ldots, 2^\ell - 1\}$, i.e., we further divide each password $\mathsf{pw}_i$ into $\ell$ bits $\mathsf{pw}_{i,j}$, where $p < 2^\ell$ ($p$ is the order of the group $\mathbb{G}$). Notice that although we allow full-size passwords of up to $\ell$ bits (the size of $p$), users are of course permitted to choose shorter passwords.

The authors of [1] discussed the use of such passwords to combine properly into a private key $\mathsf{sk}$: the combination should be reproducible, it should allow to recover either of the passwords from the key and the other passwords, and it should preserve the joint entropy of the set of paswords. They also discussed possible cancellation or aliasing effects of the passwords. The preferable solution is to do standard pre-processing using hashing, i.e. that each user independently transforms his or her true password $\mathsf{pw}_i^*$ into an effective password $\mathsf{pw}_i$ by applying a suitable extractor $\mathsf{pw}_i = \mathsf{H}(i, \mathsf{pw}_i^*, Z_i)$ where $Z_i$ is any relevant public information. We can then safely take $\mathsf{sk} = \sum_i \mathsf{pw}_i$ and be assured that the entropy of $\mathsf{sk}$ will closely match the joint entropy of the vector $(\mathsf{pw}_1^*, \ldots, \mathsf{pw}_n^*)$.

The discrete-log-based key pair $(\mathsf{sk}, \mathsf{pk} = g^{\mathsf{sk}})$ is then defined as follows:

$$\mathsf{sk} = \mathsf{SecretKeyGen}(\mathsf{pw}_1, \ldots, \mathsf{pw}_n) \stackrel{\mathsf{def}}{=} \sum_{i=1}^{n} \mathsf{pw}_i$$
$$\mathsf{pk} = \mathsf{PublicKeyGen}(\mathsf{pw}_1, \ldots, \mathsf{pw}_n) \stackrel{\mathsf{def}}{=} g^{\sum \mathsf{pw}_i}$$

The password/public-key verification function is then

$$\mathsf{PublicKeyVer}(\mathsf{pw}_1, \ldots, \mathsf{pw}_n, \mathsf{pk}) \stackrel{\mathsf{def}}{=} \left(\mathsf{pk} \stackrel{?}{=} g^{\sum \mathsf{pw}_i}\right).$$

In the following, we focus on a specific format for the PrivateComp function, defined by $(\mathsf{sk}, c) \mapsto m = c^{\mathsf{sk}}$. We show how to perform it in a distributed way, and how to use if for decryption processes, and private key extraction in IBE.

**Building Blocks**

Extractable Homomorphic Commitments. As in [1], the first step of our distributed decryption protocol is for each user to commit to his password (the details are given in the following section). The commitment needs to be extractable, homomorphic, and compatible with the shape of the public key. Generally speaking, one needs a commitment $\mathsf{Commit}(\mathsf{pw}, R)$ that is additively homomorphic on pw and with certain properties on $R$. Instead of ElGamal's scheme [11] used in [1], we focus here on linear commitments $\mathsf{Commit}_g(\mathsf{pw}, r, s) = (U_1{}^{\mathsf{pw}} g_1{}^r, U_2{}^{\mathsf{pw}} g_2{}^s, g^{\mathsf{pw}} g_3{}^{r+s})$, where $(U_1, U_2, U_3 = g)$ is *not* a linear triple in basis $(g_1, g_2, g_3)$ in order to provide extractability, or encryptions $\mathsf{Encrypt}_g(\mathsf{pw}, r, s) = (g_1{}^r, g_2{}^s, g^{\mathsf{pw}} g_3{}^{r+s})$ (here, $g_1$, $g_2$ and $g_3$ are defined as before and $g$ is a generator of $\mathbb{G}$). In both cases, the hiding property or the semantic security rely on the DLin assumption. Extractability is possible granted the private/decryption key $(x_1, x_2)$, such that $g_3 = {g_1}^{x_1} = {g_2}^{x_2}$, and recalling that the users commit to bits. Denoting by $(c_1, c_2, c_3)$ the commitment, it is thus enough to check that $c_3/({c_1}^{x_1} {c_2}^{x_2}) = 1$ or $(c_3/g)/((c_1/U_1)^{x_1} (c_2/U_2)^{x_2}) = 1$.

Proofs of Membership. For the robustness and soundness of the protocols, we need some proofs of honest computations. We use witness-indistinguishable and SSNIZK proofs/arguments. The difficulty consists in designing such *simulation-sound* proofs without random oracles: they are described in Section 6. Along these lines, we use the following kinds of *non-interactive* proofs:

- $\mathsf{CDH}(g, G, h, H)$, to prove that $(g, G, h, H)$ lies in the CDH language: there exists a common exponent $x$ such that $G = g^x$ and $H = h^x$. Granted pairing-friendly groups, this can be easily done by simple pairing computations;
- $\mathsf{WIProofBit}(C)$, to prove that the commitment or the ciphertext $C$ contains a bit. We will use a WI proof from [13], which basically proves that either $C$ or $C$ divided by the basis is a linear 3-tuple;
- $\mathsf{SSNIZKEq}_{g,c}(C_1, C_2)$, to prove that the ciphertexts/commitments $C_1$ and $C_2$ contain the same value, possibly in the different bases $g$ and $c$, that is, $C_1$ encrypts/commits to $g^a$ and $C_2$ encrypts/commits to $c^a$, with the same $a$. We use a SSNIZK argument, following the overall approach by Groth [12] to obtain simulation soundness, but using the Groth-Sahai proof system [14] for efficiency (see Section 6 – the proof is omitted, but very similar to [12]).

## 4 Description of the Protocols

**The Distributed Key Generation Protocol.** This protocol is described in Figure 1 and realizes the functionality $\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$. All the users are provided with a password $\mathsf{pw}_i$ and want to obtain a public key $\mathsf{pk}$. One of them is the leader of the group, denoted by $P_1$, and the others are $P_2, \ldots, P_n$.

The protocol starts with a round of commitments of these passwords. Each user sends a commitment $C_i$ of $\mathsf{pw}_i$ (divided into $\ell$ blocks $\mathsf{pw}_{1,1}, \ldots, \mathsf{pw}_{i,\ell}$ of length $L$ — here, $L = 1$): it computes $C_{i,j} = (C_{i,j}^{(1)}, C_{i,j}^{(2)}, C_{i,j}^{(3)}) = (U_1{}^{\mathsf{pw}_{i,j}} g_1{}^{r_{i,j}}, U_2{}^{\mathsf{pw}_{i,j}} g_2{}^{s_{i,j}}, g^{\mathsf{pw}_{i,j}} g_3{}^{r_{i,j}+s_{i,j}})$ for $j = 1, \ldots, \ell$ and random values $r_{i,j}$ and $s_{i,j}$, and publishes $\mathbf{C_i} = (C_{i,1}, \ldots, C_{i,\ell})$, with a set of proofs $\mathsf{WIProofBit}(C_{i,j})$ that each commitment indeed commits to an $L$-bit block. As we see in the proof (see the full version), this commitment needs to be extractable so that the simulator is able to recover the passwords used by the adversary, which is the reason why we segmented all the passwords and make commitments of bits, along with a $\mathsf{WIProofBit}$ that the committed value is actually a bit. Each user also runs the signature key generation algorithm to obtain a signature key $\mathsf{SK}_i$ and a verification key $\mathsf{VK}_i$. The users will be split according to the values received in this first flow (i.e. the commitments, the proofs and the verification keys), as we see in the second flow where they send a signature of all they have received up to this point. Thus, the protocol cannot continue past this point if some players do not share the same values as the others (i.e. one of the signatures $\sigma_i$ will be rejected later on and at least a user will abort).

Once this first step is done, the users commit again to their passwords (by encrypting them, for efficiency reasons), but this time in a single block: $C'_i = (C_i'^{(1)}, C_i'^{(2)}, C_i'^{(3)}) = (g_1{}^{t_i}, g_2{}^{u_i}, g^{\mathsf{pw}_i} g_3{}^{t_i+u_i})$ (with random values $t_i$ and $u_i$) and publish it along with a $\mathsf{SSNIZK}$ proof that the passwords committed are the same in the two commitments: $\mathsf{SSNIZKEq}_{g,g}(C_i, C'_i)$, $C_i$ roughly being the product of the $C_{i,j}$, i.e. a commitment of $\mathsf{pw}_i$. The new encryptions $C'_i$ will be the ones used in the rest of the protocol. They need not be segmented (since we will not extract anything from them, but just make computations on encrypted values), but we ask the users to prove that they are compatible with the former commitments.

Each user $P_i$ computes $H = \mathcal{H}(\mathbf{C_1}, \ldots, \mathbf{C_n})$, and sends a signature of the values that identifies this execution, under an ephemeral one-time signature key, to avoid malleability and replay from previous sessions: $\sigma_i = \mathsf{Sign}(H; \mathsf{SK}_i)$. This allows the protocol to realize the split functionality by ensuring that everybody has received the same values in the first round (more precisely, the players have been split according to what they received in the first round, so that we can assume that they have all received the same values). Note that the protocol will fail if the adversary drops or modifies a flow received by a user, even if everything was correct. This situation is modeled by the bit $\mathsf{b}$ of the key delivery queries in the functionality, for when everything goes well but some of the players do not obtain the result.

The need for an additional extractable commitment $C_i$ of $g^{\mathsf{pw}_i}$ (and a proof that the password used is the same, and that everybody received the same value) is a requirement of the UC model, as in [10]. Indeed, we show later on that $\mathcal{S}$ needs to be able to simulate everything without knowing any passwords: Thus, he recovers the passwords by extracting them from the commitments $\mathbf{C_i}$ made by the adversary in the first round, enabling him to adjust his own values before the subsequent encryptions $C'_i$, so that all the passwords are compatible with the public key (if they should be in the situation at hand).

After these rounds of commitments/encryptions, the players check the signatures and abort if one of them is not valid. A computation step then allows them to compute the public key. Note that everything has become publicly verifiable.

Computation starts from the ciphertexts $C'_i$, and involves two "blinding rings" to raise sequentially the values $\prod_i C_i'^{(3)} = g^{\sum_i \mathsf{pw}_i} g_3^{\sum_i (t_i+u_i)}$, $g_1$, $g_2$ and $g_3$ to some distributed random exponent $\alpha = \sum_i \alpha_i$. The players then broadcast ${g_3}^{\alpha(t_i+u_i)}$ (the values $g_1$ and $g_2$ are only here to check the consistency of the values $t_i$ and $u_i$ and avoid cheating), leaving every player able to compute $g^{\alpha \sum_i \mathsf{pw}_i}$. A final "unblinding" allows for the recovery of $g^{\sum_i \mathsf{pw}_i} = \mathsf{pk}$. We stress that every user is able to check the validity of this computation (at each step, it checks the $\mathsf{CDH}$ values to ensure that the same exponent was used each time): A dishonest execution cannot continue without an honest user becoming aware of it (and aborting). Note however that an honest execution can also be stopped by a user if the adversary modifies a flow, as reflected by the bit $\mathsf{b}$ in the functionality.

**The Distributed Private Computation Protocol.** This protocol is presented in Figure 2 and realizes $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$. Here, in addition to their passwords, the users are also provided a public key $\mathsf{pk}$ and a group element $c \in \mathbb{G}$. For this given $c \in \mathbb{G}$, the leader wants to obtain $m = c^{\mathsf{sk}}$. A big difference with the previous protocol is that this result will be private to the leader. But before computing it, everybody wants to be sure that all the users are honest, or at least that the combination of the passwords is compatible with the public key.

This verification step is exactly the same as the computation step in the previous protocol. The protocol starts by verifying that they will be able to perform this computation, and thus that they indeed know a representation of the secret key into shares. Each user sends a commitment $\mathbf{C_i} = \{C_{i,j}\}_j$ of its password as before, and the associated set of $\mathsf{WIProofBit}(C_{i,j})$.

As in the former protocol, once this first step (which enables the users to be split into subgroups according to what values they have received) is done, the users commit again to their passwords in the value $C'_i$, which will be the ones used in the rest of the protocol, and also send a signature which enables them to check that they share the same public key $\mathsf{pk}$, the same group element $c$, and have received the same values in the first round. It thus avoids situations in which a group leader with an incorrect key obtains a correct private computation result, contrary to the ideal functionality. The protocol will thus fail if all these values are not the same to everyone, which is the result required by the functionality.

Next, the users make yet another encryption $A_i$ of their passwords, but this time they do a linear encryption of $\mathsf{pw}_i$ in base $c$ instead of in base $g$ (in the

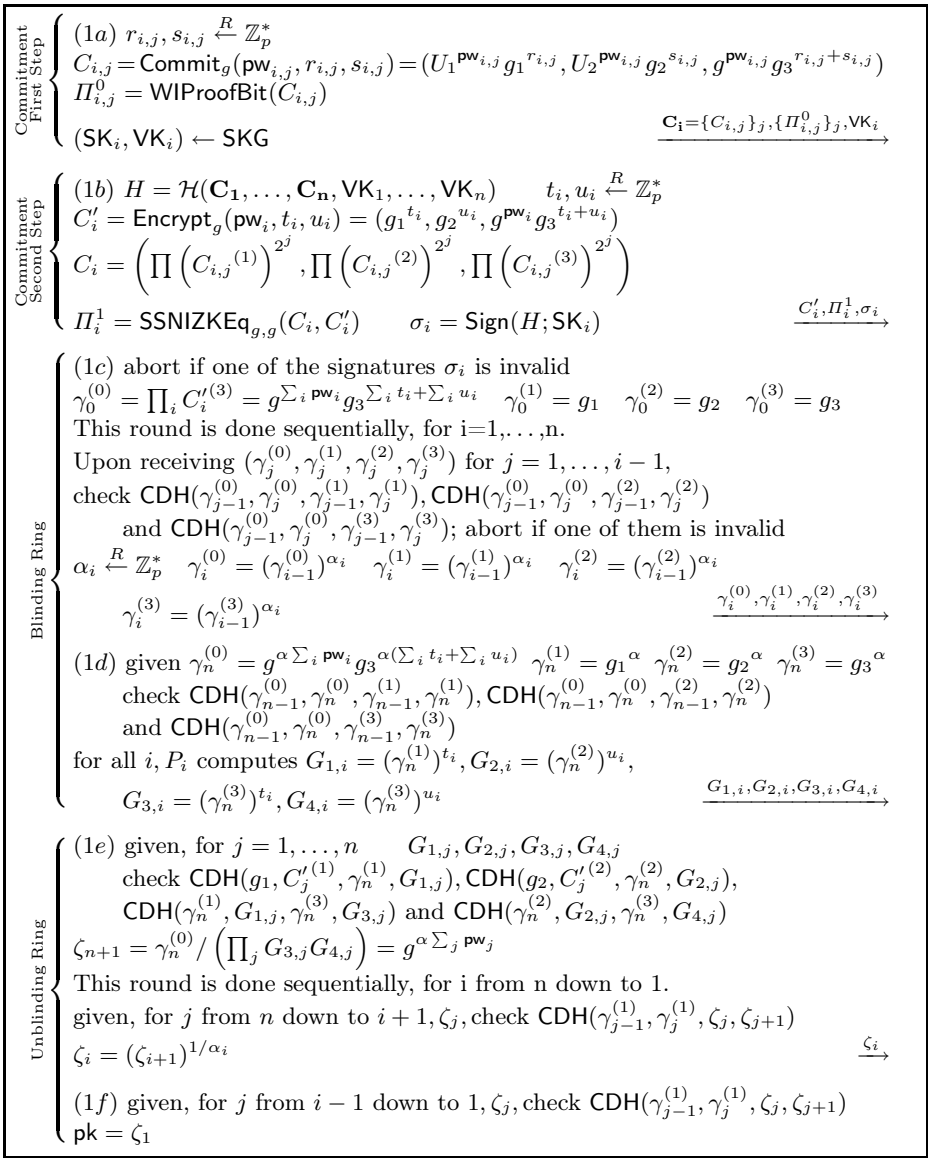



**Fig. 1.** Individual steps of the distributed key generation protocol

above $C'_i$ ciphertext): $A_i = \mathsf{Encrypt}_c(\mathsf{pw}_i, v_i, w_i) = (g_1{}^{v_i}, g_2{}^{w_i}, c^{\mathsf{pw}_i} g_3{}^{v_i+w_i})$. The ciphertexts $C'_i$ will be used to check the possibility of the private computation (i.e. that the passwords are consistent with the public key $\mathsf{pk} = g^{\mathsf{sk}}$), whereas the ciphertexts $A_i$ will be used to actually compute the expected result $c^{\mathsf{sk}}$, hence the two different bases $g$ and $c$ in $C'_i$ and $A_i$, respectively. All the users send

these last two ciphertexts to everybody, along with a SSNIZK argument that the same password was used each time: $\Pi_i^2 = \mathsf{SSNIZKEq}_{g,c}(C'_i, A_i)$.

After these rounds of commitments/encryptions, a verification step allows for all the players to check whether the public key and the passwords are compatible. Note that at this point, everything has become publicly verifiable so that the group leader will not be able to cheat and make the other players believe that everything is correct when it is not. Verification starts from the ciphertexts $C'_i$, and involves a blinding and an unblinding ring as described above. This ends with a decision by the group leader on whether to abort the protocol (when the passwords are incompatible) or go on to the computation step. Every user is able to check the validity of the group leader's decision, as in the former protocol.

If the group leader decides to go on, the players assist it in the computation of $c^{\mathsf{sk}}$, again with the help of a blinding and an unblinding rings, starting from the ciphertexts $A_i$. However, note that this time, the group leader does not reveal the values $G'_{1,1} = (\delta_n^{(1)})^{v_1}$, $G'_{2,1} = (\delta_n^{(2)})^{w_1}$, $G'_{3,1} = (\delta_n^{(3)})^{v_1}$ and $G'_{4,1} = (\delta_n^{(3)})^{w_1}$ at the end of the blinding ring, but it is the only one able to compute $c^{\beta \sum_j \mathsf{pw}_j}$. Instead of revealing it to the others, it chooses at random an exponent $x \xleftarrow{R} \mathbb{Z}_q^*$ and broadcasts the value $c^{\beta x \sum_j \mathsf{pw}_j}$. The unblinding ring then takes place as before, leading to a public value $c^{\beta_1 x \sum_j \mathsf{pw}_j}$ that the environment cannot distinguish from random thanks to the random exponent $x$. Furthermore, the whole process is robust, which means that nobody can make the decryption result become incorrect. Except of course the group leader itself who broadcasts any value it wants as $\zeta'_{n+1}$, without having to prove anything. But this does not help it to obtain a computation which it could not do alone, except the result $c^{\mathsf{sk}}$.

Note that if at some point a user fails to send its value (denial of service attack) or if the adversary modifies a flow (man-in-the-middle attack), the protocol will fail. In the ideal world this means that the simulator makes a computation delivery query to the functionality with a bit $\mathsf{b}$ set to zero. Because of the public verifications of the CDH values, in these blinding/unblinding rounds exactly the same sequence of passwords as in the first rounds has to be used by the players. This necessarily implies compatibility with the public key, but may be an even stronger condition.

As a side note, observe that all the blinding rings in the verification and computation steps could be made concurrent instead of sequential, to simplify the protocol. Notice however that the final unblinding ring of $c^{\mathsf{sk}}$ in the computation step should only be carried out after the public key and the committed passwords are known to be compatible, and the passwords to be the same in both sequences of commitments/encryptions, i.e. after the verification step succeeded.

All the witness-indistinguishable and SSNIZK proofs and arguments will be described in Section 6. We show in the full version [8] that we can *efficiently* simulate these computations without the knowledge of the $\mathsf{pw}_i$'s, so that they do not reveal anything more about the $\mathsf{pw}_i$'s than $\mathsf{pk}$ already does. More precisely, we show that such computations are indistinguishable to $\mathcal{A}$ under the DLin assumption.

**Commitment Steps**

$(2a) = (1a)$ $\xrightarrow{\{C_{i,j}, \Pi^0_{i,j}\}_j}$

$(2b) = (1b)$ except $v_i, w_i \overset{R}{\leftarrow} \mathbb{Z}_p^*$

$A_i = \mathsf{Encrypt}_c(\mathsf{pw}_i, v_i, w_i) = (g_1{}^{v_i}, g_2{}^{w_i}, c^{\mathsf{pw}_i} g_3{}^{v_i+w_i})$

$\Pi_i^2 = \mathsf{SSNIZKEq}_{g,c}(C'_i, A_i)$ $\xrightarrow{C'_i, A_i, \Pi_i^2}$

**Blind. Ring**

$(2c) = (1c)$ $\xrightarrow{\gamma_i^{(1)}, \gamma_i^{(2)}, \Pi_i^2}$

$(2d) = (1d)$ $\xrightarrow{(G_{1,i}, G_{2,i}, G_{3,i}, G_{4,i})}$

**Unblind. Ring**

$(2e) = (1e)$ $\xrightarrow{\zeta_i}$

$(2f) = (1f)$ $\quad \mathsf{pk} \overset{?}{=} \zeta_1$

---

**Blinding Ring**

$(3a)$ abort if one of the signatures $\sigma_i$ is invalid

$\delta_0^{(0)} = \prod_i A_i{}^{(3)} = c^{\sum_i \mathsf{pw}_i} g_3{}^{\sum_i v_i + \sum_i w_i} \quad \delta_0^{(1)} = g_1 \quad \delta_0^{(2)} = g_2 \quad \delta_0^{(3)} = g_3$

$P_1$ chooses at random $\beta_1 \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and computes

$\delta_1^{(0)} = (\delta_0^{(0)})^{\beta_1} \quad \delta_1^{(1)} = (\delta_0^{(1)})^{\beta_1} \quad \delta_1^{(2)} = (\delta_0^{(2)})^{\beta_1} \quad \delta_1^{(3)} = (\delta_0^{(3)})^{\beta_1}$

This round is done sequentially, for i=2,...,n.

Upon receiving $(\delta_j^{(0)}, \delta_j^{(1)}, \delta_j^{(2)}, \delta_j^{(3)})$ for $j = 1, \ldots, i-1$,

check $\mathsf{CDH}(\delta_{j-1}^{(0)}, \delta_j^{(0)}, \delta_{j-1}^{(1)}, \delta_j^{(1)})$, $\mathsf{CDH}(\delta_{j-1}^{(0)}, \delta_j^{(0)}, \delta_{j-1}^{(2)}, \delta_j^{(2)})$,

$\mathsf{CDH}(\delta_{j-1}^{(0)}, \delta_j^{(0)}, \delta_{j-1}^{(3)}, \delta_j^{(3)})$; abort if one of them is invalid

$\beta_i \overset{R}{\leftarrow} \mathbb{Z}_p^*$

$\delta_i^{(0)} = (\delta_{i-1}^{(0)})^{\beta_i} \quad \delta_i^{(1)} = (\delta_{i-1}^{(1)})^{\beta_i} \quad \delta_i^{(2)} = (\delta_{i-1}^{(2)})^{\beta_i} \quad \delta_i^{(3)} = (\delta_{i-1}^{(3)})^{\beta_i}$ $\xrightarrow{\delta_i^{(1)}, \delta_i^{(2)}}$

$(3b)$ given $\delta_n^{(0)} = c^{\beta \sum_i \mathsf{pw}_i} g_3{}^{\beta(\sum_i v_i + \sum_i w_i)} \quad \delta_n^{(1)} = g_1{}^{\beta} \quad \delta_n^{(2)} = g_2{}^{\beta} \quad \delta_n^{(3)} = g_3{}^{\beta}$

check $\mathsf{CDH}(\delta_{n-1}^{(0)}, \delta_n^{(0)}, \delta_{n-1}^{(1)}, \delta_n^{(1)})$, $\mathsf{CDH}(\delta_{n-1}^{(0)}, \delta_n^{(0)}, \delta_{n-1}^{(2)}, \delta_n^{(2)})$

$\mathsf{CDH}(\delta_{n-1}^{(0)}, \delta_n^{(0)}, \delta_{n-1}^{(3)}, \delta_n^{(3)})$

for $i \neq 1$, $P_i$ computes $G'_{1,i} = (\delta_n^{(1)})^{v_i}, G'_{2,i} = (\delta_n^{(2)})^{w_i}$,

$G'_{3,i} = (\delta_n^{(3)})^{v_i}, G'_{4,i} = (\delta_n^{(3)})^{w_i}$ $\xrightarrow{G'_{1,i}, G'_{2,i}, G'_{3,i}, G'_{4,i}}$

**Unblinding Ring**

$(3c)$ given, for $j = 1, \ldots, n$ $\quad G'_{1,j}, G'_{2,j}, G'_{3,j}, G'_{4,j}$

check $\mathsf{CDH}(g_1, A_j{}^{(1)}, \delta_n^{(1)}, G'_{1,j})$, $\mathsf{CDH}(g_2, A_j{}^{(2)}, \delta_n^{(2)}, G'_{2,j})$,

$\mathsf{CDH}(\delta_n^{(1)}, G'_{1,j}, \delta_n^{(3)}, G'_{3,j})$ and $\mathsf{CDH}(\delta_n^{(2)}, G'_{2,j}, \delta_n^{(3)}, G'_{4,j})$

$P_1$ computes $G'_{1,1} = (\delta_n^{(1)})^{v_1}, G'_{2,1} = (\delta_n^{(2)})^{w_1}, G'_{3,1} = (\delta_n^{(3)})^{v_1}, G'_{4,1} = (\delta_n^{(3)})^{w_1}$

$P_1$ chooses at random $x \overset{R}{\leftarrow} \mathbb{Z}_p^*$

and computes $\zeta'_{n+1} = \left(\delta_n^{(0)} / \prod_j (G'_{3,j} G'_{4,j})\right)^x = c^{\beta x \sum_j \mathsf{pw}_j}$ $\xrightarrow{\zeta'_{n+1}}$

This round is done sequentially, for i from n down to 2.

Upon receiving, for $j$ from $n$ down to $i+1$ $\quad \zeta'_j$

check $\mathsf{CDH}(\delta_{j-1}^{(1)}, \delta_j^{(1)}, \zeta'_j, \zeta'_{j+1})$

$\zeta'_i = (\zeta'_{i+1})^{1/\beta_i}$ $\xrightarrow{\zeta'_i}$

$(3d)$ given, for $j$ from $i-1$ down to 2 $\quad \zeta'_j$

check $\mathsf{CDH}(\gamma_{j-1}^{(1)}, \gamma_j^{(1)}, \zeta'_j, \zeta'_{j+1})$

$P_1$ gets $\zeta'_1 = (\zeta'_2)^{1/\beta_1} = c^{x \sum \mathsf{pw}_i} = c^{x\mathsf{sk}}$ and finally $c^{\mathsf{sk}}$

**Fig. 2.** Individual steps of the distributed decryption protocol

**Security Theorems.** Assuming that the proofs of membership $\mathsf{WIProofBit}$ and $\mathsf{SSNIZKEq}$ are instantiated as described in Section 6 (relying on the $\mathsf{CDH}$), we have the following results, provided that $\mathsf{DLin}$ is infeasible in $\mathbb{G}$ and $\mathcal{H}$ is collision-resistant. The proofs of these theorems can be found in the full version [8].

**Theorem 1.** *Let $\widehat{\mathcal{F}}_{\mathsf{pwDistPublicKeyGen}}$ be the concurrent multi-session extension of $\mathcal{F}_{\mathsf{pwDistPublicKeyGen}}$. The distributed key generation protocol in Figure 7 securely realizes $\widehat{\mathcal{F}}_{\mathsf{pwDistPublicKeyGen}}$ for ElGamal key generation, in the CRS model, in the presence of static adversaries.*

**Theorem 2.** *Let $\widehat{\mathcal{F}}_{\mathsf{pwDistPrivateComp}}$ be the concurrent multi-session extension of $\mathcal{F}_{\mathsf{pwDistPrivateComp}}$. The distributed decryption protocol in Figure 2 securely realizes $\widehat{\mathcal{F}}_{\mathsf{pwDistPrivateComp}}$ for ElGamal decryption, in the CRS model, in the presence of static adversaries.*

As stated above, our protocols are only proven secure against static adversaries. Unlike adaptive ones, static adversaries are only allowed to corrupt protocol participants prior to the beginning of the protocol execution.

## 5 Extensions of the Protocols

**Boneh-Franklin IBE Scheme [6].** We need to compute $d_{\mathsf{id}} = H(\mathsf{id})^{\mathsf{sk}}$ where $H(\mathsf{id})$ is a public hash of a user's identity. This is analogous to $c^{\mathsf{sk}}$, and thus our protocol works as is.

**Boneh-Boyen ($\mathbf{BB_1}$) IBE Scheme [4].** Here, $d_{id}$ is randomized and of the form $({h_0}^{\mathsf{sk}}({h_1}^{\mathsf{id}}h_2)^r, {h_3}^r)$. Since $({h_0}^{\mathsf{sk}})$ is a private value, the protocol can be adapted as follows: 1) In the commitment steps, the user also commits (once) in $(2a)$ to a value $r_i$, which will be its share of $r$. 2) Up to $(2f)$, everything works as before in order to check $\mathsf{pk}$ (there is no need to check $r$, constructed on the fly). 3) The blinding rings are made in parallel, one for $({h_0}^{\mathsf{sk}})^\beta$, one for $(({h_1}^{\mathsf{id}}h_2)^r)^\beta$, and one for $({h_3}^r)^\beta$, the $\mathsf{CDH}$ being checked to ensure that the same $r$ and $\beta_i$ are used each time. 4) The players obtain $({h_0}^{\mathsf{sk}}({h_1}^{\mathsf{id}}h_2)^r)^\beta$ and the unblinding ring is made globally for this value. An unblinding ring is also done for $({h_3}^r)^\beta$, with the same verification for the exponents $\beta_i$.

**Linear Decryptions [5].** Let $(f = g^{1/x}, g, h = g^{1/y})$ be the public key of a linear encryption scheme, $(x, y)$ being the private key. Assuming $z = y/x$, these keys can be seen as $\mathsf{pk} = (h^z, h^y, h)$ and $\mathsf{sk} = (y, z)$. Using these notations,

$$c = \mathcal{E}_{\mathsf{pk}}(m; r) = (c_1, c_2, c_3) = (f^r, h^s, mg^{r+s})$$
$$m = \mathcal{D}_{\mathsf{sk}}(c) = c_3({c_1}^x {c_2}^y)^{-1} = mg^{r+s}g^{-r}g^{-s}$$

In the first protocol, the players need to use two passwords $z_i$ and $y_i$ to create the public key $\mathsf{pk}$. In the second one, the commitment steps are doubled to commit to both $z_i$ and $y_i$. As soon as $\mathsf{pk}$ is checked, the blinding rings are made separately, one for $({c_1}^x)^\beta$ and one for $({c_2}^y)^\beta$. The players obtain $({c_1}^x {c_2}^y)^\beta$ and the unblinding ring can be made globally for this value. In both rings, the $\mathsf{CDH}$ is checked to ensure that the same $\beta_i$ is used each time.

# 6 Employed Proof Systems

## 6.1 GOS WI Proof of Commitments Being to Bits

Let $(g_1, g_2, g_3) \in \mathbb{G}^3$ be a "basis" and let $(U_1, U_2, g) \in \mathbb{G}^3$ be a commitment key (which is in general non-linear w.r.t. $(g_1, g_2, g_3)$, but for simulation purposes it will be linear). Let $C = (U_1^x g_1^r, U_2^x g_2^s, g^x g_3^{r+s})$ be a commitment to $x$ using randomness $(r, s)$. Groth *et al.* [13] construct a WI proof system to show that one of two triples is linear. Applying it to $(C_1, C_2, C_3)$ and $(C_1 U_1^{-1}, C_2 U_2^{-1}, C_3 g^{-1})$ yields a proof that $x \in \{0, 1\}$, thus implements $\mathsf{WIProofBit}$, in an efficient way and without random oracles.

## 6.2 Simulation-Sound NIZK Arguments for Relations of Ciphertexts and Commitments

We construct two simulation-sound NIZK argument systems implementing the proof $\mathsf{SSNIZKEq}$. Given two ciphertexts, the first proves that the encrypted messages $m_1$ and $m_2$ are in $\mathsf{CDH}$ w.r.t. some fixed basis $(c, d)$, i.e., $m_1 = c^\mu$ and $m_2 = d^\mu$ for some $\mu$. The second $\mathsf{SSNIZK}$ proves that for a given linear commitment to $x$ and a linear encryption of $g^y$ it holds that $x = y$. We follow the overall approach by Groth [12] to obtain simulation soundness, but using the Groth-Sahai proof system [14] we get an efficient result: the proofs themselves are efficient, and we need not *encrypt* some of the witnesses in order to guarantee extractability, as the employed Groth-Sahai proofs are witness extractable.

**Overview.** We start with some intuition on how [12] constructs simulation-sound proofs for *satisfiability of a set of pairing product equations* (PPEs) $\{E_k\}_{k=1}^{K_E}$ (and later show how to express the statements we want to prove this way). Let $\Sigma_{\text{ot}}$ be a strong one-time signature scheme[1] and let $\Sigma_{\text{cma}}$ be a signature scheme that is existentially unforgeable under chosen message attack (EUF-CMA), and whose signatures $\sigma$ on a message $M$ are verified by checking a set of PPEs over a verification key $\mathsf{vk}$ and $M$, denoted $\{V_k(\mathsf{vk}, M, \sigma)\}_{k=1}^{K_V}$.

The common reference string (CRS) of our argument system will contain a verification key $\mathsf{vk}$ for $\Sigma_{\text{cma}}$ (whose corresponding signing key serves as simulation trapdoor). When making an argument, one first chooses a key pair $(\mathsf{vk}_{\text{ot}}, \mathsf{sk}_{\text{ot}})$ for $\Sigma_{\text{ot}}$, proves a statement and, at the end, adds a signature under $\mathsf{vk}_{\text{ot}}$ on the instance and the proof. The statement one actually proves is the following: to either know a witness satisfying Equations $\{E_k\}$ *or* to know a signature on $\mathsf{vk}_{\text{ot}}$ valid under $\mathsf{vk}$. Groth [12] shows how to construct a new set of equations which is satisfiable iff $\{E_k\}$ or $\{V_k(\mathsf{vk}, \mathsf{vk}_{\text{ot}}, \cdot)\}$ are satisfiable. Moreover, knowing witnesses for either of them, one can compute witnesses of the new set of equations. Using the techniques of [14], one then commits to the witnesses and proves that the committed values satisfy the new PPEs in a witness-indistinguishable (WI) way.

To simulate an argument, after choosing a pair $(\mathsf{vk}_{\text{ot}}, \mathsf{sk}_{\text{ot}})$, one uses the trapdoor to produce a signature $\sigma$ on $\mathsf{vk}_{\text{ot}}$ valid under $\mathsf{vk}$ and uses $\sigma$ as a witness

[1] A signature scheme is *strong one-time* if no adversary, after getting a signature $\sigma$ on *one* message $m$ of his choice, can produce a valid pair $(m^*, \sigma^*) \neq (m, \sigma)$.

for $\{V_k(\mathsf{vk}, \mathsf{vk}_{\mathrm{ot}}, \cdot)\}$. (It follows from WI of the Groth-Sahai proof that this is indistinguishable from using a witness for $\{E_k\}$.) Even after seeing many proofs of this kind, no adversary is able to produce one for a new false statement: Since it has to sign the instance and the argument at the end, it must choose a *new* pair $(\mathsf{vk}^*_{\mathrm{ot}}, \mathsf{sk}^*_{\mathrm{ot}})$ (by one-time security of $\Sigma_{\mathrm{ot}}$). Soundness of Groth-Sahai proofs imposes that to prove a false statement (meaning that the first clause of the disjunction is not satisfiable), it must use a witness for the second clause, thus know a signature on $\mathsf{vk}_{\mathrm{ot}}$. This however is infeasible by EUF-CMA of $\Sigma_{\mathrm{cma}}$ (since we can extract the witnesses and thus a forged signature). We start by instantiating the mentioned building blocks.

**Building Blocks.** The main motivation for our choices of instantiations of these blocks is that their security is implied by DLin only. We insist that by admitting more exotic assumptions, the efficiency of our proof system could be improved.

The Strong One-Time Signature Scheme $\Sigma_{\mathrm{ot}}$. We pick the scheme described in [12] (but any other would equally do), since its security follows from the discrete-log assumption which is implied by DLin.

The Waters Signature Scheme. The signature scheme from [16] suits our purposes, it requires no additional assumption and—more importantly—signatures are verified by checking PPEs.

*Setup.* In a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, define parameters $f \leftarrow \mathbb{G}^*$ and $\boldsymbol{h} := (h_0, h_1, \ldots, h_\ell) \leftarrow \mathbb{G}^{\ell+1}$. A secret key $x \leftarrow \mathbb{Z}_p$ defines a public key $X := g^x$. For ease of notation, define $\mathcal{W}(M) := h_0 \prod_{i=1}^{\ell} h_i^{M_i}$.

*Signing.* To sign a message $M \in \{0,1\}^\ell$, choose $r \leftarrow \mathbb{Z}_p$ and define a signature as $\sigma := (f^x \mathcal{W}(M)^r,\ g^{-r})$.

*Verification.* A signature $\sigma = (\sigma_1, \sigma_2)$ is accepted for message $M$ iff

$$e(\sigma_1, g)\, e(\mathcal{W}(M), \sigma_2) = e(f, X) \tag{1}$$

*Security.* EUF-CMA follows from the computational Diffie-Hellman assumption which is implied by DLin.

The Groth-Sahai Proof System. Consider a set of *pairing product equations* $\{E_k\}_{k=1}^{K_E}$ on variables $\{X_i\}_{i=1}^n$ in $\mathbb{G}$ of the form

$$\prod_{i=1}^{n} e(A_{k,i}, X_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(X_i, X_j)^{\gamma_{k,i,j}} = T_k \tag{$E_k$}$$

for given $A_{k,i} \in \mathbb{G}$, $\gamma_{k,i,j} \in \mathbb{Z}_p$, and $T_k \in \mathbb{G}_T$. Groth and Sahai [14] build a non-interactive witness-indistinguishable proof of satisfiability of $\{E_k\}$ from which—given a trapdoor—can be extracted the witnesses $X_i$ (we will use their instantiation with DLin): the CRS is a (binding) key for linear commitments to group elements. The proof consists of commitments to each $X_i$ and 9 elements of $\mathbb{G}$ per equation proving that it is satisfied by the committed values. By DLin, replacing the CRS by a *hiding* commitment key is indistinguishable. In this setting now every witness $\{X_i\}_{i=1}^n$ satisfying the equations generates the same distribution of proofs, which implies witness-indistinguishability of the proofs.

Moreover, we assume a collision-resistant hash function $\mathcal{H}$ that maps strings of elements of $\mathbb{G}$ to elements in $\mathbb{Z}_p$ which we identify with their bit-representation in $\{0,1\}^{\lceil \log p \rceil}$. Thus, when we say we sign a vector of group elements, we actually mean that we sign their hash values.

**Equations for Proof of Plaintexts Being in CDH.** Let $c, d \in \mathbb{G}$ be fixed and let $(g_1, g_2, g_3)$ be a linear encryption key. Given two ciphertexts $C = (g_1^r, g_2^s, m_1 g_3^{r+s})$ and $D = (g_1^t, g_2^u, m_2 g_3^{t+u})$, we give a set of PPEs that are satisfiable by a witness $a$ if and only if there exists $\mu \in \mathbb{Z}_p$ such that $m_1 = c^\mu$ and $m_2 = d^\mu$.

$$\begin{aligned} &e(C_1, g_3) = e(g_1, a_1) \quad e(C_2, g_3) = e(g_2, a_2) \\ &e(D_1, g_3) = e(g_1, a_3) \quad e(D_2, g_3) = e(g_2, a_4) \quad e(C_3 a_1^{-1} a_2^{-1}, d) = e(c, D_3 a_3^{-1} a_4^{-1}) \end{aligned} \tag{2}$$

The witness satisfying them is $a := (g_3^r, g_3^s, g_3^t, g_3^u)$. The first four equations prove that the logarithms of the $a_i$'s are those of $C_1, C_2, D_1, D_2$ w.r.t. their respective bases. Thus, $C_3 a_1^{-1} a_2^{-1} = m_1$ and $D_3 a_3^{-1} a_4^{-1} = m_2$ and the last equation shows that $(m_1, m_2)$ is in CDH w.r.t. $(c, d)$.

**Disjunction of Equations.** Following [12] (and optimizing since the pairings have variables in common), we define a set of equations which we can prove satisfiable if we have witnesses for either (2) or (1), i.e., if we either know $a$ satisfying (2) or $\sigma$ satisfying (1). We first introduce the following new variables:

$$\chi_1, \chi_2 \qquad \phi_1, \phi_2, \phi_3, \phi_4, \phi_5 \qquad \psi_1, \psi_2, \psi_3$$

We define the following 15 equations expressing a disjunction of (2) and (1), therefore termed "(2)$\vee$(1)".

Equation for Disjunction: $e(g^{-1}\chi_1\chi_2, g) = 1$

From (1): $e(\chi_2, \psi_1^{-1}\sigma_1) = 1 \quad e(\chi_2, \psi_2^{-1}\mathcal{W}(M)) = 1 \quad e(\chi_2, \psi_3^{-1} f) = 1$

$e(\psi_1, g)\, e(\psi_2, \sigma_2)\, e(\psi_3, X)^{-1} = 1$

From (2): $e(\chi_1, \phi_1^{-1} g_1) = 1 \quad e(\chi_1, \phi_2^{-1} g_2) = 1$

$e(\chi_1, \phi_3^{-1} g_3) = 1 \quad e(\chi_1, \phi_4^{-1} c) = 1 \quad e(\chi_1, \phi_5^{-1} d) = 1$

$e(C_1, \phi_3)\, e(\phi_1, a_1)^{-1} = 1 \quad e(C_2, \phi_3)\, e(\phi_2, a_2) = 1$

$e(D_1, \phi_3)\, e(\phi_1, a_3)^{-1} = 1 \quad e(D_2, \phi_3)\, e(\phi_2, a_4) = 1$

$e(C_3 a_1^{-1} a_2^{-1}, \phi_5)\, e(\phi_4, D_3 a_3^{-1} a_4^{-1}) = 1$

Completeness. To produce a proof we proceed as follows: If we have an assignment $a$ for (2), we choose $\chi_1 := g$, $\chi_2 := 1$, satisfying thus the first equation. Moreover, set $\phi_1 := g_1$, $\phi_2 := g_2$, $\phi_3 := g_3$, $\phi_4 := c$, $\phi_5 := d$. Thus the equations of the block for (2) are satisfied, because $a$ is a witness for (2). Since $\chi_2 = 1$, we can set $\psi_i := 1$ (for all $i$) as well, which satisfies the block for (1), no matter what value we set $\sigma$.

On the other hand, if we know a signature $\sigma$ satisfying (1), we choose $\chi_1 := \phi_i := 1$ (for all $i$) and $\chi_2 := g$, $\psi_1 := \sigma_1$, $\psi_2 := \mathcal{W}(M)$, $\psi_3 := f$ and get a satisfying assignment for any choice of $a$.

Soundness. We show that if (2∨1) is satisfied then either $a$ satisfies (2) or $\sigma$ satisfies (1): From the first equation we have that either $\chi_1$ or $\chi_2$ must be non-trivial, which either confines the values of the $\phi_i$'s to $(g_1, g_2, g_3, c, d)$ or those of the $\psi_i$'s to $(\sigma_1, \mathcal{W}(M), f)$. Now this imposes that either $a$ satisfies (2) (by the last five equations of the block for (2)) or $\sigma$ satisfies (1) (by the last equation of the block for (1)).

**Equations for Proof of Commitment and Ciphertext Containing the Same Value.** Let $(g_1, g_2, g_3)$ be a key for linear encryption, and let $(U_1, U_2, g)$ be an associated commitment key. Let $C = (U_1^x g_1^r, U_2^x g_2^s, g^x g_3^{r+s})$ be a commitment to $x$ and $D = (g_1^v, g_2^w, g^y g_3^{v+w})$ be an encryption of $g^y$. We prove that $x = y$: the witness is $(a_1 = U_1^x, a_2 = U_2^x, a_3 = g^x, a_4 = g_3^r, a_5 = g_3^v)$ satisfying

$$\begin{array}{lll} e(a_1, U_2) = e(U_1, a_2) & e(C_1 a_1^{-1}, g_3) = e(g_1, a_4) & e(D_1, g_3) = e(g_1, a_5) \\ e(a_1, g) = e(U_1, a_3) & e(C_2 a_2^{-1}, g_3) = e(g_2, C_3 a_3^{-1} a_4^{-1}) & e(D_2, g_3) = e(g_2, D_3 a_3^{-1} a_5^{-1}) \end{array} \quad (3)$$

The equations in the first column show that $a_1 = U_1^z$, $a_2 = U_2^z$, $a_3 = g^z$ for some $z$, the second column proves that $(C_1 a_1^{-1}, C_2 a_2^{-1}, C_3 a_2^{-1})$ is linear (i.e., $C$ commits to $z$) and the third that $D$ is an encryption of $a_3 = g^z$.

Transformation. Transforming Equations (3) and (1) to a set (3∨1) analogously to the construction of (2∨1), we get a set of 16 equations we can prove satisfiable adding 10 new witnesses if either we have a witness for $C$ being a commitment to some $x$ and $D$ an encryption of $g^x$, or we know a signature. (Associate the $\phi_i$'s to $U_1$, $a_1$, $g_1$, $g_2$ and $g_3$.)

**Assembling the Pieces.** We describe the SSNIZK proof system for "plaintexts in CDH". The one for "commitment and ciphertext contain the same value" is obtained by replacing (2∨1) by (3∨1).

Common Reference String. Generate a key pair $(\mathsf{vk}, \mathsf{sk})$ for Waters' signature scheme, and a CRS $\mathsf{crs}_{\mathrm{GS}}$ for the Groth-Sahai proof system. Let $\mathsf{crs} := (\mathsf{vk}, \mathsf{crs}_{\mathrm{GS}})$ and let the simulation trapdoor be $\mathsf{sk}$.

Proof. Let $(C, D) \in \mathbb{G}^6$ be an instance and $a$ a witness satisfying (2). Generate a key pair $(\mathsf{vk}_{\mathrm{ot}}, \mathsf{sk}_{\mathrm{ot}})$ for $\Sigma_{\mathrm{ot}}$; using witness $a$, make a Groth-Sahai proof $\pi_{\mathrm{GS}}$ w.r.t. $\mathsf{crs}_{\mathrm{GS}}$ of satisfiability of (2∨1) with $M := \mathsf{vk}_{\mathrm{ot}}$; produce a signature $\sigma_{\mathrm{ot}}$ on $(C, D, \mathsf{vk}_{\mathrm{ot}}, \pi_{\mathrm{GS}})$ using $\mathsf{sk}_{\mathrm{ot}}$. The proof is $\pi := (\mathsf{vk}_{\mathrm{ot}}, \pi_{\mathrm{GS}}, \sigma_{\mathrm{ot}})$

Verification. Given $\pi$, verify $\sigma_{\mathrm{ot}}$ on $(C, D, \mathsf{vk}_{\mathrm{ot}}, \pi_{\mathrm{GS}})$ under $\mathsf{vk}_{\mathrm{ot}}$, and $\pi_{\mathrm{GS}}$ on the respective equations.

Simulation. Proceed as in Proof, but using $\mathsf{sk}$ produce $\sigma$ on $\mathsf{vk}_{\mathrm{ot}}$ and use that as a witness for (2∨1).

**Theorem 3.** *Under the DLin assumption, the above is a simulation-sound NIZK argument for the encryptions of two linear ciphertexts forming a CDH-pair.*

Using the ideas given in the overview, the proof is analogous to that in [12] except that we do not require perfect soundness and that we use the extraction key for $\mathsf{crs}_{\mathrm{GS}}$ to extract a forged signature on $\mathsf{vk}_{\mathrm{ot}}$ directly rather than adding encryptions to the proof.

## Acknowledgments

This work was supported in part by the European Commission through the ICT Program under Contract ICT-2007-216646 ECRYPT II, by the French ANR-07-SESU-008-01 PAMPA Project, and EADS.

## References

1. Abdalla, M., Boyen, X., Chevalier, C., Pointcheval, D.: Distributed public-key cryptography from weak secrets. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 139–159. Springer, Heidelberg (2009)
2. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 361–377. Springer, Heidelberg (2005)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993: 1st Conference on Computer and Communications Security, pp. 62–73. ACM Press, New York (1993)
4. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
5. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
6. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
8. Boyen, X., Chevalier, C., Fuchsbauer, G., Pointcheval, D.: Strong cryptography from weak secrets: Building efficient PKE and IBE from distributed passwords. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 297–315. Springer, Heidelberg (2010); Full version available from the web page of the authors
9. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, pp. 136–145. IEEE Computer Society Press, Los Alamitos (2001)
10. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
11. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
12. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006)
13. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)

14. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
15. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: ACM CCS 2005: 12th Conference on Computer and Communications Security, pp. 180–189. ACM Press, New York (2005)
16. Waters, B.R.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

# Efficient Unidirectional Proxy Re-Encryption*

Sherman S.M. Chow[1], Jian Weng[2,3,4],
Yanjiang Yang[5], and Robert H. Deng[3]

[1] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY, USA
schow@cs.nyu.edu
[2] Department of Computer Science, Jinan University, Guangzhou, China
[3] School of Information Systems, Singapore Management University, Singapore
cryptjweng@gmail.com, robertdeng@smu.edu.sg
[4] State Key Laboratory of Information Security
Institute of Software, Chinese Academy of Sciences, Beijing, China
[5] Institute for Infocomm Research, Singapore
yyang@i2r.a-star.edu.sg

**Abstract.** Proxy re-encryption (PRE) allows a semi-trusted proxy to convert a ciphertext originally intended for Alice into one encrypting the same plaintext for Bob. The proxy only needs a re-encryption key given by Alice, and cannot learn anything about the plaintext encrypted. This adds flexibility in various applications, such as confidential email, digital right management and distributed storage. In this paper, we study *unidirectional* PRE, which the re-encryption key only enables delegation in one direction but not the opposite. In PKC 2009, Shao and Cao proposed a unidirectional PRE assuming the random oracle. However, we show that it is vulnerable to chosen-ciphertext attack (CCA). We then propose an efficient unidirectional PRE scheme (without resorting to pairings). We gain high efficiency and CCA-security using the "token-controlled encryption" technique, under the computational Diffie-Hellman assumption, in the random oracle model and a relaxed but reasonable definition.

**Keywords:** proxy re-encryption, unidirection, chosen-ciphertext attack.

## 1 Introduction

Every application which requires some sort of confidentiality uses encryption as a building block. As pointed out by Mambo and Okamoto [1], the encrypted data often needs to be re-distributed in practice, i.e., the data encrypted under a public key $\mathtt{pk}_i$ should also be encrypted under another independently generated public key $\mathtt{pk}_j$. This can be easily done if the holder of the secret key $\mathtt{sk}_i$ (corresponding to $\mathtt{pk}_i$) is *online* – simply decrypts the ciphertext and re-encrypts

* This work is partially supported by the Office of Research, Singapore Management University. It is also partially supported by the National Science Foundation of China under Grant No. 60903178. We thank Jun Shao for a discussion of the attack.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 316–332, 2010.

the plaintext to $\mathtt{pk}_j$. However, this is not always practical. It is also undesirable to just disclose the secret key to some untrusted server to do the transformation of ciphertexts.

To solve this key management problem which hinders the practical adoption of encryption, Blaze, Bleumer and Strauss [2] introduced the concept of proxy re-encryption (PRE). PRE schemes allow a secret key holder to create a re-encryption key. A semi-trusted proxy can use this key to translate a message $m$ encrypted under the delegator's public key into an encryption of the same message under a delegatee's public key, as specified by the delegator. This can be done without allowing the proxy any ability to perform tasks outside of these proxy delegations. In particular, the proxy can neither recover the delegator's secret key nor decrypt the delegator's ciphertext.

Proxy re-encryption schemes have applications in digital rights management (DRM) [3], distributed file storage systems [4], law enforcement [5], encrypted email forwarding [2], and outsourced filtering of encrypted spam [4]. In all these cases, the gist is that the process of re-encryption, i.e., decrypting under one key for encryption under another key, should not allow the re-encryptor module to compromise the secrecy of encrypted messages. This was related to the compromise of Apple's iTunes DRM [3]. With a PRE scheme, the problem is solved since re-encryption can be performed without awarding the proxy any information about the encrypted message. Besides DRM, distributed file storage systems also benefit in the sense that the storage server (proxy) can re-encrypt the files for different servers without knowing the underlying file content, and hence it is less attractive for hacker attacks since compromising the server does not compromise the files. Similarly, email servers can re-encrypt emails for different users with the same effect, say when a user is on vacation and wants to forward his encrypted emails to his colleague.

### 1.1 The Use of Pairings in Proxy Re-Encryption

Blaze, Bleumer and Strauss's seminal work [2] proposed a bidirectional PRE scheme against chosen plaintext attack (CPA). Afterwards, a number of PRE schemes have been proposed. Their properties are summarized in Table 1. The schemes are chronologically arranged.

In this paper, we study unidirectional public-key-based PRE schemes which are secure against adaptive chosen-ciphertext attack (CCA). As shown in Table 1, most existing PRE schemes most existing PRE schemes no matter ID-based or not, are realized by pairings. Below we look into two schemes to see why pairing is a useful "ingredient". In the bidirectional scheme proposed by Canetti and Hohenberger [7], the transformation key is simply $\mathtt{rk}_{i \leftrightarrow j} = x_j/x_i \in \mathbb{Z}_p$ for the pair of delegation partners[1] $\mathtt{pk}_i = g^{x_i}$ and $\mathtt{pk}_j = g^{x_j}$. The ciphertext comes with the term $\mathtt{pk}_i^r$ for randomness $r \in \mathbb{Z}_p$ which can be transformed to $\mathtt{pk}_j^r$ easily by using $\mathtt{rk}_{i \leftrightarrow j}$. The ciphertext validity can be checked with the help

[1] For the bidirectional schemes, once a delegation is made, a delegator becomes a delegatee and a delegate becomes a delegator simultaneously.

**Table 1.** Summary of PRE Schemes

| Schemes | Uni/Bi Directional | Security | RO -Free | Pairing -Free | Collusion -Resistant |
|---|---|---|---|---|---|
| Public-key-based | | | | | |
| Ateniese *et al.* [4] | → | CPA | × | × | ✓ |
| Hohenberger *et al.* [6] | → | CPA | ✓ | × | ✓ |
| Canetti-Hohenberger [7] | ↔ | CCA | ✓ | × | × |
| Libert-Vergnaud [8] | → | RCCA | ✓ | × | ✓ |
| Libert-Vergnaud-Trace [9] | → | CPA | ✓ | × | ✓ |
| Deng *et al.* [10] | ↔ | CCA | × | ✓ | × |
| Shao-Cao [11] | → | CCA? | × | ✓ | × |
| Ateniese *et al.* [12] | → | CPA | ✓ | × | ✓ |
| Ours | → | CCA | × | ✓ | ✓ |
| Identity-based | | | | | |
| Green-Ateniese [13] | → | CCA | × | × | × |
| Chu-Tzeng [14] | → | RCCA | ✓ | × | × |

of the pairing function $\hat{e}(\cdot,\cdot)$ with respect to the generator $g$ and the public key $\mathtt{pk}_i$ or $\mathtt{pk}_j$. For the unidirectional PRE scheme proposed by Libert and Vergnaud [8] (hereinafter referred as $\mathcal{LV08}$), the transformation key is in the form $\mathtt{rk}_{i\leftrightarrow j} = g^{x_j/x_i}$. The ciphertext also comes with the term $\mathtt{pk}_i^r$ and the message is encrypted by $\hat{e}(g,g)^r$. To recover the message, a pairing will be applied to get $\hat{e}(g^{x_j/x_i}, \mathtt{pk}_i^r) = \hat{e}(g,g^r)^{x_j}$, $\hat{e}(g,g)^r$ can then be covered with $x_j$. These techniques for unidirectional transformation and ciphertext validity checking intrinsically require the pairings. Moreover, the security guarantee provided by $\mathcal{LV08}$ is only against replayable chosen-ciphertext attacks (RCCA) [15], a weaker variant of CCA tolerating a "harmless mauling" of the challenge ciphertext.

### 1.2 Our Contributions

From a theoretical perspective, we would like to have PRE scheme realized under a broader class of complexity assumptions, and see techniques other than using pairing in constructing CCA-secure PRE. Practically, we want a PRE scheme with simple design, short ciphertext size and high computational efficiency[2]. Removing pairing from PRE constructions is one of the open problems left by [7].

Recently, Shao and Cao [11] proposed a unidirectional PRE scheme without pairings (referred as $\mathcal{SC09}$). Let $N$ be a safe-prime modulus. $\mathcal{SC09}$ requires 4 to 5 exponentiations in $\mathbb{Z}^*_{N^2}$ for encryption, re-encryption and decryption[3], and incurs an ciphertext overhead of 3 (plus proof-of-knowledge) to 5 $\mathbb{Z}^*_{N^2}$ elements. The modulus being used is $N^2$. Its performance over pairing-based scheme

[2] In spite of the recent advances in implementation technique, compared with modular exponentiation, pairing is still considered as a rather expensive operation, especially in computational resource limited settings.

[3] Speed-up by Chinese remainder theorem is not possible except 2 exponentiations in decryption, due to the lack of the factoring of the delegator's modulus.

(e.g., *LV08*), which is instantiated on elliptic curves consist of much shorter group elements at the same security level, is questionable. Their security proof relies on the random oracle and the decisional (not computational) Diffie-Hellman assumption over $\mathbb{Z}_{N^2}^*$.

Most importantly, we identify flaws in their security proof which translate to a real-world chosen-ciphertext attack against *SC09*. A possible fix further degrades the performance in decryption time. In view of this, we propose an efficient unidirectional CCA-secure PRE scheme without pairings, under the *standard* computational Diffie-Hellman assumption, in the random oracle model. Our design is based on ElGamal encryption [16] and Schnorr signature [17], which is (arguably) simple. Our decryption process is more natural and does not require the input of the delegator's public key, which is required in *SC09*.

In this paper, collusion attack refers to any collusion of a proxy and a delegatee which aimed to comproise the security of the delegator in *any* meaningful way.[4] Finally, to the best of our knowledge, there was no (R)CCA-secure unidirectional scheme which is collusion-resistant.

## 2 Our Definitions of Unidirectional Proxy Re-Encryption

### 2.1 Framework of Unidirectional Proxy Re-Encryption

A unidirectional PRE scheme consists of the following six algorithms [7]:

$\mathsf{Setup}(\kappa)$**:** The setup algorithm takes as input a security parameter $\kappa$ and outputs the global parameters $\mathtt{param}$, which include a description of the message space $\mathcal{M}$.

$\mathsf{KeyGen}()$**:** The key generation algorithm generates a public/private key pair $(\mathtt{pk}_i, \mathtt{sk}_i)$.

$\mathsf{ReKeyGen}(\mathtt{sk}_i, \mathtt{pk}_j)$**:** The re-encryption key generation algorithm takes as input a private key $\mathtt{sk}_i$ and another public key $\mathtt{pk}_j$. It outputs a re-encryption key $\mathtt{rk}_{i\to j}$.

$\mathsf{Encrypt}(\mathtt{pk}, m)$**:** The encryption algorithm takes as input a public key $\mathtt{pk}$ and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathfrak{C}$ under $\mathtt{pk}$.

$\mathsf{ReEncrypt}(\mathtt{rk}_{i\to j}, \mathfrak{C}_i)$**:** The re-encryption algorithm takes as input a re-encryption key $\mathtt{rk}_{i\to j}$ and a ciphertext $\mathfrak{C}_i$ under public key $\mathtt{pk}_i$. It outputs a ciphertext $\mathfrak{C}_j$ under public key $\mathtt{pk}_j$. This can be either deterministic or probabilistic.

$\mathsf{Decrypt}(\mathtt{sk}, \mathfrak{C})$**:** The decryption algorithm takes as input a private key $\mathtt{sk}$ and a ciphertext $\mathfrak{C}$. It outputs a message $m \in \mathcal{M}$ or the error symbol $\perp$ if the ciphertext is invalid.

[4] For example, the collusion-resistance claimed in [11] can be more accurately described as delegator-secret-key security (also see Section 2.2), and we listed it as *not* collusion-resistant due to the following attack. A collusion of a delegatee of $X$ and his proxy can recover a weak secret key ($\mathtt{wsk}_X$) of $X$. Any re-enryption of ciphertext of $X$ to *other* delegatee contains most part of the original one, in particular, it is decryptable by applying $\mathtt{wsk}_X$ on the original components (also see Section 3.)

To lighten notations, we omit the public parameters $\mathtt{param}$ as the input of the algorithms. Correctness requires that, for any parameters $\mathtt{param}$, $m \in \mathcal{M}$, the following probabilities are equal to 1:

$$\Pr\left[\mathsf{Decrypt}(\mathtt{sk}_i, \mathfrak{C}) = m \,\middle|\, (\mathtt{sk}_i, \mathtt{pk}_i) \leftarrow \mathsf{KeyGen}(), \mathfrak{C} \leftarrow \mathsf{Encrypt}(\mathtt{pk}_i, m)\right],$$

$$\Pr\left[\mathsf{Decrypt}\left(\mathtt{sk}_j, \mathfrak{C}_j\right) = m \,\middle|\, \begin{array}{c} (\mathtt{sk}_i, \mathtt{pk}_i) \leftarrow \mathsf{KeyGen}(), \\ (\mathtt{sk}_j, \mathtt{pk}_j) \leftarrow \mathsf{KeyGen}(), \\ \mathtt{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathtt{sk}_i, \mathtt{pk}_j), \\ \mathfrak{C}_i \leftarrow \mathsf{Encrypt}(\mathtt{pk}_i, m), \\ \mathfrak{C}_j \leftarrow \mathsf{ReEncrypt}(\mathtt{rk}_{i\to j}, \mathfrak{C}_i) \end{array}\right]$$

### 2.2 Security Models for "Token-Controlled" Re-Encryption

Our game-based definitions for single-hop unidirectional PRE systems are adaptions of the definitions of the original (second level) ciphertext security and the transformed (first level) ciphertext security in [8]. As in [7,8] our static corruption model makes the knowledge of secret key (KOSK) assumption, the adversary only gets uncorrupted public key or corrupted public/private key pair from the challenger, and is not allowed to adaptively determine which parties will be compromised. Compared with [7,8], our definition considers the standard CCA security instead of RCCA security. However, this is at the expense of a relaxation requiring additional constraint on the re-encryption key that can be compromised.

**Definition 1 (Game Template of Chosen-Ciphertext Security).**

**Setup.** *The challenger $\mathcal{C}$ takes a security parameter $\kappa$ and executes the setup algorithm to get the system parameters* $\mathtt{param}$. *$\mathcal{C}$ executes the key generation algorithm $n_u$ times resulting a list of public/private keys $\mathcal{PK}_{good}, \mathcal{SK}_{good}$, and executes the key generation algorithm for $n_c$ times to get a list of corrupted public/private keys $\mathcal{PK}_{corr}, \mathcal{SK}_{corr}$. $\mathcal{A}$ gets* $\mathtt{param}$, *$\mathcal{SK}_{corr}$, and $\mathcal{PK} = (\mathcal{PK}_{good} \cup \mathcal{PK}_{corr}) = \{\mathtt{pk}_i\}_{i\in[1,n_u+n_c]}$.*

**Phase 1.** *$\mathcal{A}$ adaptively queries to oracles* OReK, OReE *and* ODec.

- OReK *oracle takes $\langle \mathtt{pk}_i, \mathtt{pk}_j \rangle$ and returns a re-encryption key $\mathtt{rk}_{i\to j}$.*
- OReE *oracle takes public keys $\langle \mathtt{pk}_i, \mathtt{pk}_j \rangle$ and a ciphertext $\mathfrak{C}$ and returns a re-encryption of $\mathfrak{C}$ from $\mathtt{pk}_i$ to $\mathtt{pk}_j$.*
- ODec *oracle takes a public key* $\mathtt{pk}$ *and a ciphertext $\mathfrak{C}$ and returns the decryption of $\mathfrak{C}$ using the private key with respect to* $\mathtt{pk}$.

**Challenge.** *When $\mathcal{A}$ decides that Phase 1 is over, it also decides whether it wants to be challenged with a original ciphertext or a transformed ciphertext. It outputs two equal-length plaintexts $m_0, m_1 \in \mathcal{M}$, and a target public key $\mathtt{pk}_{i^*}$. Challenger $\mathcal{C}$ flips a random coin $\delta \in \{0,1\}$, and sends to $\mathcal{A}$ a challenge ciphertext $\mathfrak{C}^*$ depending on $\mathtt{pk}_{i^*}$ and $m_\delta$*

**Phase 2.** *$\mathcal{A}$ issues queries as in Phase 1.*

**Guess.** *Finally, $\mathcal{A}$ outputs a guess $\delta' \in \{0,1\}$.*

*The public keys supplied by $\mathcal{A}$ subject to the following constraints:*

1. *The public keys involved in all queries must come from $\mathcal{PK}$.*
2. *The target public key $\mathtt{pk}_{i^*}$ is from $\mathcal{PK}_{good}$, i.e., uncorrupted.*

The actual construction of $\mathfrak{C}^*$ and the constraints on the queries made by $\mathcal{A}$ are to be defined according to different security notions.

**Definition 2 (Original Ciphertext Security).** *For original ciphertext security, the adversary $\mathcal{A}$ plays the CCA game with the challenger $\mathcal{C}$ as in Definition 1, where the challenge ciphertext is formed by $\mathfrak{C}^* = \mathsf{Encrypt}(\mathtt{pk}_{i^*}, m_\delta)$, and $\mathcal{A}$ has the following additional constraints:*

1. *$\mathsf{ORek}(\mathtt{pk}_{i^*}, \mathtt{pk}_j)$ is only allowed if $\mathtt{pk}_j$ came from $\mathcal{PK}_{good}$.*
2. *If $\mathcal{A}$ issued $\mathsf{ORe E}(\mathtt{pk}_i, \mathtt{pk}_j, \mathfrak{C}_i)$ where $\mathtt{pk}_j$ came from $\mathcal{PK}_{corr}$, $(\mathtt{pk}_i, \mathfrak{C}_i)$ cannot be a derivative of $(\mathtt{pk}_{i^*}, \mathfrak{C}^*)$ (to be defined later).*
3. *$\mathsf{ODec}(\mathtt{pk}, \mathfrak{C})$ is only allowed if $(\mathtt{pk}, \mathfrak{C})$ is not a derivative of $(\mathtt{pk}_{i^*}, \mathfrak{C}^*)$.*

**Definition 3 (Derivative for Chosen-Ciphertext Security).** *Derivative of $(\mathtt{pk}_{i^*}, \mathfrak{C}^*)$ in the CCA setting is inductively defined in [11] as below, which is adopted from the RCCA-based definition in [7][5]:*

1. *Reflexivity: $(\mathtt{pk}_{i^*}, \mathfrak{C}^*)$ is a derivative of itself.*
2. *Derivation by re-encryption: If $\mathcal{A}$ has issued a re-encryption query $\langle \mathtt{pk}, \mathtt{pk}', \mathfrak{C} \rangle$ and obtained the resulting re-encryption ciphertext $\mathfrak{C}'$, then $(\mathtt{pk}', \mathfrak{C}')$ is a derivative of $(\mathtt{pk}, \mathfrak{C})$.*
3. *Derivation by re-encryption key: If $\mathcal{A}$ has issued a re-encryption key generation query $\langle \mathtt{pk}, \mathtt{pk}' \rangle$ to obtain the re-encryption key $\mathtt{rk}$, and $\mathfrak{C}' = \mathsf{ReEncrypt}(\mathtt{rk}, \mathfrak{C})$, then $(\mathtt{pk}', \mathfrak{C}')$ is a derivative of $(\mathtt{pk}, \mathfrak{C})$.*

**Definition 4 (Transformed Ciphertext Security).** *For transformed ciphertext, the adversary $\mathcal{A}$ plays the CCA game with the challenger $\mathcal{C}$ as in Definition 1, where $\mathcal{A}$ can also specify the delegator $\mathtt{pk}_{i'}$. The challenge ciphertext is then created by the re-encryption process, specifically, $\mathfrak{C}^* = \mathsf{ReEncrypt}(\mathtt{rk}_{i' \to i^*}, \mathsf{Encrypt}(\mathtt{pk}_{i'}, m_\delta))$. The only constraints of $\mathcal{A}$ are:*

1. *$\mathsf{ODec}(\mathtt{pk}_{i^*}, \mathfrak{C}^*)$ is not allowed.*
2. *If $\mathtt{pk}_{i'}$ came from $\mathcal{PK}_{corr}$, $\mathcal{C}$ would not return $\mathtt{rk}_{i' \to i^*}$ to $\mathcal{A}$ in phase 2.*
3. *If $\mathcal{A}$ obtained $\mathtt{rk}_{i' \to i^*}$, $\mathcal{A}$ cannot choose $\mathtt{pk}_{i'}$ as the delegator in the challenge phase.*

This can be considered as a weaker notion when compared with [8].

**Definition 5 (CCA Security of a PRE).** *We define $\mathcal{A}$'s advantage in attacking the PRE scheme as $Adv^{IND\text{-}PRE\text{-}CCA}_{PRE,\mathcal{A}} = \left| \Pr[\delta' = \delta] - 1/2 \right|$, where the probability is taken over the random coins consumed by the challenger and the adversary.*

[5] These original definitions also consider transitivity – If $(\mathtt{pk}, \mathfrak{C})$ is a derivative of $(\mathtt{pk}_{i^*}, \mathfrak{C}^*)$ and $(\mathtt{pk}', \mathfrak{C}')$ is a derivative of $(\mathtt{pk}, \mathfrak{C})$, then $(\mathtt{pk}', \mathfrak{C}')$ is a derivative of $(\mathtt{pk}_{i^*}, \mathfrak{C}^*)$. However, this is irrelevant for single-hop scheme like ours and [11].

*A single-hop unidirectional PRE scheme is defined to be* $(t, n_u, n_c, q_{rk}, q_{re}, q_d, \epsilon)$*-IND-PRE-CCA secure, if for any* $t$*-time IND-PRE-CCA adversary* $\mathcal{A}$ *who makes at most* $q_{rk}$ *re-encryption key generation queries, at most* $q_{re}$ *re-encryption queries and at most* $q_d$ *decryption queries, we have* $\mathrm{Adv}^{\mathrm{IND-PRE-CCA}}_{\mathrm{PRE},\mathcal{A}} \leq \epsilon$.

**Derivative and Two Different Kinds of Security.** Intuitively speaking, original ciphertext security models the an adversary $\mathcal{A}$ challenged with an untransformed ciphertext encrypted for a target user $i^*$. In a PRE scheme, however, $\mathcal{A}$ can ask for the re-encryption of many ciphertexts or even a set of re-encryption keys. These queries are allowed as long as they would not allow $\mathcal{A}$ to decrypt trivially. For examples, $\mathcal{A}$ *should not* get the re-encryption key from user $i^*$ to user $j$ if the secret key of user $j$ has been compromised; on the other hand, $\mathcal{A}$ can certainly get a re-encryption of the challenge ciphertext from user $i^*$ to user $j$ as long as $j$ is an honest user and the decryption oracle of user $j$ has not been queried with the resulting transformed ciphertext. This explains the intuition behind the notion of derivative and the associated restrictions.

Since $\mathcal{A}$ can derive a transformed ciphertext with a certain related re-encryption key, one may wonder why there is another notion about transformed ciphertext security. This latter notion makes sense when the PRE system is *single-hop*, i.e., a transformed ciphertext cannot be re-encrypted further to someone else. If a proxy colludes with a delegatee, by the correct functionalities of a PRE, this collusion group can certainly decrypt any *original* ciphertext of the target user. However, for a single-hop scheme, there is no reason that this collusion group can decrypt any *transformed* ciphertext since it cannot be re-encrypted further. To conclude, the adversary is allowed to transform an original ciphertext in the former notion, but there are some re-encryption keys which it is not allowed to get (recall the constraints related to derivatives); while in the latter, the adversary only sees the transformed ciphertext but not the original one, and the adversary can get more re-encryption keys.

**Our Definition of Transformed Ciphertext Security.** The second constraint deserves more discussion. The compromise of $\mathtt{rk}_{i' \to i^*}$ corresponds to the fact that the proxy, which is designated by the delegator $\mathtt{pk}_{i'}$ for the delegation to the delegatee $\mathtt{pk}_{i^*}$, is compromised. Ideally, it seems that whether the delegator $\mathtt{pk}_{i'}$ is compromised or not in this situation does not affect the security of the transformed ciphertext for $\mathtt{pk}_{i^*}$. This is also what has been modelled by the definition in [8]. However, if the adversary $\mathcal{A}$ compromised the delegator $\mathtt{pk}_{i'}$ and also the proxy, $\mathcal{A}$ can simply ask the proxy to surrender the original ciphertext $\mathsf{Encrypt}(\mathtt{pk}_{i'}, m_\delta)$ before any actual transformation, and use $\mathtt{sk}_{i'}$ to decrypt trivially. It is true that if the proxy was initially honest and erased the original ciphertexts after their transformation, the same attack does not apply; however, ciphertext is by definition public in nature and the adversary may have captured the ciphertext already and decrypt it when $\mathtt{sk}_{i'}$ is obtained. We believe that the relaxed notion still have significance in the real world.

**Nontransformable (First-Level) Ciphertext.** To view the above relaxation from another angle, one may feel that we lost a possible benefit of a single-hop scheme – some ciphertexts are not further transformable so very sensitive information can be encrypted in this form ("first level" ciphertext that cannot be re-encrypted). Actually, our definition does not rule out this possibility. Our definition given above only considers *transformed* ciphertext, that is, the challenge ciphertext which is generated from the re-encryption algorithm. It does not rule out the possibility of having another encryption algorithm $\mathsf{Encrypt}_1$ which directly produces nontransformable ciphertext, when $\mathsf{ReEncrypt}(\mathtt{rk}_{i'\to i^*}, \mathsf{Encrypt}(\mathtt{pk}_{i'}, m_\delta))$ and $\mathsf{Encrypt}_1(\mathtt{pk}_{i^*}, m_\delta)$ are actually distinguishable.

We view this as one way to get CCA security instead of RCCA security. Using *LV08*, it is possible to directly encrypt ciphertexts that cannot be re-encrypted which is indistinguishable from re-encryption, and the reason is that re-randomization can be done in the re-encryption process. Recall that the security guarantee of *LV08* actually allows the adversary to compromise all proxies of the system; indeed, the re-randomizaation in *LV08* can be done by any one without any secret knowledge – this explains why *LV08* is at most RCCA secure.

Of course, it is required to augment the PRE systems with yet another encryption algorithm. However, it is often the case that the original decryption algorithm sufficies to decrypt ciphertext produced in this way. The interface of $\mathsf{Encrypt}_1$ and its correctness requirement are exactly the same as those of $\mathsf{Encrypt}$. The security definition is also simple.

**Definition 6 (Nontransformable Ciphertext Security).** *For nontransformable ciphertext, the adversary $\mathcal{A}$ plays the CCA game with the challenger $\mathcal{C}$ as in Definition 1, where the challenge ciphertext is given by $\mathfrak{C}^* = \mathsf{Encrypt}_1(\mathtt{pk}_{i^*}, m_\delta)$, and $\mathcal{A}$ is disallowed from making $\mathsf{ODec}(\mathtt{pk}_{i^*}, \mathfrak{C}^*)$ query only. In particular, $\mathcal{A}$ can get* all *the re-encryption keys.*

**Delegator/Master Secret Security.** Delegator secret security[6] is considered in Ateniese *et al.* [4] which captures the intuition that, even if a dishonest proxy colludes with the delegatee, they still cannot derive the delegator's private key in full. The attack mode is quite simple and can be covered by the nontransformable / first-level ciphertext security [8].

## 3 Analysis of a CCA-Secure Unidirectional PRE Scheme

### 3.1 Review of Shao-Cao's Scheme

*SC09* [11] is reviewed as below, up to minor notational differences. We use □ to highlight the places which introduce the vulnerability.

[6] This notion is named as *master secret security* in [4] since the delegator's public key is the master public key in their secure distributed storage application. It is also called "collusion-resistance" in some literatures.

$\mathsf{Setup}(\kappa)$**:** Given a security parameter $\kappa$, choose three hash functions $H_1 : \{0,1\} \rightarrow \{0,1\}^{\ell_1}$, $H_2 : \{0,1\} \rightarrow \{0,1\}^{\ell_2}$, and $H_3 : \{0,1\} \rightarrow \{0,1\}^{\ell_3}$, where $\ell_1$, $\ell_2$ and $\ell_3$ are determined by $\kappa$, and the message space $\mathcal{M}$ is $\{0,1\}^{\ell_2}$. The parameters are $\mathtt{param} = (\kappa, H_1, H_2, H_3, \ell_1, \ell_2, \ell_3)$.

$\mathsf{KeyGen}()$**:** Given a security parameter $\kappa$, perform the following steps:

1. Choose two distinct Sophie Germain primes $p'$ and $q'$ of $\kappa$-bit.
2. Compute safe primes $p = 2p' + 1$ and $q = 2q' + 1$ (their primalities are guaranteed since $p'$ and $q'$ are Sophie Germain primes).
3. Compute a safe-prime modulus $N = pq$.
4. Store $\mathtt{sk} = (p, q, p', q')$ as the long term secret key.
5. Choose a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_{N^2}$.
6. Pick $a, b \xleftarrow{\$} [1, pp'qq']$, store $\mathtt{wsk} = (a, b)$ as the "weak" secret key.
7. Randomly pick $\alpha \in \mathbb{Z}^*_{N^2}$, set $g_0 = \alpha^2 \bmod N^2$, $g_1 = g_0^a \bmod N^2$, and $g_2 = g_0^b \bmod N^2$; the public key is $\mathtt{pk} = (H(), N, g_0, g_1, g_2)$.

Either secret key can be used to decrypt (any) ciphertexts, but both of them are required to produce a re-encryption key. Note that in the following description, the elements from the key of user $X$ contain an additional subscript of $X$, e.g., $\mathtt{pk}_X = (H_X(\cdot), N_X, g_{X0}, g_{X1} = g_{X0}^{a_X}, g_{X2})$.

$\mathsf{ReKeyGen}(\mathtt{sk}_X, \mathtt{pk}_Y)$**:** On input a long term secret key $(p_X, q_X, p'_X, q'_X)$, a weak secret $(a_X, b_X)$, and a public key $\mathtt{pk}_Y = (H_Y, N_Y, g_{Y0}, g_{Y1}, g_{Y2})$, it outputs the re-encryption key $rk_{X\rightarrow Y} = (rk^{(1)}_{X\rightarrow Y}, rk^{(2)}_{X\rightarrow Y})$, where $rk^{(1)}_{X\rightarrow Y} = (\dot{A}, \dot{B}, \dot{C})$, as follows:

1. Pick $\dot{\beta} \xleftarrow{\$} \{0,1\}^{\ell_1}$, compute $rk^{(2)}_{X\rightarrow Y} = a_X - \dot{\beta} \bmod (p_X q_X p'_X q'_X)$.
2. Pick $\dot{\sigma} \xleftarrow{\$} \mathbb{Z}_{N_Y}$, compute $r_{X\rightarrow Y} = H_Y(\dot{\sigma} \| \dot{\beta})$.
3. Compute $\dot{C} = H_1(\dot{\sigma}) \oplus \dot{\beta}$.
4. Compute $\dot{A} = (g_{Y0})^{r_{X\rightarrow Y}} \bmod (N_Y)^2$.
5. Compute $\dot{B} = (g_{Y2})^{r_{X\rightarrow Y}} \cdot (1 + \dot{\sigma} N_Y) \bmod (N_Y)^2$.

$\mathsf{Encrypt}(\mathtt{pk} = (H(), N, g_0, g_1, g_2), m)$**:** To encrypt a message $m \in \mathcal{M}$:

1. Randomly pick $\sigma \in \mathbb{Z}_N$, compute $r = H(\sigma \| m)$.
2. Compute $C = H_2(\sigma) \oplus m$.
3. Compute $A = (g_0)^r \bmod N^2$, $B = (g_1)^r \cdot (1 + \sigma N) \bmod N^2$ and $D = (g_2)^r \bmod N^2$.
4. Run $(c, s) \leftarrow \mathsf{SoK.Gen}(A, D, g_0, g_2, (B, C))$, where the underlying hash function is $H_3$.[7]
5. Output the ciphertext $\mathfrak{C} = (A, B, C, D, c, s)$.

[7] A signature of knowledge $(c, s)$ of the discrete logarithm of both $y_0 = g_0^x$ w.r.t. base $g_0$ and $y_2 = g_2^x$ w.r.t. base $g_2$, on a message $(B, C) \in \{0,1\}^*$ can be computed by first picking $t \in \{0, \ldots, 2^{|N^2|+k} - 1\}$, then computing $c = H_3(y_0 || y_2 || g_0 || g_2 || g_0^t || h_0^t || m)$ and $s = t - cx$. This requires 2 exponentiations.

ReEncrypt($\mathtt{rk}_{X\to Y}, \mathfrak{C}_X, \mathtt{pk}_X, \mathtt{pk}_Y$): On input a re-encryption key $\mathtt{rk}_{X\to Y} = (\mathtt{rk}^{(1)}_{X\to Y}, \mathtt{rk}^{(2)}_{X\to Y})$ and a ciphertext $\mathfrak{C} = (A, B, C, D, c, s)$ under key $\mathtt{pk}_X = (H_X, N_X, g_{X0}, g_{X1}, g_{X2})$,

1. Check if $c = H_3(A\|D\|g_{X0}\|g_{X2}\|(g_{X0})^s A^c\|(g_{X2})^s D^c \ |(B\|C))$. If not, return $\perp$.
2. Otherwise, compute $A' = A^{\mathtt{rk}^{(2)}_{X\to Y}}$.
3. Output $\mathfrak{C}_Y = (A, \boxed{A'}, B, C, \mathtt{rk}^{(1)}_{X\to Y}) = (A, A', B, C, \dot{A}, \dot{B}, \dot{C})$.

The only "new" thing in $\mathfrak{C}_Y$ is $A' = (g_{X0})^{r(a_X - \dot{\beta})} \bmod (N_X)^2 = (g_{X1})^r \boxed{(g_{X0})^{-r\dot{\beta}}} \bmod (N_X)^2$. The second equality holds since $g_{X1} = g_{X0}^{a_X}$, by the public key construction in KeyGen.

Decrypt($\mathtt{sk}, \mathfrak{C}$): On input a private key and a ciphertext $\mathfrak{C}$, parse $\mathfrak{C}$,

- If $\mathfrak{C}$ is an original ciphertext in the form $\mathfrak{C} = (A, B, C, D, c, s)$:
  1. Return $\perp$ if $c \neq H_3(A\|D\|g_0\|g_2\|(g_0)^s A^c\|(g_2)^s D^c\|(B\|C))$.
  2. If $\mathtt{sk}$ is in the form of $(a, b)$, compute $\sigma = \frac{B/(A^a) - 1 \bmod N^2}{N}$.
  3. If $\mathtt{sk} = (p, q, p', q')$, compute $\sigma = \frac{(B/g_0^{w_1})^{2p'q'} - 1 \bmod N^2}{N} \cdot \pi \bmod N$, where $w_1$ is computed as that in [18], and $\pi$ is the inverse of $2p'q' \bmod N$.
  4. Compute $m = C \oplus H_2(\sigma)$.
  5. If $B = (g_1)^{H(\sigma\|m)} \cdot (1 + \sigma N) \bmod N^2$, return $m$; else return $\perp$.
- If $\mathfrak{C} = (A, A', B, C, \dot{A}, \dot{B}, \dot{C})$ re-encrypted from $\mathtt{pk}_X$ to $\mathtt{pk}_Y$:
  1. If $\mathtt{sk}$ is in the form of $(a, b)$, compute $\dot{\sigma} = \frac{\dot{B}/(\dot{A}^b) - 1 \bmod N_Y^2}{N_Y}$.
  2. If $\mathtt{sk} = (p, q, p', q')$, similar to decrypting an original ciphertext, compute $\dot{\sigma} = \frac{(\dot{B}/g_{Y0}^{w_1})^{2p'q'} - 1 \bmod N_Y^2}{N_Y} \cdot \pi \bmod N_Y$, .
  3. Compute $\dot{\beta} = \dot{C} \oplus H_1(\dot{\sigma})$.
  4. If $\dot{B} \neq (g_{Y1})^{H_Y(\dot{\sigma}\|\dot{\beta})} \cdot (1 + \dot{\sigma} N_Y) \bmod N_Y^2$, return $\perp$.
  5. Compute $\sigma = (B/(\boxed{A' \cdot A^{\dot{\beta}}}) - 1 \bmod N_X^2)/N_X$.
  6. Compute $m = C \oplus H_2(\sigma)$.
  7. Return $m$ if $B = (g_{X1})^{H_X(\sigma\|m)} \cdot (1 + \sigma N_X) \bmod N_X^2$; else $\perp$.

  The **delegator**'s public key $(H_X, N_X, g_{X0}, g_{X1}, g_{X2})$ is required in the last few steps. This deviates from our framework in Section 2.

### 3.2 Our Attack

Shao and Cao [11] claimed that their PRE scheme is CCA-secure. However, in this section, we demonstrate that it is not the case.

Before describing our attack, we briefly explain how the re-encryption key is generated in *SC09*. Their ReKeyGen algorithm follows the "token-controlled encryption" paradigm, which is adopted by [13,14] and our scheme to be presented. Specifically, ReKeyGen first selects a random token $\dot{\beta}$ to "hide" (some form of) the delegator's secret key $a_X$ (i.e., $rk^{(2)}_{X\to Y} = a_X - \dot{\beta}$), and then encrypts this

token $\dot{\beta}$ under the delegatee's public key (i.e., $rk^{(1)}_{X\rightarrow Y} = (\dot{A}, \dot{B}, \dot{C})$). First, we found that *any* re-encryption query (not necessary of the challenge ciphertext) reveals partial information about $\dot{\beta}$. Moreover, there is no validity check on the $A'$ component of the transformed ciphertext. The combined effect leads us to the following efficient attacker $\mathcal{A}$, which aims to decrypt challenge ciphertext $\mathfrak{C}^* = (A, B, C, D, c, s)$ encrypted for $\mathtt{pk}^*_X = (H_X(\cdot), N_X, g_{X0}, g_{X1}, g_{X2})$.

1. Randomly pick $m \in \mathcal{M}$ and $r \in \mathbb{Z}_{(N_X)^2}$, compute $\mathfrak{C} \leftarrow \mathsf{Encrypt}_{\mathtt{pk}^*_X}(m; r)$, i.e., using $r$ as the randomness in the first step of Encrypt.
   *(Being a public key encryption, anyone can perform the encryption.)*
2. Issue a re-encryption oracle query to re-encrypt the ciphertext $\mathfrak{C}$ from $\mathtt{pk}^*$ to $\mathtt{pk}$, in particular, $\mathcal{A}$ obtains $Z' = g_{X0}^{r(a_X - \dot{\beta})}$ as the second component of the resulting transformed ciphertext $\mathfrak{C}_0$. *($Z'$ here corresponds to* $\boxed{A'}$ *in the above description of SC09.)*
3. Since $Z'$ is in the form of $(g_{X1})^r \boxed{(g_{X0})^{-r\dot{\beta}}} \bmod (N_X)^2$, $\mathcal{A}$ can compute $(g_{X0})^{-r\dot{\beta}} \leftarrow (Z'/(g_{X1})^r)$. *($\mathfrak{C}$ is prepared by $\mathcal{A}$, so $\mathcal{A}$ knows $r$.)*
4. Issue a re-encryption oracle query to re-encrypt the ciphertext $\mathfrak{C}^*$ from $\mathtt{pk}^*$ to $\mathtt{pk}$, and obtain $\mathfrak{C}_1 = (A, A', B, C, \dot{A}, \dot{B}, \dot{C})$ as a result.
   *(The secret key of* $\mathtt{pk}$ *is not compromised by $\mathcal{A}$, so this is legitimate.)*
5. Pick $s \xleftarrow{\$} \mathbb{Z}_{(N_X)^2}$, compute $\mathfrak{A}' \leftarrow A' \cdot (g_{X0}^{-r\dot{\beta}})^s$ and $\mathfrak{A} \leftarrow A \cdot (g_{X0})^{rs}$.
6. Prepare $\mathfrak{C}' = (\mathfrak{A}, \mathfrak{A}', B, C, \dot{A}, \dot{B}, \dot{C})$ issue a decryption oracle query under $\mathtt{pk}$ to decrypt $\mathfrak{C}'$, and the result is the message encrypted in $\mathfrak{C}^*$.

To see the correctness of the attack, first note that $B, C, \dot{A}, \dot{B}, \dot{C}$ just come from the derivative $(\mathtt{pk}, \mathfrak{C}_1)$ of the challenge $(\mathtt{pk}^*, \mathfrak{C}^*)$, and they are the only values from the ciphertext being used for the first three steps of Decrypt, so the correct value of $\dot{\beta}$ can be recovered. Moreover, in Decrypt (refer to $\boxed{A' \cdot A^{\dot{\beta}}}$), $\mathfrak{A}'\mathfrak{A}^{\dot{\beta}} = A'(g_{X0}{}^{-r\dot{\beta}})^s(A \cdot g_{X0}{}^{rs})^{\dot{\beta}} = A' \cdot g_{X0}^{-r\dot{\beta}s} \cdot A^{\dot{\beta}} \cdot g_{X0}^{r\dot{\beta}s} = A'A^{\dot{\beta}}$, which is exactly what Decrypt will compute for the challenge.

Finally, $\mathfrak{C}'$ is *not* a derivative of $\mathfrak{C}^*$. To check against the definition of derivative: 1) $\mathfrak{C}^* \neq \mathfrak{C}'$; 2) $\mathcal{A}$ has made two re-encryption queries, $\mathfrak{C}$ has *nothing* to do with the challenge $\mathfrak{C}^*$, only $(\mathtt{pk}, \mathfrak{C}_1)$ is considered as a derivative of the challenge, but $(\mathtt{pk}, \mathfrak{C}')$, where $\mathfrak{C}_1 \neq \mathfrak{C}'$, is *not* its derivative; and 3) $\mathcal{A}$ has not made any re-encryption key generation oracle query at all.

### 3.3 Flaws in the Proof and A Possible Fix

This attack originated from some flaws in their proof [11], specifically, two rejection rules regarding $A$ in the decryption oracle simulation. There is no checking of $A$ when decrypting a transformed ciphertext in the real scheme, which makes a noticeable difference to the adversary. The crux of our attack is the formulation of a new $A$ component. One possible fix is to re-compute $A$ in Decrypt and

check whether it is correctly generated, which requires one more exponentiation in $\mathbb{Z}_{N^2}$.

## 4 Our Proposed Unidirectional PRE Scheme

### 4.1 Construction

Our proposed unidirectional PRE scheme extends the *bidirectional* scheme proposed by Deng *et al.* [10], again by the "token-controlled encryption" technique. As previously discussed in Section 3, however, this should be carefully done to avoid possible attacks.

$\mathsf{Setup}(\kappa)$**:** Choose two primes $p$ and $q$ such that $q|p-1$ and the bit-length of $q$ is the security parameter $\kappa$. Let $g$ be a generator of group $\mathbb{G}$, which is a subgroup of $\mathbb{Z}_q^*$ with order $q$. Choose four hash functions $H_1 : \{0,1\}^{\ell_0} \times \{0,1\}^{\ell_1} \to \mathbb{Z}_q^*, H_2 : \mathbb{G} \to \{0,1\}^{\ell_0+\ell_1}, H_3 : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_4 : \mathbb{G} \to \mathbb{Z}_q^*$. The former three will be modeled as random oracles in our security proof. Here $\ell_0$ and $\ell_1$ are security parameters determined by $\kappa$, and the message space $\mathcal{M}$ is $\{0,1\}^{\ell_0}$. The parameters are $\mathtt{param} = (q, \mathbb{G}, g, H_1, H_2, H_3, H_4, \ell_0, \ell_1)$.

$\mathsf{KeyGen}()$**:** Pick $\mathtt{sk}_i = (x_{i,1} \xleftarrow{\$} \mathbb{Z}_q^*, x_{i,2} \xleftarrow{\$} \mathbb{Z}_q^*)$ and set $\mathtt{pk}_i = (\mathtt{pk}_{i,1}, \mathtt{pk}_{i,2}) = (g^{x_{i,1}}, g^{x_{i,2}})$.

$\mathsf{ReKeyGen}(\mathtt{sk}_i, \mathtt{pk}_j)$**:** On input user $i$'s private key $\mathtt{sk}_i = (x_{i,1}, x_{i,2})$ and user $j$'s public key $\mathtt{pk}_j = (\mathtt{pk}_{j,1}, \mathtt{pk}_{j,2})$, this algorithm generates the re-encryption key $\mathtt{rk}_{i\to j}$ as below:

1. Pick $h \xleftarrow{\$} \{0,1\}^{\ell_0}$ and $\pi \xleftarrow{\$} \{0,1\}^{\ell_1}$, compute $v = H_1(h, \pi)$.
2. Compute $V = \mathtt{pk}_{j,2}^v$ and $W = H_2(g^v) \oplus (h \| \pi)$.
3. Define $\mathtt{rk}_{i\to j}^{\langle 1 \rangle} = \frac{h}{x_{i,1} H_4(\mathtt{pk}_{i,2}) + x_{i,2}}$. Return $\mathtt{rk}_{i\to j} = (\mathtt{rk}_{i\to j}^{\langle 1 \rangle}, V, W)$.

$\mathsf{Encrypt}(\mathtt{pk}_i = (\mathtt{pk}_{i,1}, \mathtt{pk}_{i,2}), m)$**:** To encrypt a plaintext $m \in \mathcal{M}$:

1. Pick $u \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $D = \left(\mathtt{pk}_{i,1}^{H_4(\mathtt{pk}_{i,2})} \mathtt{pk}_{i,2}\right)^u$.
2. Pick $\omega \xleftarrow{\$} \{0,1\}^{\ell_1}$, compute $r = H_1(m, \omega)$.
3. Compute $E = \left(\mathtt{pk}_{i,1}^{H_4(\mathtt{pk}_{i,2})} \mathtt{pk}_{i,2}\right)^r$ and $F = H_2(g^r) \oplus (m \| \omega)$.
4. Compute $s = u + r \cdot H_3(D, E, F) \mod q$.
5. Output the ciphertext $\mathfrak{C} = (D, E, F, s)$.

$\mathsf{ReEncrypt}(\mathtt{rk}_{i\to j}, \mathfrak{C}_i, \mathtt{pk}_i, \mathtt{pk}_j)$**:** On input a re-encryption (user $i$ to user $j$) key $\mathtt{rk}_{i\to j} = (\mathtt{rk}_{i\to j}^{\langle 1 \rangle}, V, W)$, an original ciphertext $\mathfrak{C}_i = (D, E, F, s)$ under public key $\mathtt{pk}_i = (\mathtt{pk}_{i,1}, \mathtt{pk}_{i,2})$, this algorithm re-encrypts $\mathfrak{C}_i$ into another one under public key $\mathtt{pk}_j = (\mathtt{pk}_{j,1}, \mathtt{pk}_{j,2})$ as follows:

1. If $\left(\mathtt{pk}_{i,1}^{H_4(\mathtt{pk}_{i,2})}\mathtt{pk}_{i,2}\right)^{s} = D \cdot E^{H_3(D,E,F)}$ does not hold, return $\perp$.
2. Otherwise, compute $E' = E^{\mathtt{rk}_{i\to j}^{\langle 1\rangle}}$, and output $(E', F, V, W)$.

Let $r = H_1(m, \omega)$, $v = H_1(h, \pi)$, the transformed ciphertext is of the following forms:

$$\mathfrak{C}_j = (E', F, V, W) = \left(g^{r\cdot h}, H_2(g^r) \oplus (m\|\omega), \mathtt{pk}_{j,2}^{v}, H_2(g^v) \oplus (h\|\pi)\right).$$

$\mathsf{Encrypt}_1(\mathtt{pk}_i = (\mathtt{pk}_{i,1}, \mathtt{pk}_{i,2}), m)$**:** To create a nontransformable ciphertext under public key $\mathtt{pk}_i$ of a message $m \in \mathcal{M}$:

1. Pick $h \xleftarrow{\$} \{0,1\}^{\ell_0}$ and $\pi \xleftarrow{\$} \{0,1\}^{\ell_1}$, compute $v = H_1(h, \pi)$.
2. Compute $V = \mathtt{pk}_{j,2}^{v}$ and $W = H_2(g^v) \oplus (h\|\pi)$.
3. Pick $\omega \xleftarrow{\$} \{0,1\}^{\ell_1}$, compute $r = H_1(m, \omega)$.
4. Output the ciphertext $\mathfrak{C} = (E', F, V, W)$.

$\mathsf{Decrypt}(\mathtt{sk}_i, \mathfrak{C}_i)$**:** On input a private key $\mathtt{sk}_i = (x_{i,1}, x_{i,2})$ and ciphertext $\mathfrak{C}_i$, parse $\mathfrak{C}_i$, then work according to two cases:

- $\mathfrak{C}$ is an original ciphertext in the form $\mathfrak{C} = (D, E, F, s)$:
  1. If $\left(\mathtt{pk}_{i,1}^{H_4(\mathtt{pk}_{i,2})}\mathtt{pk}_{i,2}\right)^{s} = D \cdot E^{H_3(D,E,F)}$ does not hold, return $\perp$.
  2. Otherwise, compute $(m\|\omega) = F \oplus H_2(E^{\frac{1}{x_{i,1}H_4(\mathtt{pk}_{i,2})+x_{i,2}}})$.
  3. Return $m$ if $E = \left(\mathtt{pk}_{i,1}^{H_4(\mathtt{pk}_{i,2})}\mathtt{pk}_{i,2}\right)^{H_1(m,\omega)}$ holds; else return $\perp$.
- $\mathfrak{C}$ is a transformed ciphertext in the form $\mathfrak{C} = (E', F, V, W)$:
  1. Compute $(h\|\pi) = W \oplus H_2(V^{1/\mathtt{sk}_{i,2}})$ and $(m\|\omega) = F \oplus H_2(E'^{1/h})$.
  2. Return $m$ if $V = \mathtt{pk}_{i,2}^{H_1(h,\pi)}$ and $E' = g^{H_1(m,\omega)\cdot h}$ hold; else $\perp$.

### 4.2 Security Analysis

The intuition of CCA security can be seen from the below properties.

1. The validity of the original ciphertexts can be *publicly verifiable* by everyone including the proxy; otherwise, it will suffer from an attack as illustrated in [10]. For our scheme, the ciphertext component $(D, s)$ in the original ciphertext $(D, E, F, s)$ can be viewed as a signature signing the "message" $(E, F)$, that is how we get pubic verifiability.
2. The original ciphertexts should be CCA-secure. The original ciphertext produced by our scheme is indeed a "hashed" CCA-secure ElGamal encryption tightly integrated with a Schnorr signature.
3. The transformed ciphertexts should be CCA-secure In our scheme, a transformed ciphertext can be viewed as two seamlessly integrated "hashed" CCA-secure ElGamal encryptions.

We make four observations on the re-encryption key computation.

1. It takes the input of $\mathtt{sk}_i$, but not $\mathtt{sk}_j$, so our scheme is unidirectional.
2. Even though $h$ can be recovered by anyone who owns $\mathtt{sk}_j$, $\mathtt{rk}_{i\to j}^{\langle 1\rangle}$ only gives information about $x_{i,1}H_4(\mathtt{pk}_{i,2}) + x_{i,2}$ (no matter whom the delegatee $j$ is),

but not the concrete value of $x_{i,1}$ or $x_{i,2}$. This gives an intuition why our scheme achieves delegator secret security.

3. A collusion of the delegatee and the proxy cannot recover $x_{i,1}$, which is needed to decrypt original ciphertexts.
4. If the delegatee $j$ is now a delegator to someone else (say $k$). Again, only $x_{j,1}H_4(\mathtt{pk}_{j,2}) + x_{j,2}$ is known to a collusion of the delegatee $k$ and a proxy, which is not useful in recovering the token $h$ in $\mathtt{rk}_{i\to j}$, hence the chain collusion attack suffered by [13,14] does not apply.

**Theorem 1.** *Our scheme is IND-PRE-CCA secure in the random oracle model, if the CDH assumption holds in group $\mathbb{G}$ and the Schnorr signature [17] is existentially unforgeable against chosen message attack.*

The detailed proof can be found in the full version of this paper [19]. The proof first uses Coron's technique [20] to implant our hard problem to many uncorrupted public keys. At the same time, for those uncorrupted public keys which is generated as usual (without the problem embedded), re-encryption key can still be generated with non-negligible probability.

To prove the original ciphertext security is relatively simple. For transformed ciphertext, an implicitly defined random $h$ value which is unknown to the simulator may be used in the re-encryption key returned as the response to the oracles query. To answer decryption oracle queries, the simulator can extract the random $h$ value used from the random oracle and unwrap the given ciphertext. For the challenge ciphertext generation, our definition of security rules out the case that both the delegator and the proxy are compromised, so any partial information regarding the value of $h$ used in the re-encryption key would not affect the (different) $h$ value associated with the challenge ciphertext.

For nontransformable ciphertext security, the situation is much simpler. The $h$ value used in the challenge ciphertext is essentially a one-time pad, and the reduction boils down to the underlying hashed ElGamal encryption, so the simulator can compute all the re-encryption keys.

### 4.3 Efficiency Comparisons

In Table 2, we compare our scheme with *SC09* [11] with our suggested fix. We use $t_{\text{exp}}$ to denote the computational cost of an exponentiation. In our calculation, a multi-exponentiation (m-exp) (which we assume it multiplies only up to 3 exponentiations in one shot) is considered as $1.5t_{\text{exp}}$. Encrypt of *LV08*, ReEncrypt and Decrypt($\mathfrak{C}$) of *SC09* used 1, 2 and 2 m-exp respectively. In our scheme, we assume $\mathtt{pk}_{i,1}^{H_4(\mathtt{pk}_{i,2})}\mathtt{pk}_{i,2}$ is pre-computed. Even not, it only adds at most $1t_{\text{exp}}$ in Encrypt, ReEncrypt and Decrypt($\mathfrak{C}$) using m-exp, since there are other exponentiations to be done. The comparison indicates that our scheme beats *SC09* in all aspects.

**Table 2. Comparisons of Unidirectional Proxy Re-Encryption Schemes.** $\mathfrak{C}$ denotes an original ciphertext and $\mathfrak{C}'$ denotes a transformed ciphertext, $|\mathfrak{C}|$ and $|\mathfrak{C}'|$ are their size. $N_X$ ($N_Y$) is the safe-prime modulus used by the delegator (delegatee).

| Schemes | *SC09* [11] | Our Scheme |
|---|---|---|
| Encrypt | $5t_{\text{exp}}$ (in $\mathbb{Z}_{N^2}$) | $3t_{\text{exp}}$ (in $\mathbb{G}$) |
| ReEncrypt | $4t_{\text{exp}}$ (in $\mathbb{Z}_{N^2}$) | $2.5t_{\text{exp}}$ (in $\mathbb{G}$) |
| Decrypt($\mathfrak{C}$) | $5t_{\text{exp}}$ (in $\mathbb{Z}_{N^2}$) | $3.5t_{\text{exp}}$ (in $\mathbb{G}$) |
| Decrypt($\mathfrak{C}'$) | $5t_{\text{exp}}$ (in $\mathbb{Z}_{N^2}$) | $4t_{\text{exp}}$ (in $\mathbb{G}$) |
| $\lvert\mathfrak{C}\rvert$ | $2k + 3\lvert(N_X)^2\rvert + \lvert m\rvert$ | $3\lvert\mathbb{G}\rvert + \lvert\mathbb{Z}_q\rvert$ |
| $\lvert\mathfrak{C}'\rvert$ | $\ell_1 + 3\lvert(N_X)^2\rvert + 2\lvert(N_Y)^2\rvert + \lvert m\rvert$ | $2\lvert\mathbb{G}\rvert + 2\lvert\mathbb{Z}_q\rvert$ |
| Security | Not Collusion-Resistant | CCA-Secure |
| Assumption | DDH over $\mathbb{Z}_{N^2}$ | CDH over $\mathbb{G}$ |
| RO-Free | × | × |
| Nature of Decrypt | Decryption of $\mathfrak{C}'$ requires $\mathtt{pk}_X$ of the delegator | No delegator public key is required |

## 5 Conclusions

Most existing unidirectional proxy re-encryption (PRE) schemes rely on pairing except a recently proposed scheme by Shao and Cao [11]. However, we showed that their CCA-security proof in the random oracle model is flawed, and presented a concrete attack. Possible fixes of their scheme further degrades either the decryption efficiency or the transformed ciphertext length. We then presented a natural construction of CCA-secure unidirectional PRE scheme without pairings that is very efficient.

Our scheme is single-hop and relies on the random oracle. It would be interesting to construct a multi-hop scheme in the standard model. It seems to be possible to use the token-controlled encryption approach to build a multi-hop scheme; however, the design may be inelegant and the efficiency may not be ideal. We remark that our scheme is proven under a relaxed security definition. We left it as an open problem to devise a pairing-free CCA-secure scheme without this relaxation. Another interesting problem, which possibly requires a different set of techniques, is to construct other schemes in proxy re-cryptography, such as conditional PRE schemes [21] and proxy re-signatures [22,23], without pairings.

## References

1. Mambo, M., Okamoto, E.: Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E80-A(1), 54–63 (1997)
2. Blaze, M., Bleumer, G., Strauss, M.: Divertible Protocols and Atomic Proxy Cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)

3. Smith, T.: DVD Jon: Buy DRM-less Tracks from Apple iTunes (2005), http://www.theregister.co.uk/2005/03/18/itunes_pymusique
4. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. ACM Trans. Inf. Syst. Secur. 9(1), 1–30 (2006)
5. Ivan, A.A., Dodis, Y.: Proxy Cryptography Revisited. In: NDSS. The Internet Society (2003)
6. Hohenberger, S., Rothblum, G.N., Shelat, A., Vaikuntanathan, V.: Securely Obfuscating Re-encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 233–252. Springer, Heidelberg (2007)
7. Canetti, R., Hohenberger, S.: Chosen-Ciphertext Secure Proxy Re-Encryption. In: ACM Conference on Computer and Communications Security, pp. 185–194. ACM, New York (2007)
8. Libert, B., Vergnaud, D.: Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008)
9. Libert, B., Vergnaud, D.: Tracing Malicious Proxies in Proxy Re-encryption. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 332–353. Springer, Heidelberg (2008)
10. Deng, R.H., Weng, J., Liu, S., Chen, K.: Chosen-Ciphertext Secure Proxy Re-encryption without Pairings. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 1–17. Springer, Heidelberg (2008)
11. Shao, J., Cao, Z.: CCA-Secure Proxy Re-encryption without Pairings. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 357–376. Springer, Heidelberg (2009)
12. Ateniese, G., Benson, K., Hohenberger, S.: Key-Private Proxy Re-encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 279–294. Springer, Heidelberg (2009)
13. Green, M., Ateniese, G.: Identity-Based Proxy Re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007)
14. Chu, C.K., Tzeng, W.G.: Identity-Based Proxy Re-encryption Without Random Oracles. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 189–202. Springer, Heidelberg (2007)
15. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing Chosen-Ciphertext Security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003)
16. Gamal, T.E.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
17. Schnorr, C.P.: Efficient Signature Generation by Smart Cards. J. Cryptology 4(3), 161–174 (1991)
18. Bresson, E., Catalano, D., Pointcheval, D.: A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 37–54. Springer, Heidelberg (2003)
19. Chow, S.S.M., Weng, J., Yang, Y., Deng, R.H.: Efficient Unidirectional Proxy Re-Encryption. Cryptology ePrint Archive, Report 2009/189 (2009), http://eprint.iacr.org/

20. Coron, J.S.: On the Exact Security of Full Domain Hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000)
21. Chu, C.K., Weng, J., Chow, S.S.M., Zhou, J., Deng, R.H.: Conditional Proxy Broadcast Re-Encryption. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 327–342. Springer, Heidelberg (2009)
22. Chow, S.S.M., Phan, R.C.W.: Proxy Re-signatures in the Standard Model. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 260–276. Springer, Heidelberg (2008)
23. Libert, B., Vergnaud, D.: Multi-use Unidirectional Proxy Re-Signatures. In: ACM Conference on Computer and Communications Security, pp. 511–520. ACM, New York (2008)

# Public-Key Encryption with Non-interactive Opening: New Constructions and Stronger Definitions

David Galindo[1], Benoît Libert[2], Marc Fischlin[3], Georg Fuchsbauer[4], Anja Lehmann[3], Mark Manulis[3], and Dominique Schröder[3]

[1] University of Luxembourg
david.galindo@uni.lu
[2] Université catholique de Louvain, Crypto Group, Belgium
[3] TU Darmstadt & CASED, Germany
[4] École normale supérieure, LIENS - CNRS - INRIA, Paris, France

**Abstract.** Public-key encryption schemes with non-interactive opening (PKENO) allow a receiver to non-interactively convince third parties that a ciphertext decrypts to a given plaintext or, alternatively, that such a ciphertext is invalid. Two practical generic constructions for PKENO have been proposed so far, starting from either identity-based encryption or public-key encryption with witness-recovering decryption (PKEWR). We show that the known transformation from PKEWR to PKENO fails to provide chosen-ciphertext security; only the transformation from identity-based encryption remains thus valid. Next, we prove that PKENO can alternatively be built out of robust non-interactive threshold public-key cryptosystems, a primitive that differs from identity-based encryption. Using the new transformation, we construct two efficient PKENO schemes: one based on the Decisional Diffie-Hellman assumption (in the Random-Oracle Model) and one based on the Decisional Linear assumption (in the standard model). Last but not least, we propose new applications of PKENO in protocol design. Motivated by these applications, we reconsider proof soundness for PKENO and put forward new definitions that are stronger than those considered so far. We give a taxonomy of all definitions and demonstrate them to be satisfiable.

**Keywords:** public-key encryption, non-interactive proofs, security definitions, constructions.

## 1 Introduction

Public-key encryption allows a receiver Bob to generate a pair of a private and a public key $(sk_B, pk_B)$ such that anyone can encrypt messages under $pk_B$ which can only be decrypted by Bob who knows $sk_B$. The primitive *public-key encryption with non-interactive opening* (PKENO), introduced by Damgård et al. [DHKT08], allows Bob to prove to a verifier Alice that a given ciphertext $C$ decrypts to a certain message. By using PKENO, Bob can do so convincingly

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 333–350, 2010.

and without further interaction, neither with Alice nor with the original sender of the ciphertext. More precisely, Bob runs a proving algorithm $\mathsf{Prove}$ on inputs his secret key $sk_B$ and the intended ciphertext $C$, thereby generating a proof $\pi$. On the other hand, Alice runs a verification algorithm $\mathsf{Ver}$ on inputs Bob's public key $pk_B$, ciphertext $C$, a plaintext $m$, and an opening proof $\pi$. The soundness property guarantees that the verification algorithm outputs 1 if $C$ was indeed an encryption of $m$, and 0 otherwise. An interesting feature of PKENO is that Bob can also convince Alice of the fact that a given ciphertext $C$ is *invalid*, i.e., it is rejected by the decryption algorithm. PKENO turns out to be a useful primitive for protocol design. In addition to the use of PKENO in multiparty computation protocols, as highlighted in [DT07, DHKT08], we identify further applications, which we introduce below.

Secure Message Transmission with PKENO. One of the classical ways to realize secure message transmission in a public-key setting is to let the sender encrypt the message and then sign the ciphertext, i.e., the so-called *encrypt-then-sign* paradigm [ADR02] in which the transmitted ciphertext also includes a signature $\mathsf{Sign}(sk_s, \mathsf{Enc}(pk_r, pk_s||m)||pk_r)$, with $sk_s$ being the signing key of the sender and $pk_r$ the encryption key of the receiver. If the sender uses a standard PKE scheme, the receiver is in general not able to provide a non-repudiable proof for the origin of the received message $m$. To do so, the receiver should convincingly open the encryption $\mathsf{Enc}(pk_r, pk_s||m)$, which he cannot do, unless he is willing to expose his decryption key $sk_r$. Replacing PKE with PKENO allows the receiver to prove the origin for the decrypted *message*, and thus authenticated encryption with non-repudiation is achieved.

Group Signatures. The most common way to achieve anonymity in group signatures [CvH91] is the following: a group member first encrypts his membership certificate under the opener's public key while adding a non-interactive proof of validity of the encrypted data. The opening authority is then able to identify the signer by merely decrypting the ciphertext.

In the model of dynamic group signatures given by Bellare et al. [BSZ05], the opening authority is required to give a *proof* that it traced the correct user. Using PKENO rather than plain encryption enables the opener to do so in a simple manner. In the game modeling the anonymity of signatures in [BSZ05], an adversary is given an opening oracle that opens adversarially-chosen signatures and outputs proofs of correct opening. The security of the employed PKENO scheme (together with simulation-sound zero knowledge of the proof of well-formedness) ensures that an adversary cannot distinguish signatures from distinct users.

### 1.1 Our Contributions

Difficulty of Building PKENO. Damgård et al. [DHKT08] showed that a PKENO can be built out of Identity-Based Encryption (IBE). Although IBE can now be realized under a variety of assumptions and without bilinear maps (see [BGH07, GPV08, AB09, CHK09, Pei09] for instance), it remains a very specialized and powerful cryptographic primitive. Towards narrowing the gap between

sufficient and necessary conditions for PKENO, it is interesting to see whether it can be obtained without resorting to all the functionalities provided by IBE (e.g. non-interactive user key derivation). In [DHKT08], the authors mentioned that PKENO can also be based upon a seemingly weaker primitive, called *public-key encryption with witness-recovering decryption* (PKEWR) [PW08]. In a PKEWR scheme, the receiver Bob is able to recover the random coins $r$ used to encrypt a ciphertext $C$. Damgård *et al.* proposed to use the coins $r$ as the proof, and verification proceeds by re-encrypting $C' = \mathsf{Enc}(pk_B, m; r)$ and checking whether $C = C'$. However, this approach can only be guaranteed to be sound for *valid* ciphertexts, i.e., ciphertexts that have been output by the encryption algorithm. As a consequence, for invalid ciphertexts "the coins used to construct $C$" might not be well defined. Indeed, we show in Section 4.1 how the (apparently) straightforward construction of PKENO out of PKEWR fails to provide security in the sense of [DHKT08]. This then motivates the quest for both new generic and concrete constructions for PKENO.

Non-Interactive Threshold Cryptosystems Imply PKENO. Somewhat surprisingly, we show that starting from a robust *non-interactive threshold cryptosystem* (TPKC), we can construct a PKENO scheme in a generic way. We only ask the threshold cryptosystem to satisfy some appropriate notion of decryption consistency. We emphasize that, although this notion is stronger than the one initially formalized by Shoup and Gennaro [SG98], it remains fairly mild in that most known robust threshold cryptosystems satisfy it.

Threshold cryptosystems distribute the ability to decrypt among several parties. The private decryption key is shared among $n$ servers such that at least $t$ servers are needed for decryption. If the *combiner* wishes to decrypt some ciphertext $C$, it sends $C$ to the decryption servers. After receiving at least $t$ partial decryption shares from the servers, the combiner is able to reconstruct the plaintext from these shares. A *robust* TPKC [SG98, BBH06] provides the additional property that, whenever the decryption of valid ciphertexts fails, the combiner can sieve out bad decryption shares and reveal the identity of the server having sent an invalid partial decryption. We show an efficient transformation from robust TPKC to PKENO. When applied to the schemes in [SG98, AT09], the conversion provides new practical PKENO schemes based on the Decisional Diffie-Hellman (in the random oracle model) and the Decisional Linear assumptions, respectively.

Stronger Soundness Definitions. The main motivation for introducing PKENO was protocol design: some player sends a message to Bob securely by encrypting it under Bob's public key. If Bob finds out (possibly later) that the message is somehow "invalid", he can convince other participants of this fact without getting back to the (possibly) dishonest sender. *Proof soundness* ensures that Bob can do so convincingly; in particular, it states that if a ciphertext $C$ encrypts a message $m$, then Bob cannot make a proof for $C$ being an encryption of a different message $m'$ (including the case of invalid messages $m' = \perp$). In the game that formally defines this security notion [DHKT08, Gal09], the challenger produces a private/public key pair, hands it to the adversary, who outputs a

message of which he receives an encryption $C$. The adversary wins if he outputs a different message and makes a valid proof that this was the opening.

Thus, previous definitions of proof soundness [DHKT08, Gal09] only considered the case of honestly chosen keys, where a malicious receiver tries to claim a different decryption result under the genuine keys. In real-world applications, however, the keys are usually chosen by the users themselves. It seems thus natural to let *the adversary* choose the keys in the security experiment to reflect this fact. Hence, we define two stronger flavors of proof soundness, where the first one is analogous to the original definition given by [DHKT08], but lets the adversary choose his keys. The second one is akin to the binding property of commitment schemes and states that no adversary can find a public key, a ciphertext with two messages and valid proofs for each of them. We relate all notions formally.

Note that strengthening proof soundness also makes sense for the other applications given above. It can be used towards reducing the need for trusted setup in group signatures: the opener could choose his opening key and add corresponding information to the public parameters. Strong proof soundness then guarantees non-frameability even in this setting.

A Note on PKENO From General Assumptions. In [DHKT08], Damgård et al. already discussed how to construct PKENO from general assumptions using general but rather inefficient non-interactive zero-knowledge (NIZK) proofs. The idea of the construction is as follows. The receiver commits initially to its secret key. Whenever the proof algorithm is executed, it outputs a non-interactive zero-knowledge proof showing that the secret key committed to corresponds to the public key, and that decryption of the ciphertext $C$ indeed yields the message $m$. Although this construction satisfies the security definitions of [DHKT08], it does not seem to be sufficient for our stronger soundness definitions. In particular, this construction does not make any statements about "invalid" ciphertexts.

Nonetheless, we briefly discuss here how to modify the idea in order to satisfy our stronger definitions, obtaining a scheme under general assumptions meeting our security notions. In our modification, the encryption algorithm adds a NIZK proof showing the well-formedness of the ciphertext (somehow in the fashion of [NY90, Sah99]) under the public-key, allowing anyone to detect invalid ciphertexts. The prove algorithm then rejects any ciphertext whose NIZK proof is invalid. If, on the other hand, the NIZK proof in the ciphertext is valid, then the prove algorithm proceeds as before, computing a second NIZK proof as described by Damgård et al. We note that, in the scheme by [DHKT08] with weak soundness, the common reference string (CRS) for the NIZK proofs can be put into the honestly chosen public key. In contrast, for stronger soundness with adversarially chosen keys (as in our case), we need to assume that the CRS is a public parameter (common reference string model).

Future Work. We leave as an open problem the construction of an efficient PKENO scheme based on a standard assumption like the Decision Diffie-Hellman assumption in the standard model.

## 2 Preliminaries

NOTATION. If $x$ is a string then $|x|$ denotes its length, while if $S$ is a set then $|S|$ denotes its size. If $k$ is a natural number, then $1^k$ denotes the string of $k$ ones. If $S$ is a set then $s_1, \ldots, s_n \xleftarrow{\$} S$ denotes the operation of picking $n$ elements $s_i$ of $S$ independently and uniformly at random. We write $\mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and by $z \leftarrow \mathcal{A}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs $(x, y, \ldots)$ and letting $z$ be the output. The abbreviation PPT refers to "probabilistic polynomial-time" algorithms [Gol01].

### 2.1 Public Key Encryption with Non-interactive Opening

A PKENO scheme $\mathsf{PKENO} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Prove}, \mathsf{Ver})$ is a tuple of five PPT algorithms:

- $\mathsf{Gen}$ is a randomized algorithm taking as input a security parameter $1^k$ and returns a key pair $(pk, sk)$, where the public key $pk$ includes a description of the message space $\mathcal{M}_{pk}$.
- $\mathsf{Enc}$ is a probabilistic algorithm taking as inputs a public key $pk$ and a message $m \in \mathcal{M}_{pk}$. It returns a ciphertext $C$.
- $\mathsf{Dec}$ is a deterministic algorithm that takes as inputs a ciphertext $C$ and a secret key $sk$. It returns a message $m \in \mathcal{M}_{pk}$ or the special symbol $\perp$ meaning that $C$ is invalid.
- $\mathsf{Prove}$ is a probabilistic algorithm taking as inputs a ciphertext $C$ and a secret key $sk$. It returns a proof $\pi$.
- $\mathsf{Ver}$ is a deterministic algorithm taking as inputs a public key $pk$, a ciphertext $C$, a plaintext $m$ and a proof $\pi$. It returns a result $res \in \{0, 1\}$ meaning accepted and rejected proof, respectively. In particular, $\mathsf{Ver}(pk, C, \perp, \pi) = 1$ must be interpreted as the verifier being convinced that $C$ is an invalid ciphertext.

Correctness requires that for an honestly generated key pair $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$, it holds that:

- For all messages $m \in \mathcal{M}_{pk}$ we have $\Pr\left[\,\mathsf{Dec}\big(sk, \mathsf{Enc}(pk, m)\big) = m\,\right] = 1$.
- For all ciphertexts $C$, $\Pr\left[\,\mathsf{Ver}\big(pk, C, \mathsf{Dec}(sk, C), \mathsf{Prove}(sk, C)\big) = 1\,\right] = 1$.

Security of PKENO is defined by indistinguishability under chosen-ciphertext and prove attacks (IND-CCPA) and proof soundness [DHKT08, Gal09]. We formally define both notions and propose strengthened definitions for proof soundness in Section 3.

**Definition 1 (IND-CCPA security).** *Let us consider the following game between a challenger and an adversary $\mathcal{A}$:*

**Setup:** *The challenger runs $\mathsf{Gen}(1^k)$ and gives $pk$ to $\mathcal{A}$.*

**Phase 1:** *The adversary issues queries of the form:*

*a) decryption query to an oracle* $\mathsf{Dec}(sk, \cdot)$*;*
*b) proof query to an oracle* $\mathsf{Prove}(sk, \cdot)$*.*

*These may be asked adaptively in that they may depend on the answers to previous queries.*

**Challenge:** *At some point,* $\mathcal{A}$ *outputs two equal-length messages* $m_0, m_1 \in \mathcal{M}_{pk}$*. The challenger chooses a random bit* $\beta$ *and returns* $C^\star \leftarrow \mathsf{Enc}(pk, m_\beta)$*.*

**Phase 2:** *As Phase 1, except that neither decryption nor proof queries on* $C^\star$ *are allowed.*

**Guess:** *The adversary* $\mathcal{A}$ *outputs a guess* $\beta' \in \{0, 1\}$*. The adversary wins the game if* $\beta = \beta'$*.*

*Define* $\mathcal{A}$*'s advantage as* $\mathsf{Adv}^{\text{ind-ccpa}}_{\mathsf{PKENO},\mathcal{A}}(1^k) := \left| \Pr[\beta' = \beta] - \frac{1}{2} \right|$*. A scheme* PKENO *is called* indistinguishable against chosen-ciphertext and prove attacks (IND-CCPA secure) *if for every PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\text{ind-ccpa}}_{\mathsf{PKENO},\mathcal{A}}(\cdot)$ *is negligible.*

We recall the original definition [DHKT08, Gal09] of proof soundness under genuine keys:

**Definition 2 (Proof Soundness).** *Consider the following game between a challenger and an adversary* $\mathcal{A}$*:*

**Stage 0:** *The challenger runs* $\mathsf{Gen}(1^k)$ *and gives the output* $(pk, sk)$ *to* $\mathcal{A}$*.*

**Stage 1:** *The adversary chooses a message* $m \in \mathcal{M}_{pk}$*.*

**Stage 2:** *The challenger computes* $C \leftarrow \mathsf{Enc}(pk, m)$ *and gives it to* $\mathcal{A}$ *which returns* $(m', \pi')$*.*

$\mathcal{A}$*'s advantage is defined as the probability*

$$\mathbf{Adv}^{\text{proof-snd}}_{\mathsf{PKENO},\mathcal{A}}(1^k) := \Pr\left[\mathsf{Ver}(pk, C, m', \pi') = 1 \ \wedge \ m' \neq m\right] \ .$$

*A scheme* PKENO *is* proof sound *if for every PPT adversary* $\mathcal{A}$ *its advantage is negligible.*

In the above definition it is understood that $\perp \notin \mathcal{M}_{pk}$ and that the adversary thus also wins if $\pi'$ is a valid proof for $m' = \perp$. It is also worth insisting that, since the adversary obtains the private key at the beginning of the game, decryption or prove oracles would be redundant.

### 2.2 Robust Non-interactive Threshold Public-Key Cryptosystems

Non-interactive threshold public-key cryptosystems, as formalized in [SG98], distribute the ability to decrypt among several parties. The private decryption key is shared among $n$ servers such that at least $t$ servers are needed for decryption. If the *combiner* wishes to decrypt some ciphertext $C$, it sends $C$ to the decryption servers. After receiving at least $t$ partial decryption shares from the servers, the combiner is able to reconstruct the plaintext from these shares. A *robust* TPKC [SG98, BBH06] provides the additional property that whenever the decryption

of valid ciphertexts fails, the combiner can sieve out bad decryption shares and reveal the identity of the server having sent an invalid partial decryption.

SYNTAX. We use the same syntax as Boneh-Boyen-Halevi [BBH06] and Shoup-Gennaro [SG98] for (robust) non-interactive threshold public-key cryptostyems (TPKC). Formally, a robust TPKC scheme

$$\mathsf{TPKC} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{ShareDecrypt}, \mathsf{ShareVerify}, \mathsf{Combine})$$

consists of the following algorithms:

- $\mathsf{Setup}(n, t, 1^k)$ takes as input a security parameter $1^k$ and integers $t, n \in \mathbb{N}$ (with $1 \leq t \leq n$) denoting the number of decryption servers $n$ and the decryption threshold $t$. It outputs a triple $(\mathsf{PK}, \mathbf{VK}, \mathbf{SK})$, where $\mathsf{PK}$ is the public key, $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$ is a vector of $n$ private-key shares and $\mathbf{VK} = (\mathsf{VK}_1, \ldots, \mathsf{VK}_n)$ is the corresponding vector of verification keys. Decryption server $i$ is given the share $(i, \mathsf{SK}_i)$ that allows to derive decryption shares for any ciphertext. For each $i \in \{1, \ldots, n\}$, the verification key $\mathsf{VK}_i$ is used to check the validity of decryption shares generated using $\mathsf{SK}_i$.
- $\mathsf{Encrypt}(\mathsf{PK}, M)$ is a randomized algorithm that given a public key $\mathsf{PK}$ and a plaintext $M$ outputs a ciphertext $C$.
- $\mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$ on input of a public key $\mathsf{PK}$, a ciphertext $C$ and a private-key share $(i, \mathsf{SK}_i)$, this (possibly randomized) algorithm outputs either a decryption share $\mu_i = (i, \hat{\mu}_i)$, or a special symbol $(i, \perp)$.
- $\mathsf{ShareVerify}(\mathsf{PK}, \mathsf{VK}_i, C, \mu_i)$ takes as input $\mathsf{PK}$, the verification key $\mathsf{VK}_i$, a ciphertext $C$ and a purported decryption share $\mu_i = (i, \hat{\mu}_i)$. It outputs either `valid` or `invalid`. In the former case, $\mu_i$ is said to be a valid decryption share.
- $\mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, \{\mu_1, \ldots, \mu_t\})$ given $\mathsf{PK}$, $\mathbf{VK}$, $C$ and a set of $t$ decryption shares $\{\mu_1, \ldots, \mu_t\}$, this algorithm outputs either a plaintext $M$ or $\perp$ if the set contains invalid decryption shares.

It is required that the consistency of $\mathsf{PK}$ with $\mathbf{VK}$ be publicly checkable. Namely, for any $t$-subset $V$ of $\mathbf{VK}$, there must be an efficient algorithm[1], which we call $\mathsf{CheckKeys}$ in the upcoming sections, allowing to make sure that $V$ is a valid set of verification keys w.r.t. $\mathsf{PK}$.

CORRECTNESS. For any $(\mathsf{PK}, \mathbf{VK}, \mathbf{SK})$ generated by $\mathsf{Setup}(n, t, 1^k)$, it is required that

1. For any ciphertext $C$, if $\mu_i = \mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$, where $\mathsf{SK}_i$ is the $i^{\text{th}}$ private-key share in $\mathbf{SK}$, then $\mathsf{ShareVerify}(\mathsf{PK}, \mathsf{VK}_i, C, \mu_i) = \texttt{valid}$. We emphasize that this must hold even in the event that $\mu_i = (i, \perp)$ (i.e., if $C$ is deemed invalid).
2. If $C$ is the output of $\mathsf{Encrypt}(\mathsf{PK}, M)$ and $S = \{\mu_1, \ldots, \mu_t\}$ is a set of decryption shares such that $\mu_i = \mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$ for $t$ distinct private-key shares in $\mathbf{SK}$, then $\mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, S) = M$.

[1] Although such an algorithm is not formally required in [SG98, BBH06], it implicitly exists in all known robust TPKC and it is convenient to be considered here.

The security of robust TPKC is defined via two properties. The first one is the usual notion of chosen-ciphertext security for public key encryption adapted to the TPKC setting, while the other one is termed *consistency of decryptions*. For the formal security definitions we refer to [SG98].

## 3 Stronger Proof Soundness Definitions

We define our stronger version of proof soundness with adversarially chosen keys, as well as a notion similar to the binding property of commitments. Jumping ahead, we note that both strengthenings imply the original soundness definition but are themselves incomparable. The application usually determines which version should be considered. Arguably, they are both somewhat more realistic to use than Definition 2 in certain applications such as multiparty protocols, where parties might be able to cheat by maliciously generating their public key.

**Definition 3 (Strong Proof Soundness).** *Consider the following game between a challenger and an adversary $\mathcal{A}$:*

**Stage 1:** $\mathcal{A}(1^k)$ *outputs a public key* $pk$ *and a message* $m \in \mathcal{M}_{pk}$.
**Stage 2:** *The challenger computes* $C \leftarrow \mathsf{Enc}(pk, m)$ *and gives it to* $\mathcal{A}$*, which returns* $(m', \pi')$.

*$\mathcal{A}$'s advantage is defined as the probability*

$$\mathbf{Adv}^{\text{s-proof-snd}}_{\mathsf{PKENO},\mathcal{A}}(1^k) := \Pr\left[\mathsf{Ver}(pk, C, m', \pi') = 1 \wedge m' \neq m\right] .$$

*A PKENO scheme is* strongly proof sound *if any PPT adversary $\mathcal{A}$ has negligible advantage.*

An alternative strong notion of soundness (with adversarially chosen keys) follows the idea that, for any ciphertext, one can only find one valid message-proof pair. We call this the *committing* property:

**Definition 4 (Committing Property).** *A PKENO scheme is* strongly committing *if for any PPT adversary $\mathcal{A}$ that outputs $(pk, C, m, \pi, m', \pi')$ on input $1^k$ the following probability is negligible:*

$$\mathbf{Adv}^{\text{s-com}}_{\mathsf{PKENO},\mathcal{A}}(1^k) := \Pr\left[\mathsf{Ver}(pk, C, m, \pi) = 1 = \mathsf{Ver}(pk, C, m', \pi') \wedge m \neq m'\right] .$$

The following shows that Definitions 3 and 4 are actually achievable—by a practical scheme.

**Theorem 1.** *Galindo's PKENO scheme [Gal09] is strongly proof sound and strongly committing.*

The proof is deferred to the full version, where we compare the different notions of proof of soundness, showing that Definitions 3 and 4 are incomparable while both are strictly stronger than the original notion of proof soundness (Def. 2). Comparing the new notions in the "Knowledge of Secret Key" (KOSK) model,

where the adversary has to prove knowledge of the secret key, we further prove that the committing property is strictly stronger than strong soundness. We note that all our proofs preserve IND-CCPA security. As for the separation we further show that if there exists a proof-sound scheme which is also IND-CCPA, then there exists an IND-CCPA scheme which is *not* strongly committing (strongly proof sound, resp.) but still proof sound. It is also easy to see that the case of adversarially chosen keys is strictly stronger, independently of the question whether the PKENO scheme is IND-CCPA secure or not. These results are formally stated and proven in the full version.

## 4 On Generic Constructions for PKENO

In this section, we first show that an apparently straightforward PKENO construction (briefly) suggested in [DHKT08] fails to provide chosen-ciphertext security (as defined in that work). Next, we propose a simple and efficient transformation from robust TPKE to PKENO. Finally, we describe two concrete PKENO schemes obtained from this transformation. The first relies on the Decisional Diffie-Hellman assumption and the Random-Oracle Model, while the second relies on the Decisional Linear assumption and is proven secure in the standard model.

### 4.1 Witness-Recovering Encryption Does Not Suffice

In a PKEWR scheme, decryption recovers the random coins $r$ used to encrypt a ciphertext $C$. Damgård *et al.* [DHKT08] proposed to use $r$ as the opening proof for a PKENO scheme. Verification then proceeds by re-encrypting the plaintext $m$ as $C' = \mathsf{Enc}(pk_B, m; r)$, checking whether $C = C'$, and accepting/rejecting the proof accordingly. A subtle issue arises when dealing with invalid ciphertexts $C$, as in this case the random coins might simply not exist, for instance if $C$ is not in the range of the encryption algorithm. This could be exploited by an adversary to abuse the security of the resulting PKENO system. We illustrate this by sketching an IND-CCPA attack against the candidate PKENO scheme one would obtain from the IND-CCA secure encryption scheme[2] of Peikert and Waters [PW08].

Let $F(\cdot), G(\cdot,\cdot)$ be trapdoor functions that can be inverted knowing the corresponding secret keys $sk_F, sk_G$; let $h$ be a pairwise independent hash function, and let $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ be a strongly unforgeable one-time signature scheme [Mer89]. Then, the challenge ciphertext of plaintext $m_\beta$ in [PW08] is constructed as follows: choose a one-time key pair $(\mathsf{SSK}^\star, \mathsf{SVK}^\star) \leftarrow \mathcal{G}(1^k)$, choose $x^\star$ uniformly at random from a certain set of strings, and compute $C_0^\star = F(x^\star)$, $C_1^\star = G(\mathsf{SVK}^\star, x^\star)$, $C_2^\star = h(x^\star) \oplus m_\beta$, $\sigma^\star = \mathcal{S}(\mathsf{SSK}^\star, (C_0^\star, C_1^\star, C_2^\star))$. The ciphertext is then $C^\star = (\mathsf{SVK}^\star, C_0^\star, C_1^\star, C_2^\star, \sigma^\star)$.

[2] In this scheme, not all the sender's coins are retrieved upon decryption since the private key of the one-time signature is not recovered. However, these unrecovered coins have no impact in our setting.

We show how an IND-CCPA attacker can abuse the prove oracle in the IND-CCPA game to mount a successful distinguishing attack.

Given the challenge $C^\star$, the adversary chooses $(\mathsf{SSK}, \mathsf{SVK}) \leftarrow \mathcal{G}(1^k)$ and submits $C := \big(\mathsf{SVK}, C_0^\star, C_1^\star, C_2^\star, \mathcal{S}(\mathsf{SSK}, (C_0^\star, C_1^\star, C_2^\star))\big)$ to the prove oracle. $C$ is invalid since $\mathsf{SVK}$ used in $C$ is different from $\mathsf{SVK}^\star$ embedded in $C_1^\star$; but what could be a proof for this? In [PW08] one can decrypt by either inverting $C_0^\star = F(x^\star)$ or $C_1^\star = G(\mathsf{SVK}^\star, x^\star)$. Inverting $F(x^\star)$ and giving $x^\star$ out to the adversary would result in trivially recovering $m_\beta$; we are thus left with inverting $G(\mathsf{SVK}^\star, x^\star)$. Inversion of $G$ is done using both the secret key $sk_G$ and the 'tag' $\mathsf{SVK}$. This will result in a pre-image $x \neq x^\star$, and the question is whether the targeted $x^\star$ can be recovered from $x$ and the publicly available information. Alas, this property is not covered in the model by [PW08]. Indeed, for certain lossy-trapdoor functions $G(\cdot,\cdot)$ the knowledge of such a pre-image $x$ allows recovering $x^\star$. For instance, for the functions by Rosen and Segev [RS08], $x = (\mathsf{SVK} - \mathsf{SVK}^\star) \cdot x^\star$ with $\mathsf{SVK}, \mathsf{SVK}^\star, x, x^\star$ being integers in a ring, and therefore $x^\star$ can be trivially recovered. This results in a successful IND-CCPA attack.

One could wonder whether PKEWR schemes in the Random-Oracle Model could be of any help here. It is rather straightforward to prove that the PKENO obtained by using the randomness as a proof in the Fujisaki and Okamoto [FO99] encryption scheme suffers from a similar attack. Finding a practical generic construction for PKENO from a primitive weaker than identity-based encryption represents therefore an open problem.

### 4.2 Stronger Decryption-Consistency Definitions for TPKC

In our generic construction, we need somewhat stronger flavors of decryption consistency. In the first one, we require the adversary's advantage to remain negligible in an enhanced game where the challenger reveals $\mathsf{PK}$ and *all* decryption shares $\mathsf{SK}_1, \ldots, \mathsf{SK}_n$ in the setup phase.

**Definition 5 (Decryption Consistency with Known Secret Keys).** *Let us consider the following game between a challenger and an adversary $\mathcal{A}$:*

**Setup:** *The challenger runs* $\mathsf{Setup}(n, t, 1^k)$ *to obtain a triple* $(\mathsf{PK}, \mathbf{VK}, \mathbf{SK})$, *where* $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$, *and sends* $(\mathsf{PK}, \mathbf{VK}, \mathbf{SK})$ *to the adversary $\mathcal{A}$.*

**Output:** *$\mathcal{A}$ generates a ciphertext $C$ and two unequal sets $S = \{\mu_1, \ldots, \mu_t\}$ and $S' = \{\mu'_1, \ldots, \mu'_t\}$ of decryption shares.*

*Define $\mathcal{A}$'s advantage* $\mathsf{Adv}^{\text{s-dec-con}}_{\mathsf{TPKC},\mathcal{A}}(1^k)$ *as the probability that the following conditions hold:*

1. *All decryption shares in $S$ and $S'$ are valid decryption shares w.r.t. the verification key $\mathbf{VK}$ and the ciphertext $C$.*
2. *$S$ and $S'$ each contain decryption shares from $t$ distinct servers.*
3. $\mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, S) \neq \mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, S')$.

*A robust TPKC is* decryption consistent with known secret keys *if, for every PPT adversary $\mathcal{A}$, the advantage* $\mathsf{Adv}^{\text{s-dec-con}}_{\mathsf{TPKC},\mathcal{A}}(1^k)$ *is negligible.*

We further strengthen the definition and let the adversary choose the keys on her own.

**Definition 6 (Strong Decryption Consistency).** *A robust TPKC is* strongly decryption consistent *if for every PPT adversary $\mathcal{A}$ the advantage in a game that is similar to the one above is negligible, when $\mathcal{A}$ is allowed to generate consistent encryption/verification keys* $(\mathsf{PK}, \mathbf{VK})$ *on her own* without *having to publish the vector of decryption shares* $\mathbf{SK}$.

### 4.3 Robust TPKC Implies PKENO

Let $\mathsf{TPKC} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{ShareDecrypt}, \mathsf{ShareVerify}, \mathsf{Combine})$ be a robust threshold cryptosystem providing chosen-ciphertext security and strong decryption consistency. We turn it into a secure PKENO scheme as follows. We can essentially restrict ourselves to the case of a single-user threshold scheme, $t = n = 1$, but nonetheless state the transformation for general parameters. We use the threshold cryptosystem in a straightforward way to encrypt messages. To decrypt ciphertexts in our derived PKENO scheme, we first generate the decryption shares locally and then run the combiner to recover the message. The decryption shares also act as a soundness proof and the share verification determines the proof verification for PKENO. Chosen-ciphertext security of the threshold cryptosystem guarantees IND-CCPA security of the resulting PKENO scheme—using the fact that in the attack on the threshold cryptosystem the adversary can request to see decryption shares, which translates to access to a $\mathsf{Prove}$ oracle in the IND-CCPA game. Additionally, decryption consistency of the underlying threshold scheme provides soundness of the PKENO.

- $\mathsf{Gen}(1^k)$ Choose arbitrary integers $t, n \in \mathbb{N}$ such that $1 \leq t \leq n$ and run $\mathsf{Setup}(n, t, 1^k)$ to obtain $\big(\mathsf{PK}, \mathbf{VK} = (\mathsf{VK}_1, \ldots, \mathsf{VK}_n), \mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)\big)$. The key pair $(pk, sk)$ for $\mathsf{PKENO}$ is defined as $pk = (\mathsf{PK}, \mathbf{VK}, n, t)$, $sk = \mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$. The plaintext (resp. ciphertext) space of $\mathsf{PKENO}$ is the plaintext (resp. ciphertext) space of $\mathsf{TPKC}$.
- $\mathsf{Enc}(pk, M)$ To encrypt $M$, parse $pk$ as $pk = (\mathsf{PK}, \mathbf{VK}, n, t)$ and compute $C = \mathsf{Encrypt}(\mathsf{PK}, M)$.
- $\mathsf{Dec}(sk, C)$ To decrypt $C$, conduct the following steps:
  1. For $i = 1, \ldots, t$, compute $\mu_i = \mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$.
  2. If there exists $j \in \{1, \ldots, t\}$ such that $\mu_j = (j, \perp)$ return $\perp$.
  3. Otherwise return $M = \mathsf{Combine}(\mathsf{PK}, \mathsf{VK}, C, S)$, where $S = \{\mu_1, \ldots, \mu_t\}$ is a set of valid shares.
- $\mathsf{Prove}(sk, C)$ A proof for the ciphertext $C$ is computed by parsing $sk$ as $(\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$ and doing the following:
  1. For $i = 1, \ldots, t$, compute $\mu_i = \mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$.
  2. Return the set of decryption shares $\pi = \{\mu_1, \ldots, \mu_t\}$.

– $\mathsf{Ver}(pk, C, M, \pi)$ parse $pk$ as $(\mathsf{PK}, \mathbf{VK}, n, t)$ and $\pi$ as a set of shares $\{\mu_1, \ldots, \mu_t\}$.
  1. Return 0 if $\pi$ contains less than $t$ shares or if $(\mathsf{VK}_1, \ldots, \mathsf{VK}_t)$ is inconsistent with $\mathsf{PK}$ (namely, if $\mathsf{CheckKeys}(\mathbf{PK}, (\mathsf{VK}_1, \ldots, \mathsf{VK}_t)) = 0$).
  2. If there exists $j \in \{1, \ldots, t\}$ s.t. $\mathsf{ShareVerify}(\mathsf{PK}, \mathsf{VK}_j, C, \mu_j) = \texttt{invalid}$, return 0. Otherwise return 1 if $M = \mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, \{\mu_1, \ldots, \mu_t\})$ and 0 otherwise.

**Theorem 2.** *Robust TPKC satisfying decryption consistency with known secret keys (resp. strong decryption consistency) implies PKENO with proof soundness (resp. strongly committing).*

The statement of the above theorem is implied by the following lemmas:

**Lemma 1.** *The above generic PKENO system provides IND-CCPA security if the underlying robust TPKC is IND-TCCA secure.*

*Proof.* Let $\mathcal{A}$ be an IND-CCPA adversary against $\mathsf{PKENO}$. We show how it readily yields a chosen-ciphertext adversary $\mathcal{B}$ against the underlying $\mathsf{TPKC}$.

$\mathcal{B}$ starts by choosing $S = \{1, \ldots, t-1\}$ as the set of decryption servers to corrupt and obtains $(\mathsf{PK}, \mathbf{VK})$ as well as $((1, \mathsf{SK}_1), \ldots, (t-1, \mathsf{SK}_{t-1}))$ from her own challenger. The PKENO adversary $\mathcal{A}$ is supplied with the public key $pk = (\mathsf{PK}, \mathbf{VK}, n, t)$ and starts making decryption and proving queries. Whenever $\mathcal{A}$ queries a proof for some ciphertext $C$, $\mathcal{B}$ is able to compute $\mu_i = \mathsf{ShareDecrypt}(\mathsf{PK}, i, \mathsf{SK}_i, C)$ for $i = 1, \ldots, t-1$ since she knows $\mathsf{SK}_1, \ldots, \mathsf{SK}_{t-1}$. To obtain the missing decryption share, $\mathcal{B}$ asks her challenger to reveal $\mu_t = \mathsf{ShareDecrypt}(\mathsf{PK}, t, \mathsf{SK}_t, C)$, which allows constructing $\pi = \{\mu_1, \ldots, \mu_t\}$ as long as $\mathsf{TPKC}$ provides correctness. It is not hard to see that $\mathcal{A}$'s decryption queries can be dealt with exactly in the same way: instead of revealing the set $\{\mu_1, \ldots, \mu_t\}$, $\mathcal{B}$ returns the output of $\mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, \{\mu_1, \ldots, \mu_t\})$.

At the challenge step, $\mathcal{A}$ outputs equal-length messages $M_0, M_1$ that are transmitted to $\mathcal{B}$'s challenger. The latter replies with a challenge TPKC ciphertext $C^\star$, which $\mathcal{B}$ relays to $\mathcal{A}$. In the second stage, $\mathcal{A}$ is allowed to make further decryption/proof queries. Since these never involve the challenge ciphertext $C^\star$, $\mathcal{B}$ is always able to answer them by invoking her own challenger as in the first phase. The game ends with $\mathcal{A}$ outputting a bit $b \in \{0, 1\}$, which is also $\mathcal{B}$'s result. It is straightforward to observe that if $\mathcal{A}$ is successful then so is $\mathcal{B}$. □

**Lemma 2.** *The above generic PKENO scheme is sound (resp. strongly committing) if it builds on a robust TPKC satisfying decryption consistency with known secret keys (resp. strong decryption consistency).*

*Proof.* We first show that if an adversary $\mathcal{A}$ defeats the soundness of $\mathsf{PKENO}$ in the sense of Definition 2 then there exists an adversary $\mathcal{B}$ breaking the decryption consistency with known secret keys in $\mathsf{TPKC}$ with the same advantage.

Namely, our adversary $\mathcal{B}$ obtains $\mathsf{PK}$, $\mathbf{VK}$ and $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$ from her challenger. The weak-soundness adversary $\mathcal{A}$ then receives $pk = (\mathsf{PK}, \mathbf{VK}, n, t)$, $sk = \mathbf{SK}$. In Stage 1 of the game, $\mathcal{A}$ chooses a plaintext $m$ that $\mathcal{B}$ encrypts using the public key $\mathsf{PK}$ of $\mathsf{TPKC}$. Upon receiving the resulting ciphertext $C =$

$\mathsf{Encrypt}(\mathsf{PK}, m)$, $\mathcal{A}$ attempts to produce a pair $(m', \pi')$ such that $\mathsf{Ver}(pk, C, m', \pi') = 1$ and $m' \neq m$. Since $\pi'$ is a valid proof, it can necessarily be parsed as a set $\{\mu'_1, \ldots, \mu'_t\}$ of valid decryption shares. The correctness property of TPKC implies that, since $\mathcal{B}$ knows $\mathbf{SK} = (\mathsf{SK}_1, \ldots, \mathsf{SK}_n)$, it must be able to generate another set $\pi = \{\mu_1, \ldots, \mu_t\}$ of decryption shares such that

$$m = \mathsf{Combine}(\mathsf{PK}, \mathbf{VK}, C, \{\mu_1, \ldots, \mu_t\}) \ .$$

It follows that the sets $\pi$ and $\pi'$ are valid $t$-sets of decryption shares that break the decryption consistency with known secret keys of TPKC.

Proving that the strong decryption consistency of TPKC implies the strong committing property of PKENO is fairly straightforward: from a strong-committingness adversary $\mathcal{A}$, we immediately obtain a strong-decryption-consistency adversary $\mathcal{B}$ that returns whatever $\mathcal{A}$ outputs. □

Since in the KOSK model any strongly committing PKENO scheme is also strongly proof sound, Lemma 2 implies that a strongly proof-sound PKENO scheme can be obtained from a strongly decryption-consistent TPKC. In general, however, it seems that strong decryption consistency is not sufficient to imply strong proof soundness as well.

It turns out that for concrete TPKC constructions, such as the schemes by Shoup and Gennaro [SG98] and Arita and Tsurudome [AT09], it is possible to set $n = t = 1$ for improved efficiency. For instance, the consistency check between PK and $(\mathsf{VK}_1, \ldots, \mathsf{VK}_t)$ becomes trivial in Step 1 of the verification algorithm. We recall those TPKC in the full paper and describe the resulting efficient PKENO schemes in the next section.

*Remark 1.* The reader might wonder whether an efficient transformation from PKENO to robust non-interactive threshold cryptosystem exists. The answer is in the affirmative if we allow[3] these primitives to support *labels* [Sho04]. A label is an arbitrary string that is given as additional input to every algorithm of the PKENO and TPKC primitives, except the key generation algorithms. Then, the transformations from standard PKE to (non-robust) non-interactive threshold cryptosystem by Dodis and Katz [DK05] yield robust TPKC when replacing PKE by PKENO. Due to space limitations, we omit the details here but they follow easily from [DK05, Section 4.2].

## 5 New PKENO Constructions Implied by TPKC

This section describes new concrete schemes obtained from the transformation in Section 4.3.

[3] The reason of this restriction is the difficulty of *efficiently* constructing a PKENO system supporting labels from an ordinary PKENO. The standard black-box technique to include labels (by simply appending them to the plaintext upon encryption) in any public key encryption scheme fails to preserve security (in the sense of Definition 1) in the context of PKENO.

### 5.1 PKENO without Pairings in the Random-Oracle Model

In [SG98], Shoup and Gennaro described two CCA2-secure threshold cryptosystems in the random-oracle model. We show in the full version of the paper that the most efficient scheme TDH2 satisfies strong decryption consistency (although a weaker notion of consistency was considered in [SG98]). This scheme makes use of a prime-order group $\mathbb{G}$ where the Decision Diffie-Hellman problem[4] is assumed to be hard. It is easily seen to give rise to the following PKENO system.

- $\mathsf{Gen}(1^k)$: chooses a group $\mathbb{G}$ of prime order $p > 2^k$, $g, \bar{g} \xleftarrow{\$} \mathbb{G}$ and $x \xleftarrow{\$} \mathbb{Z}_p$, and sets $h = g^x$. The public key $pk$ includes $g, h, \bar{g}$, the description of the plaintext space $\mathcal{M}_{pk} = \{0,1\}^l$, where $l$ depends polynomially on $k$, and hash functions $H_0 : \mathbb{G} \to \{0,1\}^l$, $H_1, H_2 : \{0,1\}^* \to \mathbb{Z}_p$ (to be modeled as random oracles). The secret key $sk = x$.
- $\mathsf{Enc}(pk, m)$: to encrypt a message $m \in \{0,1\}^l$, it proceeds as follows. It chooses $r, s \xleftarrow{\$} \mathbb{Z}_p$, it sets $K = h^r$ and computes

$$c = H_0(h^r) \oplus m,\ u = g^r,\ w_1 = g^s,\ \bar{u} = \bar{g}^r,\ \overline{w}_1 = \bar{g}^s,\ f_1 = s + re_1,$$

  where $e_1 = H_1(c, u, w_1, \bar{u}, \overline{w}_1)$. Let us note that $(w_1, \overline{w}_1, f_1)$ constitutes a non-interactive zero-knowledge proof of equality of discrete logarithms $\log_g u = \log_{\bar{g}} \bar{u}$ [CP92]. The ciphertext is $C = (c, u, \bar{u}, e_1, f_1)$.
- $\mathsf{Dec}(sk, C)$: given $sk = x$ and $C = (c, u, \bar{u}, e_1, f_1)$, the decryption algorithm first checks whether $e_1 = H_1(c, u, w_1, \bar{u}, \overline{w}_1)$, where $w_1 = g^{f_1}/u^{e_1}$, $\overline{w}_1 = \bar{g}^{f_1}/\bar{u}^{e_1}$. If this is not the case, it returns $\perp$, meaning that $C$ is invalid. Otherwise, it returns $m = c \oplus H_0(u^x)$.
- $\mathsf{Prove}(sk, C)$: given $C = (c, u, \bar{u}, e_1, f_1)$ and the secret key $sk = x$, the algorithm first checks if $e_1 = H_1(c, u, w_1, \bar{u}, \overline{w}_1)$, where $w_1 = g^{f_1}/u^{e_1}$, $\overline{w}_1 = \bar{g}^{f_1}/\bar{u}^{e_1}$. If this is not satisfied, it returns $\emptyset$, meaning that the ciphertext is invalid. Otherwise it computes $K = u^x$, chooses $s \leftarrow \mathbb{Z}_p$ and returns $\pi = (K, e_2, f_2)$, where

$$w_2 = g^s,\ \overline{w}_2 = u^s,\ e_2 = H_2(K, w_2, \overline{w}_2),\ f_2 = s + xe_2\ .$$

  Note that $(w_2, \overline{w}_2, f_2)$ constitutes a non-interactive zero-knowledge proof of equality of discrete logarithms $\log_g h = \log_u K$.
- $\mathsf{Ver}(pk, c, m, \pi)$: parses $C$ as $(c, u, \bar{u}, e_1, f_1)$ and $\pi$ as $(K, e_2, f_2)$. Then it performs the following tests:
  1. $e_1 \stackrel{?}{=} H_1(c, u, w_1, \bar{u}, \overline{w}_1)$, where $w_1 = g^{f_1}/u^{e_1}$, $\overline{w}_1 = \bar{g}^{f_1}/\bar{u}^{e_1}$
  2. $e_2 \stackrel{?}{=} H_2(K, w_2, \overline{w}_2)$, where $w_2 = g^{f_2}/h^{e_2}$, $\overline{w}_2 = u^{f_2}/K^{e_2}$

  If these tests are both correct and $c \oplus H_0(K) = m$ it returns 1; and 0 otherwise. If Test 1 fails, it outputs 1 iff $\pi = \emptyset$ and $m = \perp$. In any other case (e.g., Test 2 fails or can not be computed because $\pi = \emptyset$) it outputs 0.

[4] A slightly less efficient threshold cryptosystem described in [SG98] relies on the Computational Diffie-Hellman assumption (in the random oracle model) and can be turned into a PKENO system in the same way.

Since the underlying threshold cryptosystem is IND-TCCA secure and strongly decryption consistent, it follows that the above PKENO is IND-CCPA secure and strongly committing.

### 5.2 PKENO Based on the Decision Linear Assumption

Recently, Arita and Tsurudome [AT09] described an efficient way to thresholdize the decryption algorithm of Kiltz's tag-based encryption scheme [Kil06] using bilinear maps to achieve robustness. Their scheme readily yields another PKENO with strong soundness since it also provides decryption consistency in the strongest sense. The security proof of the resulting scheme is in the standard model under the Decision Linear assumption [BBS04] (in bilinear groups), which is the infeasibility of distinguishing $g^{c+d}$ from random given $(g, g^a, g^b, g^{ac}, g^{bd})$, where $a, b, c, d \xleftarrow{\$} \mathbb{Z}_p$.

One of the advantages of this PKENO scheme is that it can be used in CCA2-anonymous group signatures that rely on the *linear encryption* technique [BBS04]. For instance, it can be used to obtain simpler and more efficient proofs of correct opening (as required by the model of Bellare *et al.* [BSZ05] in the context of dynamic groups) in Groth's fully anonymous group signatures [Gro07]: such a proof only consists of two group elements and its verification only entails two pairing evaluations, which is significantly cheaper than checking a pairing-based non-interactive witness indistinguishable proof as in [Gro07].

The description hereafter requires a strongly unforgeable [Mer89, ADR02] one-time signature scheme $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ as in the original CHK transformation [CHK04], where we assume for simplicity that the scheme's verification keys $\mathsf{SVK}$ can be embedded in $\mathbb{Z}_p$ (else one should first hash the key with a target-collision resistant hash function). We note that shorter ciphertexts can be obtained using Waters' technique [Wat05] in the same way as in the encryption scheme of [BMW05, Section 3.1]: at the expense of longer public keys (comprising $O(k)$ group elements), ciphertext components $\mathsf{SVK}$ and $\sigma$ can be eliminated.

- $\mathsf{Gen}(1^k)$: chooses groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^k$ that are equipped with a bilinear map $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, $g \xleftarrow{\$} \mathbb{G}$ and $x, y, u, v \xleftarrow{\$} \mathbb{Z}_p$. The public key $pk$ comprises $(X, Y, U, V) = (g^x, g^y, g^u, g^v)$, the description of the plaintext space $\mathcal{M}_{pk} = \mathbb{G}$ and that of a strong one-time signature $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$. The secret key is $sk = (x, y, u, v)$.
- $\mathsf{Enc}(pk, m)$: to encrypt a message $m \in \mathbb{G}$, the algorithm first generates a one-time signature key pair $(\mathsf{SSK}, \mathsf{SVK}) \leftarrow \mathcal{G}(1^k)$. It chooses $r, s \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$C_1 = X^r,\ C_2 = Y^s,\ D_1 = (g^{\mathsf{SVK}} U)^r,\ D_2 = (g^{\mathsf{SVK}} V)^s,\ E = m \cdot g^{r+s},$$

  and $\sigma = \mathcal{S}(\mathsf{SSK}, (C_1, C_2, D_1, D_2, E))$. The ciphertext is $C = (\mathsf{SVK}, C_1, C_2, D_1, D_2, E, \sigma)$.

- $\mathsf{Dec}(sk, C)$: given $sk = (x, y, u, v)$ and $C = (\mathsf{SVK}, C_1, C_2, D_1, D_2, E, \sigma)$, the algorithm checks if $\mathcal{V}(\mathsf{SVK}, \sigma, (C_1, C_2, D_1, D_2, E)) = 1$, $D_1 = C_1^{(\mathsf{SVK}+u)/x}$ and $D_2 = C_2^{(\mathsf{SVK}+v)/y}$. If these checks fail, it returns $\perp$. Otherwise, it outputs $m = E \cdot C_1^{-1/x} \cdot C_2^{-1/y}$.
- $\mathsf{Prove}(sk, C)$: given $C = (\mathsf{SVK}, C_1, C_2, D_1, D_2, E, \sigma)$ and $sk = (x, y, u, v)$, the algorithm returns $\emptyset$ if $\mathcal{V}(\mathsf{SVK}, \sigma, (C_1, C_2, D_1, D_2, E)) = 0$ or if $D_1 \neq C_1^{(\mathsf{SVK}+u)/x}$ or $D_2 \neq C_2^{(\mathsf{SVK}+v)/y}$. Otherwise it computes and returns $\pi = (\pi_1, \pi_2) = (C_1^{1/x}, C_2^{1/y})$.
- $\mathsf{Ver}(pk, c, m, \pi)$: parses $C$ as $(\mathsf{SVK}, C_1, C_2, D_1, D_2, E, \sigma)$ and $\pi$ as $(\pi_1, \pi_2) \in \mathbb{G}^2$ (and outputs 0 if they cannot be parsed properly). Then, it performs the following tests:

  1. $\mathcal{V}(\mathsf{SVK}, \sigma, (C_1, C_2, D_1, D_2, E)) \stackrel{?}{=} 1$, $e(C_1, g^{\mathsf{SVK}} U) \stackrel{?}{=} e(X, D_1)$, $e(C_2, g^{\mathsf{SVK}} V) \stackrel{?}{=} e(Y, D_2)$.
  2. $e(\pi_1, X) \stackrel{?}{=} e(g, C_1)$, $e(\pi_2, Y) \stackrel{?}{=} e(g, C_2)$, $E \stackrel{?}{=} m \cdot \pi_1 \cdot \pi_2$.

  If both tests are correct, it returns 1. If Test 1 fails, it outputs 1 iff $\pi = \emptyset$ and $m = \perp$. In any other situation, it outputs 0.

In comparison with [Gal09] (if we assume that CCA2-security is acquired using the technique of [BMW05, Section 3.1] in both schemes), the above system provides faster decryption (since no pairing evaluation is needed) at the expense of longer ciphertexts whereas proofs are equally expensive to verify. Its main advantage over [Gal09], in our opinion, lies in its possible use to provide simple proofs of correct opening in pairing-based group signatures.

It is also worth mentioning that other cryptosystems [Kil07, Boy07] also admit CCA2-secure threshold variants which can be proved strongly decryption consistent. They thus imply strongly committing PKENO instances bearing similarities with the above scheme. The Paillier-based TPKE scheme of [FP01] can be proved decryption consistent in the known secret key setting (cf. Definition 5). Proving it strongly decryption consistent seems harder.

## Acknowledgments

The work described in this paper was supported in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II, by the Emmy Noether Program Fi 940/2-1 of the German Research Foundation (DFG), and through the Center for Advanced Security Research Darmstadt (`www.cased.de`). The second author acknowledges the financial support of the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.). The fourth author is supported by the French ANR 07-TCOM-013-04 PACE Project and EADS.

## References


[AB09] Agrawal, S., Boyen, X.: Identity-based encryption from lattices in the standard model (July 2009) (manuscript)

[ADR02] An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 83–107. Springer, Heidelberg (2002)

[AT09] Arita, S., Tsurudome, K.: Construction of threshold public-key encryptions through tag-based encryptions. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 186–200. Springer, Heidelberg (2009)

[BBH06] Boneh, D., Boyen, X., Halevi, S.: Chosen ciphertext secure public key threshold encryption without random oracles. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 226–243. Springer, Heidelberg (2006)

[BBS04] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

[BGH07] Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), pp. 647–657 (2007)

[BMW05] Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM Conference on Computer and Communications Security 2005, pp. 320–329 (2005)

[Boy07] Boyen, X.: Miniature cca2 pk encryption: Tight security without redundancy. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 485–501. Springer, Heidelberg (2007)

[BSZ05] Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)

[CHK04] Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)

[CHK09] Cash, D., Hofheinz, D., Kiltz, E.: How to delegate a lattice basis. Cryptology ePrint Archive: Report 2009/351 (July 2009)

[CP92] Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)

[CvH91] Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)

[DHKT08] Damgård, I., Hofheinz, D., Kiltz, E., Thorbek, R.: Public-key encryption with non-interactive opening. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 239–255. Springer, Heidelberg (2008)

[DK05] Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 188–209. Springer, Heidelberg (2005)

[DT07] Damgård, I., Thorbek, R.: Non-interactive proofs for integer multiplication. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 412–429. Springer, Heidelberg (2007)

[FO99] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)

[FP01] Fouque, P.-A., Pointcheval, D.: Threshold cryptosystems secure against chosen-ciphertext attacks. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 351–368. Springer, Heidelberg (2001)

[Gal09] Galindo, D.: Breaking and repairing Damgård et al. public key encryption scheme with non-interactive opening. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 389–398. Springer, Heidelberg (2009)

[Gol01] Goldreich, O.: Foundations of Cryptography - Basic Tools. Cambridge University Press, Cambridge (2001)

[GPV08] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: 40th Annual ACM Symposium on Theory of Computing (STOC 2008), pp. 197–206 (2008)

[Gro07] Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)

[Kil06] Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 581–600. Springer, Heidelberg (2006)

[Kil07] Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on gap hashed diffie-hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)

[Mer89] Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)

[NY90] Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attack. In: Proc. of the Twenty-Second Annual ACM Symposium on Theory of Computing, pp. 427–437. ACM, New York (1990)

[Pei09] Peikert, C.: Bonsai trees (or, arboriculture in lattice-based cryptography). Cryptology ePrint Archive: Report 2009/359 (July 2009)

[PW08] Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC, pp. 187–196. ACM, New York (2008)

[RS08] Rosen, A., Segev, G.: Efficient lossy trapdoor functions based on the composite residuosity assumption. Cryptology ePrint Archive, Report 2008/134 (2008), http://eprint.iacr.org/

[Sah99] Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th Annual Symposium on Foundations of Computer Science, pp. 543–553. IEEE Computer Society Press, Los Alamitos (1999)

[SG98] Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)

[Sho04] Shoup, V.: Draft ISO/IEC 18033-2: An emerging standard for public-key encryption. Technical report, ISO/IEC (2004), http://www.shoup.net/iso

[Wat05] Waters, B.: Efficient identity-based encryption without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

# Flexible Group Key Exchange with On-demand Computation of Subgroup Keys

Michel Abdalla[1], Céline Chevalier[2], Mark Manulis[3], and David Pointcheval[1]

[1] École Normale Supérieure, CNRS-INRIA, Paris, France
[2] Telecom ParisTech, Paris, France
[3] Cryptographic Protocols Group
Department of Computer Science, TU Darmstadt & CASED, Germany
mark@manulis.eu

**Abstract.** Modern multi-user communication systems, including popular instant messaging tools, social network platforms, and cooperative-work applications, offer flexible forms of communication and exchange of data. At any time point concurrent communication sessions involving different subsets of users can be invoked. The traditional tool for achieving security in a multi-party communication environment are group key exchange (GKE) protocols that provide participants with a secure group key for their subsequent communication. Yet, in communication scenarios where various user subsets may be involved in different sessions the deployment of classical GKE protocols has clear performance and scalability limitations as each new session should be preceded by a separate execution of the protocol. The motivation of this work is to study the possibility of designing more flexible GKE protocols allowing not only the computation of a group key for some initial set of users but also efficient derivation of independent secret keys for all potential subsets. In particular we improve and generalize the recently introduced GKE protocols enabling on-demand derivation of peer-to-peer keys (so called GKE+P protocols). We show how a group of users can agree on a secret group key while obtaining some additional information that they can use on-demand to efficiently compute independent secret keys for *any* possible subgroup. Our security analysis relies on the Gap Diffie-Hellman assumption and uses random oracles.

## 1 Introduction

Despite more than 20 years of research (see surveys in [6, 29]), group key exchange (GKE) protocols are still far from being widely used in practice, especially if one compares to two-party key (2KE) exchange protocols which found their deployment in various standards and daily applications. Among the main reasons for this limited spectrum of applications is the fact that GKE protocols are rather complex, costly to implement, and not versatile enough. Modern communication platforms, including diverse instant-messaging tools and collaborative applications, allow their users to communicate and exchange data within almost any subset of participants. Had classical GKE protocols been deployed

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 351–368, 2010.

in this multi-user environment, then every new communication attempt would require a different execution of the GKE protocol. This constraint is anchored in the design rationale behind existing GKE protocols. A possible improvement would be to design more flexible GKE protocols, which would not only provide their participants with the group key but would also leave space for the efficient computation of secret keys for use within any subset of the original group.

Recently, Manulis [28] suggested flexible GKE protocols allowing for a secure combination of two communication forms: secure communication within a group and secure communication amongst any two parties from the group. In particular, he designed GKE protocols enabling on-demand derivation of peer-to-peer (p2p) keys, denoted GKE+P, which provide any pair of participants with an independent secure p2p key in addition to the common group key. The main ingredient of GKE+P constructions in [28] is the parallel execution of the classical Diffie-Hellman key exchange (PDHKE) protocol [18] and, in particular, the user's ability to re-use the same value $g^x$ in the computation of group and p2p keys, where $g$ is a generator and $x$ is the private user's exponent.

Building on this, we investigate the even more generalized and flexible approach — the extension of GKE protocols with the ability to compute an independent session key for any possible *subgroup* of the initial GKE participants. Similar to [28], we are interested in solutions which would allow the derivation of the subgroup keys in a more efficient way than simply running an independent session of a GKE protocol for each subgroup. GKE protocols enriched in this way, which we denote GKE+S, would allow the combination of different forms of secure communication. For example, a single file deposited in a file sharing network or broadcasted to the group may contain documents encrypted for the whole group and different attachments encrypted for different subgroups.

The required independence between different key types imposes further security challenges on GKE+S protocols: The classical GKE security requirement concerning the secrecy of the group key with respect to the external parties (typically called AKE-security [9,23]) should now be preserved even if some subgroup keys leak, and the independence of any subgroup key implicitly requires us to handle (collusion) attacks from parties that are external to that subgroup but internal to the preliminary computation process of the common group key or to a different subgroup with membership overlap. As a result, the specification of the adequate security requirements for GKE+S protocols with respect to these threats appears to be an interesting task from a formal point of view.

The protocols being considered in this paper are all based on the well-studied Burmester-Desmedt (BD) group key exchange protocol [14]. Interestingly, Manulis [28] had shown that, if users try to use the same exponents in the computation of group and p2p keys in the original BD protocol, then the AKE-security of p2p keys could no longer be guaranteed. In this work, we illustrate how a small modification of the original BD protocol suffices to obtain a secure GKE+P protocol. Interestingly, this modification of the original BD protocol only applies to the computation steps and leaves the communication complexity unchanged. In particular, our GKE+P protocol has better overall complexity than those

proposed in [28]. After presenting our new GKE+P protocol, we show how to extend it into a secure GKE+S protocol. Our GKE+S protocol requires only one additional communication round for setting up any subgroup key as opposed to the two-round communication that one would need to compute a subgroup key via an independent protocol execution.

### 1.1 Related Work

Security of key exchange protocols is usually defined with the requirement of Authenticated Key Exchange (AKE) security; see [4,15,16,27] for the 2KE case and [9,11,19,22,23] for the GKE case. AKE-security models the indistinguishability of the established session group key with respect to an active adversary treated as an external entity from the perspective of the attacked session. The signature-based compilation technique by Katz and Yung [23] (see also the work by Bresson, Manulis, and Schwenk [13]) can be used to achieve AKE-security for GKE protocols that already provide such indistinguishability but with regard to the passive attacks only. Besides AKE-security some security models for GKE protocols (e.g. [11,12,19,22]) define optional security against insider attacks.

The notion of GKE+P protocols has been put forth by Manulis [28]. He showed how to compile the so-called family of *Group Diffie-Hellman* protocols, i.e. protocols such as [20,32,14,33,25,26,31,17] which extend the classical Diffie-Hellman method to the group setting, in such a way, that at the end of the protocol any pair of users can derive their own p2p key on-demand and without subsequent communication. The main building block of his GKE+P compiler is the parallel Diffie-Hellman key exchange (PDHKE) in which each user broadcasts a value of the form $g^x$ and uses $x$ for the derivation of different p2p keys. For the two efficient two-round unauthenticated GKE protocols by Burmester and Desmedt (BD) [14] and by Kim, Perrig, and Tsudik (KPT) [25], in which users need to broadcast $g^x$ anyway, Manulis analyzed optimizations based on the reuse of the exponent $x$ for the computation of both group and p2p keys showing that KPT remains secure whereas BD not.

We also notice that PDHKE has been used by Jeong and Lee [21] for the simultaneous computation of multiple two-party keys amongst a set of users, yet without considering collusion attacks against the secrecy of keys computed by non-colluding users and without considering group keys. In another work, Biswas [5] proposed a slight modification to the original Diffie-Hellman protocol allowing its participants to obtain up to 15 different shared two-party keys.

While GKE+P protocols take the top-down approach in the sense that the computation of p2p keys for any pair of users is seen as a feature of the GKE protocol there have been several suggestions, e.g. [30,1,34], that construct GKE protocols from 2KE protocols used as a building block. For example, Abdalla et al. [1], as well as Wu and Zhu [34] order protocol participants into a cycle (in a BD fashion) and a 2KE protocol is executed only between the neighbors. However, these constructions do not explicitly meet the requirements of GKE+P protocols as an independent p2p key may not be available for every pair of users.

### 1.2 Contributions and Organization

In Section 2, we recall the classical Burmester-Desmedt (BD) protocol [14] and explain problems behind the re-use of its exponents for PDHKE following the analysis by Manulis [28]. In Section 3, we propose a slight modification to the computations steps of the BD protocol such that the overall communication complexity remains unchanged, yet secure, non-interactive derivation of p2p keys through the re-use of users' exponents becomes possible. Our GKE+P protocol obtained in this way and denoted mBD+P is more efficient than other solutions proposed in [28].

We then continue in Section 4 with our second contribution — the description of a GKE+S protocol in which users compute the common group key first and then any subgroup of these users can agree on an independent subgroup key. Our GKE+S protocol, denoted mBD+S, requires two-rounds for the computation of the group key and only one round for the subsequent on-demand computation of any subgroup key. In comparison to the naive approach of computing a subgroup key through an independent execution of the full GKE protocol our solution is more scalable as it halves the number of communication rounds and required messages.

In Section 5, we formally model the security of GKE+S protocols. We address AKE-security of both group and subgroup keys whereby security of the latter is modeled under consideration of possible collusion attacks representing the main challenge is such protocols. Our model generalizes the security model by Manulis [28], which considered only derivation of p2p keys.

Finally, we apply the model in Section 6 to prove that mBD+P provides AKE-security for group and p2p keys and that mBD+S additionally provides AKE-security for the derived subgroup keys. We conclude our work in Section 7 by comparing the communication and computation complexity of our protocols with the previous solutions showing the expected efficiency gain.

## 2 Preliminaries and Background

### 2.1 Notations and Assumptions

Throughout the paper, unless otherwise specified, by $\mathbb{G} := \langle g \rangle$ we denote a cyclic group of prime order $Q$ generated by $g$. By $\mathtt{H_g}, \mathtt{H_p}, \mathtt{H_s} : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ we denote three cryptographic hash functions, which will be used to derive group, p2p, and subgroup keys, respectively, and by $\mathtt{H} : \mathbb{G} \mapsto \{0,1\}^\kappa$ an auxiliary hash function which will be used to slightly modify the computations of the original Burmester-Desmedt protocol. The symbol "|" will be used for the concatenation of bit-strings. Since we are interested in authenticated protocols we will further use a digital signature scheme $\Sigma = (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify})$, which we assume to be existentially unforgeable under chosen message attacks (EUF-CMA). Additionally, we recall the following well-known cryptographic assumption (see e.g. [7]):

**Definition 1.** *Let* $\mathbb{G} := \langle g \rangle$ *as above and* $a, b, c \in_R \mathbb{Z}_Q$. *The* Gap Diffie-Hellman (GDH) *problem in* $\mathbb{G}$ *is hard if the following success probability is negligible:*

$$\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa) := \max_{\mathcal{A}'} \Big| \Pr_{a,b} \big[\mathcal{A}'^{\mathcal{D}(\cdot)}(g, g^a, g^b) = g^{ab}\big] \Big|,$$

*where* $\mathcal{D}(\cdot)$ *denotes the* Decision Diffie-Hellman (DDH) *oracle, i.e.* $\mathcal{D}(g, g^a, g^b, g^c)$ *outputs 1 if* $c = ab$ *and 0 otherwise.*

*Note that* $\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa)$ *is computed over all PPT adversaries* $\mathcal{A}'$ *running within time* $\kappa$. *In other words, the* Computation Diffie-Hellman (CDH) *problem in* $\mathbb{G}$ *is assumed to be hard even if the DDH problem in* $\mathbb{G}$ *is easy.*

### 2.2 Burmester-Desmedt Group Key Exchange

The Burmester-Desmedt (BD) protocol from [14] is one of the best known unauthenticated GKE protocols. Its technique has influenced many GKE protocols, including [24,2]. The BD protocol arranges participants $U_1, \ldots, U_n$ into a cycle, and requires two communication rounds:

**Round 1.** Each $U_i$ broadcasts $y_i := g^{x_i}$ for some random $x_i \in_R \mathbb{Z}_Q$.
**Round 2.** Each $U_i$ broadcasts $z_i := (y_{i+1}/y_{i-1})^{x_i}$ (the indices $i$ form a cycle, i.e. $0 = n$ and $n + 1 = 1$).
**Group Key Computation.** Each $U_i$ computes the secret group element

$$k'_i := (y_{i-1})^{nx_i} \cdot z_i^{n-1} \cdot z_{i+1}^{n-2} \cdots z_{i+n-2} = g^{x_1x_2 + x_2x_3 + \ldots + x_nx_1},$$

which is then used to derive the group key.

As shown by Katz and Yung [23] the group element $k'_i$ remains indistinguishable from a randomly chosen element in $\mathbb{G}$ under the DDH assumption with regard to a passive adversary; using the general authentication technique from [23] this indistinguishability can be extended to resist active attacks.

### 2.3 Parallel Diffie-Hellman Key Exchange

The following one-round parallelized Diffie-Hellman protocol (PDHKE) with the additional key derivation step based on a hash function $\mathtt{H_p}$ (modeled as a random oracle) has been deployed by Manulis [28] for the computation of independent p2p keys between any two users $U_i$ and $U_j$:

**Round 1.** Each $U_i$ chooses a random $x_i \in_R \mathbb{Z}_Q$ and broadcasts $y_i := g^{x_i}$.
**P2P Key Computation.** Each $U_i$ proceeds as follows:

- for the input identity $U_j$ compute $k'_{i,j} := y_j^{x_i} = g^{x_i x_j}$,
- derive $k_{i,j} := \mathtt{H_p}(k'_{i,j}, U_i | y_i, U_j | y_j)$.

(W.l.o.g. we assume that both users $U_i$ and $U_j$ use the same order for the inputs to $\mathtt{H_p}$.)

As motivated in [28] the input $(U_i|y_i, U_j|y_j)$ to $\mathtt{H_p}$ in addition to $k'_{i,j} = g^{x_i x_j}$ is necessary in order to ensure the independence of p2p keys in the presence of collusion attacks. This independence follows from the uniqueness of user identities and the negligible probability that an honest user $U_i$ chooses the same exponent $x_i$ in two different sessions. It has been shown in [28] that PDHKE provides security of p2p keys in the presence of passive attacks and that the authentication technique from [23] is also sufficient to preserve this property in the presence of active attacks.

### 2.4 Insecure Merge of BD and PDHKE

Given BD and PDHKE one of the questions investigated by Manulis [28] was whether a user $U_i$ can safely *re-use* own exponent $x_i$ for the computation of the group key and any p2p key. In other words whether a two-round merged version of original BD and PDHKE would result in a secure GKE+P protocol in which the keys are computed as follows: the group key $k_i := \mathtt{H_g}(k'_i, U_1|y_1, \ldots, U_n|y_n)$ and any p2p key $k_{i,j} := \mathtt{H_s}(k'_{i,j}, U_i|y_i, U_j|y_j)$.

The analysis given in [28] shows that this merge is insecure. More precisely, for any group size $n \geq 3$ an attack (possibly by colluding participants) was presented that would break the AKE-security of any p2p key $k_{i,j}$.

In the next section we revise this result. We show how to slightly modify the computation of $z_i$ in the original BD protocol in order to allow secure computation of $k_i := \mathtt{H_g}(k'_i, U_1|y_1, \ldots, U_n|y_n)$ and $k_{i,j} := \mathtt{H_s}(k'_{i,j}, U_i|y_i, U_j|y_j)$. We stress that our changes do not increase the original communication complexity of the BD protocol which is the actual goal for considering its merge with PDHKE. Then, based on our new construction we show how to obtain a secure GKE+S protocol where the communication effort for the derivation of subgroup keys requires only one round; this in contrast to two rounds that would be necessary to establish a subgroup key using the original BD protocol from the scratch.

## 3 GKE+P Protocol from Modified BD and PDHKE

In order to obtain a secure merge of BD and PDHKE we make use of the following trick in the computation of $z_i$: Instead of computing $z_i := (y_{i+1}/y_{i-1})^{x_i}$ we let each $U_i$ first compute $k'_{i,i+1} := y_{i+1}^{x_i}$ and $k'_{i-1,i} := y_{i-1}^{x_i}$ and then compute $z_i$ as an XOR sum $\mathtt{H}(k'_{i-1,i}) \oplus \mathtt{H}(k'_{i,i+1})$. This does not introduce new communication costs to the BD protocol but has impact on the computation of the group key. We observe that similar trick has been applied for a different purpose by Kim, Lee, and Lee [24], who considered possible extensions of BD-like protocols to handle dynamic membership events such as join and leave or to speed up the computation process, whereas here we use the trick exclusively to achieve independence between the group and p2p keys.

The actual description of the protocol which we denote mBD+P follows. Since we are interested in AKE-secure constructions we describe the necessary authentication steps as well. For this we assume that each $U_i$ is in possession of a long-lived key pair $(sk_i, pk_i) \leftarrow \mathtt{KeyGen}(1^\kappa)$ for the EUF-CMA secure digital signature

scheme $\Sigma = (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify})$. The authentication procedure is similar to the general authentication technique from [23], except that we construct the session id using elements from $\mathbb{G}$ as nonces; thus saving the communication costs. The protocol proceeds in two stages: the group stage involves all members of the group and results in the computation of the common group key, whereas the p2p stage is executed on-demand by any two group members wishing to compute an independent p2p key.

*Group Stage.* Let the group be defined by $\mathtt{pid} = (U_1, \ldots, U_n)$. In the following description we assume that user indices form a cycle such that $U_0 = U_n$ and $U_{n+1} = U_1$.

**Round 1.** Each $U_i$ computes $y_i := g^{x_i}$ for some random $x_i \in_R \mathbb{Z}_Q$ and broadcasts $(U_i, y_i)$.

**Round 2.** Each $U_i$ proceeds as follows:

- compute $\mathtt{sid}_i := (U_1|y_1, \ldots, U_n|y_n)$,
- $k'_{i-1,i} := y_{i-1}^{x_i}$ and $k'_{i,i+1} := y_{i+1}^{x_i}$,
- $z'_{i-1,i} := \mathtt{H}(k'_{i-1,i}, \mathtt{sid}_i)$ and $z'_{i,i+1} := \mathtt{H}(k'_{i,i+1}, \mathtt{sid}_i)$,
- $z_i := z'_{i-1,i} \oplus z'_{i,i+1}$,
- $\sigma_i := \mathtt{Sign}(sk_i, (U_i, z_i, \mathtt{sid}_i))$,
- broadcast $(U_i, z_i, \sigma_i)$.

**Group Key Computation.** Each $U_i$ checks whether $z_1 \oplus \ldots \oplus z_n = 0$ and whether all received signatures $\sigma_j$ are valid and aborts if any of these checks fails. Otherwise, $U_i$ proceeds as follows:

- iteratively for each $j = i, \ldots, i+n-1$: $z'_{j,j+1} := z'_{j-1,j} \oplus z_j$,
- accept $k_i := \mathtt{H_g}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$.

*P2P Stage.* On input any user identity $U_j \in \mathtt{pid}_i$ the corresponding user $U_i$ proceeds as follows:

- compute $k'_{i,j} := y_j^{x_i} = g^{x_i x_j}$,
- derive $k_{i,j} := \mathtt{H_p}(k'_{i,j}, U_i|y_i, U_j|y_j)$.

(W.l.o.g. we assume that both users $U_i$ and $U_j$ use the same order for the inputs to $\mathtt{H_p}$.)

Here we observe that the computation of p2p keys proceeds without any interaction.

## 4 GKE+S Protocol from Modified BD and PDHKE

In this section we extend our previous protocol to a secure GKE+S solution. We call it mBD+S. The design rationale is as follows: Users run the group stage to compute the group key and then any subgroup can on-demand repeat the second round of the protocol re-using the exponents from the group stage. Observe that each subgroup identified by some $\mathtt{spid} \subset \mathtt{pid}$ uniquely determines its own cycle. However, there can be many subgroups involving the same pair of

users $U_i$ and $U_j$ in which they are located at neighboring positions. In order to avoid the use of the same value $z'_{i,j}$ for the derivation of different subgroup keys we include a subgroup session id $\mathtt{ssid}_i$ given as the concatenation of all $(U_i, y_i)$ with $U_i \in \mathtt{spid}$ as additional input to $\mathtt{H}$.

Since the group stage of the current protocol does not change with respect to the protocol in the previous section, we only present below the subgroup stage.

*Subgroup Stage.* On input any subgroup $\mathtt{spid} \subset \mathtt{pid}$ the corresponding users perform the following steps. For the ease of presentation we assume that $\mathtt{spid} = (U_1, \ldots, U_m)$ with $m < n$ and that $U_0 = U_m$ and $U_{m+1} = U_1$.

**Round 1.** Each $U_i \in \mathtt{spid}$ proceeds as follows:

- extract $\mathtt{ssid}_i := (U_1|y_1, \ldots, U_m|y_m)$ from $\mathtt{sid}_i$;
- $k'_{i-1,i} := y_{i-1}^{x_i}$ and $k'_{i,i+1} := y_{i+1}^{x_i}$,
- $z'_{i-1,i} := \mathtt{H}(k'_{i-1,i}, \mathtt{ssid}_i)$ and $z'_{i,i+1} := \mathtt{H}(k'_{i,i+1}, \mathtt{ssid}_i)$,
- $z_i := z'_{i-1,i} \oplus z'_{i,i+1}$,
- $\sigma_i := \mathtt{Sign}(sk_i, (U_i, z_i, \mathtt{ssid}_i))$,
- broadcast $(U_i, z_i, \sigma_i)$.

**Subgroup Key Computation.** Each $U_i$ checks whether $z_1 \oplus \ldots \oplus z_m = 0$ and whether all received signatures $\sigma_j$ are valid and aborts if any of these checks fails. Otherwise, $U_i$ proceeds as follows:

- iteratively for each $j = i, \ldots, i + m - 1$: $z'_{j,j+1} := z'_{j-1,j} \oplus z_j$,
- accept $k_{i,J} := \mathtt{H_s}(z'_{1,2}, \ldots, z'_{m,1}, \mathtt{ssid}_i)$.

(W.l.o.g. we assume that all users in $\mathtt{spid}$ use the same order for the inputs to $\mathtt{H_s}$.)

Note that for subgroups of size two, i.e. containing only $U_i$ and $U_j$, both users can still derive their p2p key without executing the subgroup stage simply as $k_{i,j} := \mathtt{H_s}(z'_{i,j}, (U_i|y_i, U_j|y_j))$.

## 5 Generalized Security Model

Here we generalize the GKE+P model by Manulis [28] towards consideration of subgroup keys computed by participants of a GKE protocol.

### 5.1 Participants, Sessions, and Correctness of GKE+P Protocols

Let $\mathcal{U}$ denote a set of at most $N$ users (more precisely, their identities which are assumed to be unique) in the universe. We assume that any subset of $n$ users $(2 \leq n \leq N)$ can be invoked for a single session of a GKE+S protocol $\mathcal{P}$. Each $U_i \in \mathcal{U}$ holds a (secret) long-lived key $LL_i$. The participation of $U_i$ in distinct, possibly concurrent protocol sessions is modeled via an unlimited number of *instances* $\Pi_i^s$, $s \in \mathbb{N}$. An instance $\Pi_i^s$ can be invoked for one GKE+S session with some partner id $\mathtt{pid}_i^s \subseteq \mathcal{U}$ encompassing the identities of all the intended participants (including $U_i$). An execution of a GKE+S protocol is then split in two stages, denoted as *group stage* and *subgroup stage*, described in the following.

The group stage results in $\Pi_i^s$ holding a *session id* $\mathtt{sid}_i^s$ which uniquely identifies the current protocol session. Any two instances $\Pi_i^s$ and $\Pi_j^t$ are considered as being *partnered* if $\mathtt{sid}_i^s = \mathtt{sid}_j^t$ and $\mathtt{pid}_i^s = \mathtt{pid}_j^t$. The success of the group stage for some instance $\Pi_i^s$ is modeled through its *acceptance* with some *session group key* $k_i^s$.

Each instance $\Pi_i^s$ that has accepted in the group stage can later be invoked for the subgroup stage on input some *subgroup partner id* $\mathtt{spid}_i^s \subset \mathtt{pid}_i^s$ (which includes $U_i$). This invocation can be performed several times for different subgroups of $\mathtt{pid}_i^s$. The success of the subgroup stage for some $\Pi_i^s$ is modeled through its *acceptance* with some *session subgroup key* $k_{i,J}^s$, whereby $J$ denotes the set of indices of users in $\mathtt{spid}_i^s$, i.e. it includes all $j$ with $U_j \in \mathtt{spid}_i^s$.

**Definition 2 (GKE+S/GKE+P Protocols).** *$\mathcal{P}$ is a* group key exchange protocol enabling on-demand derivation of subgroup keys (GKE+S) *if $\mathcal{P}$ consists of the following two protocols/algorithms:*

$\mathcal{P}$.GKE$(U_1, \ldots, U_n)$**:** *For each $U_i$, a new instance $\Pi_i^s$ with $\mathtt{pid}_i^s = (U_1, \ldots, U_n)$ is created and a probabilistic interactive protocol between these instances is executed such that at the end every instance $\Pi_i^s$ accepts holding the session group key $k_i^s$. This protocol defines the* group stage.

$\mathcal{P}$.SKE$(\Pi_i^s, \mathtt{spid}_i^s)$**:** *On input an accepted instance $\Pi_i^s$ and a subgroup partner id $\mathtt{spid}_i^s \subset \mathtt{pid}_i^s$ this deterministic (possibly interactive) algorithm outputs the session subgroup key $k_{i,J}^s$. This algorithm defines the* subgroup stage. *(We assume that* SKE *is given only for groups of size $n \geq 3$ since for $n = 2$ the group key is sufficient.)*

*A GKE+S protocol $\mathcal{P}$ is* correct *if (when no adversary is present) all instances invoked for the group stage $\mathcal{P}$.*GKE *accept with identical group keys and for all instances $\Pi_j^t$ partnered with $\Pi_i^s$ the subgroup stage results in $\mathcal{P}$.*SKE$(\Pi_j^t, \mathtt{spid}_j^t)$ $= \mathcal{P}$.SKE$(\Pi_i^s, \mathtt{spid}_i^s)$ *if $\mathtt{spid}_j^t = \mathtt{spid}_i^s$.*

*$\mathcal{P}$ is a* group key exchange protocol enabling on-demand derivation of p2p keys (GKE+P) *if it is a GKE+S protocol in which the only admissible input to* SKE *is of the form $\mathtt{spid}_i^s = (U_i, U_j)$. The execution of* SKE *in this case defines the* p2p stage.

### 5.2 Adversarial Model and Security Goals

Security of GKE+S protocols must ensure independence of the session group key $k_i^s$ and any subgroup key $k_{i,J}^s$. In particular the secrecy of $k_i^s$ must hold even if any subgroup key $k_{i,J}^s$ is leaked to the adversary. Similarly, the leakage of the group key $k_i^s$ must guarantee the secrecy of any subgroup key $k_{i,J}^s$. Since the computation of subgroup keys is triggered on-demand we must provide the adversary with the ability to schedule the execution of $\mathcal{P}$.SKE on the subgroups of its choice.

Additionally, GKE+S protocols must ensure independence amongst different subgroup keys. That is the knowledge of some $k_{i,J}^s$ should not reveal information

about any other subgroup key computed by $\Pi_i^s$ or any partner $\Pi_j^t$. This requirement implicitly assumes that some participants $U_j$ with $j \notin J$ may collude during the protocol execution.

Finally, the described independence amongst the group and subgroup keys must hold across different sessions of GKE+S.

**Adversarial Model.** The adversary $\mathcal{A}$, modeled as a PPT machine, can schedule the protocol execution and mount attacks through a set of queries:

- *Execute*$(U_1, \ldots, U_n)$: This query executes the group stage protocol between new instances of $U_1, \ldots, U_n \in \mathcal{U}$ and provides $\mathcal{A}$ with the execution transcript.
- *Send*$(\Pi_i^s, m)$ : With this query $\mathcal{A}$ can deliver a message $m$ to $\Pi_i^s$ whereby $U$ denotes the identity of its sender. $\mathcal{A}$ is then given the protocol message generated by $\Pi_i^s$ in response to $m$ (the output may also be empty if $m$ is unexpected or if $\Pi_i^s$ accepts). A special invocation query of the form *Send*$(U_i, (\textit{'start'}, U_1, \ldots, U_n))$ creates a new instance $\Pi_i^s$ with $\mathtt{pid}_i^s := (U_1, \ldots, U_n)$ and provides $\mathcal{A}$ with the first protocol message. This query can be used by $\mathcal{A}$ also during the subgroup stage if $\mathcal{P}$.SKE requires interaction.
- *SKE*$(\Pi_i^s, \mathtt{spid}_i^s)$: This query allows $\mathcal{A}$ to schedule the on-demand computation of subgroup keys. If $\mathcal{P}$.SKE requires interaction then $\Pi_i^s$ returns $\mathcal{A}$ its first message for the subgroup stage; otherwise $\Pi_i^s$ computes the subgroup key $k_{i,J}^s$. This query is processed only if $\Pi_i^s$ has accepted and $\mathtt{spid}_i^s \subset \mathtt{pid}_i^s$. Additionally, this query can be asked only once per input $(\Pi_i^s, \mathtt{spid}_i^s)$.
- *RevealGK*$(\Pi_i^s)$: This query models the leakage of group keys and provides $\mathcal{A}$ with $k_i^s$. It is answered only if $\Pi_i^s$ has accepted in the group stage.
- *RevealSK*$(\Pi_i^s, \mathtt{spid}_i^s)$: This query models the leakage of subgroup keys and provides $\mathcal{A}$ with the corresponding $k_{i,J}^s$; the query is answered only if the algorithm *SKE*$(\Pi_i^s, \mathtt{spid}_i^s)$ has already been invoked and the subgroup key computed.
- *Corrupt*$(U_i)$: This query provides $\mathcal{A}$ with $LL_i$. Note that in this case $\mathcal{A}$ does not gain control over the user's behavior, but might be able to communicate on behalf of the user.
- *TestGK*$(\Pi_i^s)$: This query models indistinguishability of session group keys. Depending on a given (privately flipped) bit $b$ $\mathcal{A}$ is given, if $b = 0$ a random session group key, and if $b = 1$ the real $k_i^s$. This query can be asked only once and is answered only if $\Pi_i^s$ has accepted in the group stage.
- *TestSK*$(\Pi_i^s, \mathtt{spid}_i^s)$: This query models indistinguishability of session subgroup keys. Depending on a given (privately flipped) bit $b$ $\mathcal{A}$ is given, if $b = 0$ a random session p2p key, and if $b = 1$ the real $k_{i,J}^s$. This query is answered only if the algorithm *SKE*$(\Pi_i^s, \mathtt{spid}_i^s)$ has already been invoked and the subgroup key computed.

**Terminology.** We say that $U$ is *honest* if no *Corrupt*$(U)$ has been asked by $\mathcal{A}$; otherwise, $U$ is *corrupted* (or *malicious*). This also refers to the instances of $U$.

**Two Notions of Freshness.** The classical notion of freshness imposes several conditions in order to prevent any trivial break of the AKE-security. Obviously, we need two definitions of freshness to capture such conditions for the both key types.

First, we define the notion of *instance freshness* which will be used in the definition of AKE-security of group keys. Our definition is essentially the one given in [23].

**Definition 3 (Instance Freshness).** *An instance $\Pi_i^s$ is* fresh *if $\Pi_i^s$ has accepted in the group stage and none of the following is true, whereby $\Pi_j^t$ denotes an instance partnered with $\Pi_i^s$: (1) RevealGK($\Pi_i^s$) or RevealGK($\Pi_j^t$) has been asked, or (2) Corrupt($U'$) for some $U' \in \mathtt{pid}_i^s$ was asked before any Send($\Pi_i^s, \cdot$).*

Note that in the context of GKE+S the above definition restricts $\mathcal{A}$ from active participation on behalf of any user during the attacked session, but implicitly allows for the leakage of (all) subgroup keys.

Additionally, we define the notion of *instance-subgroup freshness* which will be used to specify the AKE-security of subgroup keys.

**Definition 4 (Instance-Subgroup Freshness).** *An instance-subgroup pair $(\Pi_i^s, \mathtt{spid}_i^s)$ is* fresh *if $\Pi_i^s$ has accepted in the group stage and none of the following is true, whereby $\Pi_j^t$ is assumed to be partnered with $\Pi_i^s$ and $\mathtt{spid}_j^t = \mathtt{spid}_i^s$: (1) RevealSK($\Pi_i^s, \mathtt{spid}_i^s$) or RevealSK($\Pi_j^t, \mathtt{spid}_j^t$) has been asked, or (2) Corrupt($U_i$) or Corrupt($U_j$) was asked before any Send($\Pi_i^s, \cdot$) or Send($\Pi_j^s, \cdot$).*

Here $\mathcal{A}$ is explicitly allowed to actively participate in the attacked session on behalf of any user except for $U_i$ and any $U_j \in \mathtt{spid}_i^s$. Also $\mathcal{A}$ may learn the group key $k_i$ and all subgroup keys except for $k_{i,J}$ returned by $\mathcal{P}.\mathrm{SKE}(\Pi_i^s, \mathtt{spid}_i^s)$. This models possible collusion of participants from $\mathtt{pid}_i^s \setminus \mathtt{spid}_i^s$ during the execution of the protocol aiming to break the secrecy of the subgroup key $k_{i,J}$.

**AKE-Security of Group and Subgroup Keys.** For the AKE-security of group keys we follow the definition from [23].

**Definition 5 (AKE-Security of Group Keys).** *Let $\mathcal{P}$ be a correct GKE+P protocol and $b$ a uniformly chosen bit. By $\mathsf{Game}_{\mathcal{A},\mathcal{P}}^{\mathsf{ake\text{-}g},b}(\kappa)$ we define the following adversarial game, which involves a PPT adversary $\mathcal{A}$ that is given access to all queries:*

- *$\mathcal{A}$ interacts via queries;*
- *at some point $\mathcal{A}$ asks a TestGK($\Pi_i^s$) query for some instance $\Pi_i^s$ which is (and remains) fresh;*
- *$\mathcal{A}$ continues interacting via queries;*
- *when $\mathcal{A}$ terminates, it outputs a bit, which is set as the output of the game.*

*We define*

$$\mathsf{Adv}_{\mathcal{A},\mathcal{P}}^{\mathsf{ake\text{-}g}}(\kappa) := \left| 2\Pr[\mathsf{Game}_{\mathcal{A},\mathcal{P}}^{\mathsf{ake\text{-}g},b}(\kappa) = b] - 1 \right|$$

*and denote with* $\mathsf{Adv}^{\mathsf{ake\text{-}g}}_{\mathcal{P}}(\kappa)$ *the maximum advantage over all PPT adversaries* $\mathcal{A}$*. We say that* $\mathcal{P}$ *provides* AKE-security of group keys *if this advantage is negligible.*

Finally, we define AKE-security of subgroup keys while considering possible collusion attacks, as above but with a *TestSK*$(\Pi^s_i, \mathtt{spid}^s_i)$ query.

**Definition 6 (AKE-security of Subgroup Keys).** *Let* $\mathcal{P}$ *be a correct GKE+S protocol and* $b$ *a uniformly chosen bit. By* $\mathsf{Game}^{\mathsf{ake\text{-}s},b}_{\mathcal{A},\mathcal{P}}(\kappa)$ *we define the following adversarial game, which involves a PPT adversary* $\mathcal{A}$ *that is given access to all queries:*

- $\mathcal{A}$ *interacts via queries;*
- *at some point* $\mathcal{A}$ *asks a TestSK*$(\Pi^s_i, \mathtt{spid}^s_i)$ *query for some instance-subgroup pair* $(\Pi^s_i, \mathtt{spid}^s_i)$ *which is (and remains) fresh;*
- $\mathcal{A}$ *continues interacting via queries;*
- *when* $\mathcal{A}$ *terminates, it outputs a bit, which is set as the output of the game.*

*We define*

$$\mathsf{Adv}^{\mathsf{ake\text{-}s}}_{\mathcal{A},\mathcal{P}}(\kappa) := \left|2\Pr[\mathsf{Game}^{\mathsf{ake\text{-}s},b}_{\mathcal{A},\mathcal{P}}(\kappa) = b] - 1\right|$$

*and denote with* $\mathsf{Adv}^{\mathsf{ake\text{-}s}}_{\mathcal{P}}(\kappa)$ *the maximum advantage over all PPT adversaries* $\mathcal{A}$*. We say that* $\mathcal{P}$ *provides* AKE-security of subgroup keys *if this advantage is negligible.*

## 6 Security of Our GKE+P and GKE+S Protocols

In this section we analyze security of our mBD+P and mBD+S protocols using our generalized security model. The corresponding proofs of our theorems can be found in appendix (for the first one) and in the full version of this paper [3] (for the other ones). The security results hold in the random oracle model. The following two theorems show that mBD+P is a secure GKE+P protocol.

**Theorem 1.** *If the GDH problem is hard in* $\mathbb{G}$ *then our protocol* mBD+P *provides AKE-security of group keys and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ake\text{-}g}}_{\mathrm{mBD+P}}(\kappa) \leq & \frac{2N(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_g} + \mathsf{q}_{\mathtt{H}_p})^2}{2^{\kappa-1}} \\ & + 2N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\kappa) + 2\mathsf{q}_{\mathtt{Se}}\left(N\mathsf{q}_{\mathtt{H}}\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa) + \frac{\mathsf{q}_{\mathtt{H}_g}}{2^{\kappa}}\right)\end{aligned}$$

*with at most* $(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})$ *sessions being invoked via Execute and Send queries and at most* $\mathsf{q}_{\mathtt{H}_g}$, $\mathsf{q}_{\mathtt{H}_p}$, *and* $\mathsf{q}_{\mathtt{H}}$ *random oracle queries being asked.*

**Theorem 2.** *If the GDH problem is hard in* $\mathbb{G}$ *then our protocol* mBD+P *provides AKE-security of p2p keys and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ake\text{-}p}}_{\mathrm{mBD+P}}(\kappa) \leq & \frac{2N(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_g} + \mathsf{q}_{\mathtt{H}_p})^2}{2^{\kappa-1}} \\ & + 2N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\kappa) + 2\mathsf{q}_{\mathtt{Se}}\left((N\mathsf{q}_{\mathtt{H}} + \mathsf{q}_{\mathtt{SKE}}\mathsf{q}_{\mathtt{H}_p})\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa)\right)\end{aligned}$$

*with at most* $(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})$ *sessions being invoked via Execute and Send queries, at most* $\mathsf{q}_{\mathtt{SKE}}$ *p2p keys being computed via SKE queries, and at most* $\mathsf{q}_{\mathtt{H}_g}$, $\mathsf{q}_{\mathtt{H}_p}$, *and* $\mathsf{q}_{\mathtt{H}}$ *random oracle queries being asked.*

Our next two theorems prove that mBD+S is a secure GKE+S protocol. The main difference in the security analysis is the consideration of the additional communication round for the subgroup stage (Theorem 4). Since the group stage of mBD+S does not differ from that of mBD+P, the proof of Theorem 3 follows from that of Theorem 1.

**Theorem 3.** *If the GDH problem is hard in* $\mathbb{G}$ *then our protocol* mBD+S *provides AKE-security of group keys and*

$$\mathsf{Adv}^{\mathsf{ake\text{-}g}}_{\mathrm{mBD+S}}(\kappa) \leq \frac{2N(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_g} + \mathsf{q}_{\mathtt{H}_s})^2}{2^{\kappa-1}} + 2N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\kappa) + 2\mathsf{q}_{\mathtt{Se}}\left(N\mathsf{q}_{\mathtt{H}}\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa) + \frac{\mathsf{q}_{\mathtt{H}_g}}{2^{\kappa}}\right)$$

*with at most* $(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})$ *sessions being invoked via Execute and Send queries and at most* $\mathsf{q}_{\mathtt{H}_g}$, $\mathsf{q}_{\mathtt{H}_s}$, *and* $\mathsf{q}_{\mathtt{H}}$ *random oracle queries being asked.*

**Theorem 4.** *If the GDH problem is hard in* $\mathbb{G}$ *then our protocol* mBD+S *provides AKE-security of subgroup keys and*

$$\mathsf{Adv}^{\mathsf{ake\text{-}s}}_{\mathrm{mBD+S}}(\kappa) \leq \frac{2N(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_g} + \mathsf{q}_{\mathtt{H}_s})^2}{2^{\kappa-1}} + 2N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\kappa) + 2\mathsf{q}_{\mathtt{Se}}\left(\big(N + (N-1)\mathsf{q}_{\mathtt{SKE}}\big)\mathsf{q}_{\mathtt{H}}\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa) + \frac{\mathsf{q}_{\mathtt{SKE}}\mathsf{q}_{\mathtt{H}_s}}{2^{\kappa}}\right)$$

*with at most* $(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})$ *sessions being invoked via Execute and Send queries, at most* $\mathsf{q}_{\mathtt{SKE}}$ *subgroup stages being invoked via SKE queries, and at most* $\mathsf{q}_{\mathtt{H}_g}$, $\mathsf{q}_{\mathtt{H}_s}$, *and* $\mathsf{q}_{\mathtt{H}}$ *random oracle queries being asked.*

## 7 Performance Comparison

In Table 1 we compare the complexity of our protocols. We measure the communication costs as a total number of transmitted elements in $\mathbb{G}$, and computation costs as a number of modular exponentiations *per* $U_i$ (in the case of BD we count only exponentiations with $x_i$ assuming that $|x_i| \gg n$). We omit signature generation and verification costs, which are equal for all considered protocols.

The first part of the table is devoted to GKE+P protocols. By GKE+P BD we denote the protocol from [28] in which the original Burmester-Desmedt protocol is executed in parallel with PDHKE, i.e. each user $U_i$ uses two independent exponents $x_i$ and $\bar{x}_i$ for the computation of the group key and any p2p key, respectively. By GKE+P KPT we denote the tree-based Kim-Perrig-Tsudik protocol from [25] in which each user $U_i$ holds only one exponent $x_i$ and uses it to compute both types of keys, which has been proven secure in [28]. Since

**Table 1.** Performance Comparison of GKE+P and GKE+S Protocols

| GKE+P/S Protocols | Rounds | Communication (in $\log Q$ bits) | Computation (in mod. exp. per $U_i$) |
|---|---|---|---|
| GKE+P BD [28] | 2 | $3n$ | 3 |
| GKE+P KPT [28] | 2 | $2n-2$ | $n+2-i$ ($2n-2$ for $U_1$) |
| our mBD+P | 2 | $2n$ | 3 |
| GKE+S BD | 2 | $2m$ | 2 |
| our mBD+S | 1 | $m$ | $\leq 2$ |

all GKE+P protocols apply the PDHKE technique for the derivation of p2p keys we also exclude the computation costs needed to compute a Diffie-Hellman secret $k'_{i,j}$ that requires constantly one exponentiation per each $U_j$. Then, we observe that in comparison to GKE+P BD our mBD+P protocol has better communication complexity since it requires each $U_i$ to hold only one exponent $x_i$. Note also that the PDHKE-KPT protocol has asymmetric costs, depending on the position of $U_i$ in the sequence $U_1, \ldots, U_n$. Although this asymmetry may have benefits in groups with heterogeneous devices we remark that in general this protocol has much worse computation complexity.

The second part of the table compares the effort needed to derive a subgroup key for any subgroup of size $m < n$. By GKE+S BD we consider the trivial solution in which the original BD protocol from [14] is executed for each new subgroup. We compare it to the subgroup stage of our mBD+S protocol, which requires only one communication round per subgroup. Observe that the group stage of mBD+S is executed only once and its complexity is identical to the complete execution of the BD protocol. By $\leq 2$ in the computation costs of mBD+S we point out that users can re-use the intermediate values $k'_{i-1,i}$ and $k'_{i,i+1}$ possibly computed during the group stage or during the subgroup stage for another subgroup, provided the relative position of $U_{i-1}$ and $U_i$ or of $U_i$ and $U_{i+1}$ in the cyclic order of the new subgroup remains the same. In this way our mBD+S protocol also offers a trade-off between space and computation complexity depending on whether users wish to cache the intermediate values that re-occur in different protocol stages.

## 8 Conclusion

The increasing popularity of multi-user communication systems offering various forms of communication can be secured using flexible GKE protocols that provide more than just a secret group key for the initial set of their participants. This paper addressed the extension of this basic functionality of GKE protocols towards the computation of different types of keys allowing a secure mix of group, subgroup, and peer-to-peer communication. While our mBD+P protocol with improved complexity compared to the protocols from [28] allows a

secure mix of group and peer-to-peer communication our mBD+S protocol extends this approach to obtain the additional secure communication within any possible subgroup.

## Acknowledgments

This work was supported in part by the European Commission through the ICT Program under Contract ICT-2007-216646 ECRYPT II, by the French ANR-07-SESU-008-01 PAMPA Project, and through the Center for Advanced Security Research Darmstadt (`www.cased.de`).

## References

1. Abdalla, M., Bohli, J.-M., Vasco, M.I.G., Steinwandt, R.: (Password) Authenticated Key Establishment: From 2-Party to Group. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 499–514. Springer, Heidelberg (2007)
2. Abdalla, M., Bresson, E., Chevassut, O., Pointcheval, D.: Password-Based Group Key Exchange in a Constant Number of Rounds. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 427–442. Springer, Heidelberg (2006)
3. Abdalla, M., Chevalier, C., Manulis, M., Pointcheval, D.: Flexible Group Key Exchange with On-Demand Computation of Subgroup Keys. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 351–368. Springer, Heidelberg (2010); Full version available from the web page of the authors
4. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
5. Biswas, G.P.: Diffie-Hellman Technique: Extended to Multiple Two-Party Keys and One Multi-Party Key. IET Inf. Sec. 2(1), 12–18 (2008)
6. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer, Heidelberg (2003)
7. Boldyreva, A.: Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)
8. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 321–336. Springer, Heidelberg (2002)
9. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.-J.: Provably Authenticated Group Diffie-Hellman Key Exchange. In: ACM CCS 2001, pp. 255–264. ACM, New York (2001)
10. Bresson, E., Manulis, M.: Malicious Participants in Group Key Exchange: Key Control and Contributiveness in the Shadow of Trust. In: Xiao, B., Yang, L.T., Ma, J., Muller-Schloer, C., Hua, Y. (eds.) ATC 2007. LNCS, vol. 4610, pp. 395–409. Springer, Heidelberg (2007)
11. Bresson, E., Manulis, M.: Contributory Group Key Exchange in the Presence of Malicious Participants. IET Inf. Sec. 2(3), 85–93 (2008)
12. Bresson, E., Manulis, M.: Securing Group Key Exchange against Strong Corruptions. In: ACM ASIACCS 2008, pp. 249–260. ACM Press, New York (2008)

13. Bresson, E., Manulis, M., Schwenk, J.: On Security Models and Compilers for Group Key Exchange Protocols. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 292–307. Springer, Heidelberg (2007)
14. Burmester, M., Desmedt, Y.: A Secure and Efficient Conference Key Distribution System. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
15. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
16. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 585–604. Springer, Heidelberg (2005)
17. Desmedt, Y., Lange, T.: Revisiting Pairing Based Group Key Exchange. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 53–68. Springer, Heidelberg (2008)
18. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Tran. on Inf. Th. 22(6), 644–654 (1976)
19. Gorantla, M.C., Boyd, C., González Nieto, J.M.: Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 105–123. Springer, Heidelberg (2009)
20. Ingemarsson, I., Tang, D.T., Wong, C.K.: A Conference Key Distribution System. IEEE Tran. on Inf. Th. 28(5), 714–719 (1982)
21. Jeong, I.R., Lee, D.H.: Parallel Key Exchange. J. of Univ. Comp. Sci. 14(3), 377–396 (2008)
22. Katz, J., Shin, J.S.: Modeling Insider Attacks on Group Key-Exchange Protocols. In: ACM CCS 2005, pp. 180–189. ACM Press, New York (2005)
23. Katz, J., Yung, M.: Scalable Protocols for Authenticated Group Key Exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
24. Kim, H.-J., Lee, S.-M., Lee, D.H.: Constant-Round Authenticated Group Key Exchange for Dynamic Groups. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 245–259. Springer, Heidelberg (2004)
25. Kim, Y., Perrig, A., Tsudik, G.: Group Key Agreement Efficient in Communication. IEEE Tran. on Comp. 53(7), 905–921 (2004)
26. Kim, Y., Perrig, A., Tsudik, G.: Tree-Based Group Key Agreement. ACM Trans. on Inf. and Syst. Sec. 7(1), 60–96 (2004)
27. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
28. Manulis, M.: Group Key Exchange Enabling On-Demand Derivation of Peer-to-Peer Keys. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 1–19. Springer, Heidelberg (2009)
29. Manulis, M.: Security-Focused Survey on Group Key Exchange Protocols. Cryptology ePrint Archive, Report 2006/395 (2006)
30. Mayer, A., Yung, M.: Secure Protocol Transformation via "Expansion": From Two-Party to Groups. In: ACM CCS 1999, pp. 83–92. ACM Press, New York (1999)
31. Nam, J., Paik, J., Kim, U.-M., Won, D.: Constant-Round Authenticated Group Key Exchange with Logarithmic Computation Complexity. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 158–176. Springer, Heidelberg (2007)
32. Steer, D.G., Strawczynski, L., Diffie, W., Wiener, M.J.: A Secure Audio Teleconference System. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 520–528. Springer, Heidelberg (1990)

33. Steiner, M., Tsudik, G., Waidner, M.: Diffie-Hellman Key Distribution Extended to Group Communication. In: ACM CCS 1996, pp. 31–37. ACM Press, New York (1996)
34. Wu, S., Zhu, Y.: Constant-Round Password-Based Authenticated Key Exchange Protocol for Dynamic Groups. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 69–82. Springer, Heidelberg (2008)

# Appendix

## A Proof of Theorem 1

In the following we show that the advantage of $\mathcal{A}$ in distinguishing $k_i$ from some random element from $\{0,1\}^\kappa$ is negligible. We construct a sequence of games $\mathbf{G}_0, \ldots, \mathbf{G}_4$ and denote by $\mathsf{Win}_i^{\mathsf{ake\text{-}g}}$ the event that the bit $b'$ output by $\mathcal{A}$ is identical to the randomly chosen bit $b$ in the $i$-th game. Recall that in each game $\mathit{TestGK}(\Pi_i^s)$ is answered only if the instance $\Pi_i^s$ is fresh.

**Game $\mathbf{G}_0$.** This is the real execution of mBD+P in which a simulator $\Delta$ truly answers all queries of $\mathcal{A}$ on behalf of the instances as defined in $\mathsf{Game}^{\mathsf{ake\text{-}g},b}_{\mathcal{A},\mathrm{mBD+P}}(\kappa)$.

We assume that $\mathcal{A}$ has access to the hash queries for the hash functions $\mathtt{H}$, $\mathtt{H_g}$, and $\mathtt{H_p}$, which are modeled as random oracles in the classical way, i.e., by returning new random values for new queries and replaying answers if the queries were previously made.

**Game $\mathbf{G}_1$.** In this game we exclude for every honest user $U_i$ the collisions of the transcripts $(U_i, y_i)$ and group keys $k_i$ computed in different sessions. We also exclude any collisions between $k_i$ and any p2p key $k_{i,j}$. Regarding the transcripts we observe that if $U_i$ is honest then its session value $y_i$ is randomly distributed in $\mathbb{G}$ (as a result of $y_i := g^{x_i}$ for $x_i \in_R \mathbb{Z}_Q$). Thus, according to the birthday paradox the collision on transcripts occurs with the probability of at most $N(\mathtt{q_{Ex}}+\mathtt{q_{Se}})^2/Q$ over all possible users (recall that sessions can be invoked via *Execute* and *Send* queries). The uniqueness of transcripts also implies the uniqueness of inputs to $\mathtt{H_g}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$. By construction inputs to $\mathtt{H_g}$ remain always different from the inputs to $\mathtt{H_p}$. Since $\mathtt{H_g}$ and $\mathtt{H_p}$ are modeled as random functions we can also apply the birthday paradox and upper-bound the probability of collisions for $k_i$ and collisions between $k_i$ and any $k_{i,j}$ by $(\mathtt{q_{H_g}} + \mathtt{q_{H_p}})^2/2^\kappa$. Thus,

$$|\Pr[\mathsf{Win}_1^{\mathsf{ake\text{-}g}}] - \Pr[\mathsf{Win}_0^{\mathsf{ake\text{-}g}}]| \leq \frac{N(\mathtt{q_{Ex}} + \mathtt{q_{Se}})^2}{Q} + \frac{(\mathtt{q_{H_g}} + \mathtt{q_{H_p}})^2}{2^\kappa}.$$

**Game $\mathbf{G}_2$.** In this game we assume that $\Delta$ fails and bit $b'$ is set at random if $\mathcal{A}$ queries *Send* containing $(U_i, z_i, \sigma_i)$ with $\sigma_i$ being a valid signature that has not been previously output by an uncorrupted oracle $\Pi_i^s$. In other words the simulation aborts if $\mathcal{A}$ outputs a successful forgery. Following the classical reductionist argument (see for instance [9]) we can build a forger against the signature scheme and upper-bound the probability difference

$$|\Pr[\mathsf{Win}_2^{\mathsf{ake\text{-}g}}] - \Pr[\mathsf{Win}_1^{\mathsf{ake\text{-}g}}]| \leq N\mathsf{Succ}_\Sigma^{\mathsf{euf\text{-}cma}}(\kappa).$$

**Game $\mathbf{G}_3$.** In this game we let $\Delta$ guess a value $q^* \in [1, \mathsf{q_{Se}}]$ and abort if the *TestGK* query is not asked for the session invoked by the $q^*$-th query. Let $\mathsf{Q}$ be the event that this guess is correct and $\Pr[\mathsf{Q}] = 1/\mathsf{q_{Se}}$. Thus,

$$\Pr[\mathsf{Win}_3^{\mathsf{ake\text{-}g}}] = \Pr[\mathsf{Win}_2^{\mathsf{ake\text{-}g}}]\frac{1}{\mathsf{q_{Se}}} + \frac{1}{2}\left(1 - \frac{1}{\mathsf{q_{Se}}}\right).$$

This implies,

$$\Pr[\mathsf{Win}_2^{\mathsf{ake\text{-}g}}] = \mathsf{q_{Se}}\left(\Pr[\mathsf{Win}_3^{\mathsf{ake\text{-}g}}] - \frac{1}{2}\right) + \frac{1}{2}.$$

**Game $\mathbf{G}_4$.** In this game we assume that $\Delta$ is given access to the private random oracle $\mathtt{H}' : \{0,1\}^* \to \{0,1\}^\kappa$ and computes in the simulation of the $q^*$-th session for each pair of consecutive users $(U_i, U_{i+1})$ in the cycle $z'_{i,i+1} = \mathtt{H}'(i, i+1)$. Clearly both games remain indistinguishable unless $\mathcal{A}$ queries $\mathtt{H}(k'_{i,i+1})$ for any $i \in [1, n]$. However, this query can be used to break the GDH problem in $\mathbb{G}$, i.e. $\Delta$ embeds $g^a$ and $g^b$ from the challenge of the GDH problem into the corresponding values $g^{x_i}$ and $g^{x_{i+1}}$, and uses the access to the DDH oracle $\mathcal{D}$ in order to identify $k'_{i,i+1} = g^{x_i x_{i+1}}$ provided by $\mathcal{A}$ as input to $\mathtt{H}$. Therefore,

$$|\Pr[\mathsf{Win}_4^{\mathsf{ake\text{-}g}}] - \Pr[\mathsf{Win}_3^{\mathsf{ake\text{-}g}}]| \leq N\mathsf{q_H}\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa).$$

As a result of this game the group key $k_i$ computed in the $q^*$-th session is the output of $\mathtt{H_g}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$ where all input values $z'_{1,2}, \ldots, z'_{n,1}$ are uniform in $\{0,1\}^\kappa$. Since $\mathtt{H_g}$ is modeled as a random oracle and collisions on group keys computed in two different sessions have been excluded in Game $\mathbf{G}_1$ the probability that $\mathcal{A}$ wins without querying $\mathtt{H_g}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$ is $1/2$; on the other hand, the probability that $\mathcal{A}$ asks such a query is given by $\mathsf{q}_{\mathtt{H_g}}/2^{n\kappa} < \mathsf{q}_{\mathtt{H_g}}/2^\kappa$ (for the guess of $z'_{1,2}, \ldots, z'_{n,1}$). Hence, $\Pr[\mathsf{Win}_4^{\mathsf{ake\text{-}g}}] \leq 1/2 + \mathsf{q}_{\mathtt{H_g}}/2^\kappa$.

Summarizing the above equations we obtain a negligible advantage

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ake\text{-}g}}_{\mathrm{mBD+P}}(\kappa) &= |2\Pr[\mathsf{Win}_0^{\mathsf{ake\text{-}g}}] - 1| \\ &\leq \frac{2N(\mathsf{q_{Ex}} + \mathsf{q_{Se}})^2}{Q} + \frac{(\mathsf{q}_{\mathtt{H_g}} + \mathsf{q}_{\mathtt{H_p}})^2}{2^{\kappa-1}} + 2N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\kappa) \\ &\quad + 2\mathsf{q_{Se}}\left(N\mathsf{q_H}\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa) + \frac{\mathsf{q}_{\mathtt{H_g}}}{2^\kappa}\right).\end{aligned}$$

□

# Quantum Readout of Physical Unclonable Functions

Boris Škorić

Eindhoven University of Technology

**Abstract.** Physical Unclonable Functions (PUFs) are physical structures that are hard to clone and have a unique challenge-response behaviour. In this paper we propose a new security primitive, the **quantum-readout PUF** (QR-PUF): a classical PUF which is challenged using a quantum state, and whose response is also a quantum state. By the no-cloning property of unknown quantum states, attackers cannot intercept challenges or responses without noticeably disturbing the readout process. Thus, a verifier who sends quantum states as challenges and receives the correct quantum states back can be certain that he is probing a specific QR-PUF without disturbances, even in the QR-PUF is far away 'in the field' and under hostile control. For PUFs whose information content is not exceedingly large, all currently known PUF-based authentication and anti-counterfeiting schemes require trusted readout devices in the field. Our quantum readout scheme has no such requirement.

We show how the QR-PUF authentication can be interwoven with Quantum Key Exchange (QKE), leading to an authenticated QKE protocol between two parties with the special property that it requires no a priori secret shared by the two parties, and that the quantum channel is the authenticated channel, allowing for an unauthenticated classical channel.

## 1 Introduction

### 1.1 Physical Unclonable Functions

The term PUF was coined by Pappu et al. in [21,22]. Although the use of physical structures for authentication purposes dates back a long time, the work [21,22] was the first to introduce the 'function' behaviour of such objects and to consider mathematical unclonability as well (difficulty of modelling). It was shown that an optical medium with a high density of scatterers makes an extremely good PUF: A challenge consists e.g. of the angle of incidence of a laser beam; the response is the speckle pattern resulting from multiple coherent scattering.

By now there is a whole zoo of PUF-like systems that have appeared in the literature: optical PUFs, delays in integrated circuits [12], dielectric properties of security coatings [26], two-dimensional fiber-optic configurations [17], radio-frequent probing of wire configurations [8] and thin-film resonators [30], laser probing of fibers in paper [4], startup values of SRAM cells [14], butterfly PUFs [18], phosphor patterns [6], phase-change memory states [19]. What all these

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 369–386, 2010.

have in common is a strong dependence on uncontrollable aspects of the manufacturing process. A partial overview of the field is given in [27].

Because of the variety of physical systems and security goals involved, the terminology can be confusing. There exist multiple definitions of a PUF, differing in their list of properties that must be satisfied. Often mentioned properties are physical unclonability, mathematical unclonability, uniqueness, tamper evidence, high response entropy, large number of different challenges and read-proofness. Descriptions like Physical One-Way Function, Physical Unknown Function, Physically Obfuscated/Obscured Key, and Physical Pseudorandom Function are sometimes used to specify which properties are most important for a certain application. In this paper we rely only on the following properties of the pair {physical object, measurement method}:

- Physical unclonability. It is technically/financially infeasible to make a physical clone of a given QR-PUF (given full knowledge of this QR-PUF), such that it behaves exactly as the original one for the given measurement method.
- Quantum-computational unclonability. This is a new physical assumption. It is technically/financially infeasible to build a quantum computer and input/output handling which emulates a given QR-PUF (given full knowledge of this QR-PUF) with a sufficiently small delay time.
- Uniqueness. Different challenges can be applied to the QR-PUF. The responses to these challenges (measurements) have to contain enough entropy so that all the to-be-authenticated QR-PUFs can be distinguished. In fact, uniqueness is implied by physical unclonability; if there are too few possible PUFs, then collisions can be created by manufacturing many random PUFs.

The entropy of optical PUFs was studied in [28,29,27]. We stress that the responses to all challenges, for each QR-PUF, are allowed to be public knowledge. In this paper, there is no secrecy concerning any aspect of the QR-PUFs.

Note that the PUF literature is concerned with *classical* physics. The word 'unclonability' is an assumption about the *effort* it takes to produce a clone; it does not indicate that it is impossible to produce one. There is no relation to the *provable* no-cloning theorem [31,9] of unknown quantum states. Our QR-PUF is classical in itself, but the measurement is quantum, and we will make use of the no-cloning theorem to detect tampering with the measurement process.

### 1.2 Getting Rid of the Trusted Remote Reader

Some of the early PUF-based remote authentication protocols rely on the vastness of the PUF's entropy. At enrolment, Alice measures PUF responses for a large number of random challenges. She stores the table of challenge-response pairs (CRPs). Then the PUF is given to Bob. When Alice wants to authenticate Bob, she sends him a challenge, randomly selected from her CRP table. This is done over a public channel. Bob feeds the challenge to his PUF and measures the response. He sends the response to Alice over a public channel. If Bob's response is correct, Alice is convinced that he has access to the PUF. Alice deletes the used CRP from her list.

The security of such a protocol is based on the following physical assumptions: (I) Knowledge of eavesdropped CRPs gives negligible information about the response to a new challenge; (II) It is infeasible for an attacker to characterize Bob's PUF in a short amount of time with enough accuracy to predict the response to a new random challenge.

Although optical PUFs come a long way [28,29], it is not clear whether high-entropy PUFs will be feasible in practice. *In this paper we therefore do not make assumption II.* Even if one gives up the high-entropy property, PUFs are very useful. For instance[1], an authentication method can be designed that is based solely on physical unclonability. When a PUF gets manufactured, a certification authority enrolls it. This can be done in either of two ways. (i) An identifier $I$ and a precise characterization of the PUF are entered into a publicly readable, tamper proof database; or (ii) the enroller signs a certificate containing $I$ and the PUF characterization, which certificate is then stored publicly. When a verifier wants to see if a certain PUF is authentic, he can check its challenge-response behaviour against the enrolled data. As long as the verifier is certain that responses are coming directly from a real PUF (as opposed to an emulation or a replay), he is able to verify that a physical structure is the same as the enrolled one.

In the above scheme the verifier must be in control of the measurement device, to prevent spoofing. This leads to an extra requirement for remote verification: trust in a remotely located piece of hardware. One results of our paper is that we achieve remote PUF authentication *without the trusted remote reader.*

### 1.3 Authenticated Quantum Key Exchange

Quantum Key Exchange (QKE), also known as Quantum Key Distribution (QKD), Quantum Key Agreement, and Quantum Cryptography, was first proposed in 1984 [2] (BB84). QKE is a protocol that allows Alice and Bob to establish an unconditionally secure shared secret if they have a channel at their disposal over which they can send quantum states. The security is based on the laws of physics, in particular the no cloning theorem [31,9]. Since 1984 progress has been made on all relevant aspects of QKE: single-photon sources, detectors, fiber optics, attacks and defense, use of entanglement [11], error-correction codes, privacy amplification and security proof methods. QKE products based on BB84 are now commercially available. Quantum observables other than polarization have been proposed for QKE, e.g. phase [16] and squeezed coherent states [13].

The classical communication channel between Alice and Bob is allowed to be public, but it has to be authenticated in order to prevent man-in-the-middle attacks. The aim of QKE is to achieve *unconditional* security. One way to achieve information-theoretic authentication is to let Alice and Bob share a short initial MAC key. Though this may look like cheating, it is not. QKE serves to indefinitely lengthen an initial shared secret. Another authentication method uses initial entangled states [24]. This approach has a number of drawbacks, such as the requirement that the entangled state has to be stored for a long time.

[1] Another example is secure key storage, which relies only on read-proofness.

### 1.4 Our Contributions

We introduce the concept of a Quantum Readout PUF (QR-PUF): a classical PUF that is challenged using a quantum state, and whose response is also a quantum state. Te interaction with the challenge does not affect the PUF; the challenge states evolves quantum-physically into the response state. The no-cloning theorem ensures that eavesdropping on challenges or responses cannot go unnoticed. We rely on physical assumptions about the PUF: physical unclonability and quantum-computational unclonability. A further assumption is that PUFs are enrolled in a trusted environment and that PUF enrollment data is available in a tamper-free form. We allow attackers to have access to this data.

We present a protocol for authenticating a QR-PUF remotely without reliance on a trusted remote reader device. The remote authentication also serves as a distance bounding protocol. We give a security analysis, allowing all QR-PUF properties to be public, i.e. the only secrets in the protocol are the challenge states sent by the verifier. We prove that intercept-resend attacks fail. However, a QR-PUF can be spoofed by a sufficiently powerful quantum computer, combined with sufficiently fast ways of measuring qubit states as well as transferring challenge/response states into qubits and back. One of our physical assumptions says that such a combination of quantum physical techniques is infeasible.

For a PUF which has both transmission and reflection of incoming states, we show how the authentication can be intertwined with QKE.

- The reflected states are used to authenticate the QR-PUF.
- The QR-PUF is completely 'transparent' with respect to transmitted states; Alice can choose which quantum state reaches Bob after transmission through the QR-PUF, even if the transmission process is complicated.

We sketch an authenticated QKE protocol. We prove that intercept-resend attacks fail. The scheme achieves interesting security properties:

1. The initial authentication between parties is achieved *without any shared secret* (such as a MAC key or entangled state).
2. The security of the initial authentication is based on *physical assumptions* about the QR-PUF, combined with trust in the enrolled data. Thus, we do not have unconditional security. However, we find it worth showing that physical assumptions can be used in this way.
3. Traditional QKE schemes require an authenticated classical channel. We have the completely opposite situation: an authenticated quantum channel. Hence we do not require the classical channel to be authenticated.
4. The authentication is based on the possession of an object. A secret can be stolen surreptitiously. Theft of an object is usually noticed.

## 2 Preliminaries

We use the notation $[n] = \{1, \cdots, n\}$. We use Dirac's bra-ket notation. We consider a system with 'external' (e.g. direction of motion) and 'internal' (e.g.

spin or polarization) degrees of freedom. This is denoted as a tensor product of Hilbert spaces: $\mathcal{H} = \mathcal{H}_{\text{ext}} \otimes \mathcal{H}_{\text{int}}$, where $\mathcal{H}$ is the full Hilbert space. We will be dealing with three types of state:

- **Challenge.** A quantum state is moving from Alice to Bob's PUF.
- **Reflected.** After interaction with Bob's PUF it moves back to Alice.
- **Transmitted.** The particle has interacted with Bob's PUF, and its internal state has changed. The particle does not return to Alice, but moves on.

These three situations are represented in only *two* states of motion: moving away from Alice or towards her. Any state $|\psi\rangle \in \mathcal{H}$ can be decomposed as

$$|\psi\rangle = |\text{out}\rangle \otimes |\psi_1\rangle + |\text{in}\rangle \otimes |\psi_2\rangle \tag{1}$$

where $|\text{out}\rangle, |\text{in}\rangle \in \mathcal{H}_{\text{ext}}$ and $|\psi_1\rangle, |\psi_2\rangle \in \mathcal{H}_{\text{int}}$. We have $\langle \text{out}|\text{in}\rangle = 0$. The following notation can also be used,

$$|\psi\rangle = \begin{pmatrix} |\psi_1\rangle \\ |\psi_2\rangle \end{pmatrix}. \tag{2}$$

$\mathcal{H}_{\text{int}}$ is $n$-dimensional. The interaction between the quantum system and the PUF is assumed to be completely coherent: time evolution is determined by the Schrödinger equation, without any state collapse. We abstractly represent the interaction with the PUF as a unitary time evolution operator $S$, also known as the scattering matrix or S-matrix. $S$ maps 'before' states to 'after' states. Let $|\psi'\rangle \in \mathcal{H}$ be the state after the interaction, then in the notation of (2) we have

$$\begin{pmatrix} |\psi_1'\rangle \\ |\psi_2'\rangle \end{pmatrix} = S \begin{pmatrix} |\psi_1\rangle \\ |\psi_2\rangle \end{pmatrix} \quad ; \quad S = \begin{pmatrix} T & -R^\dagger \\ R & T^\dagger \end{pmatrix}. \tag{3}$$

$T$ (the 'transmission matrix') describes the internal state when the particle emerges at the other side of the PUF, and $R$ ('reflection matrix') does the same for reflected states. The unitary evolution ($S^\dagger S = SS^\dagger = \mathbf{1}$) implies that $T^\dagger T + R^\dagger R = \mathbf{1}$ and that $T$, $T^\dagger$, $R$ and $R^\dagger$ all commute with each other[2]. Together $R$ and $T$ completely determine the challenge-response behaviour of the PUF. Hence, physical unclonability of a PUF means that it is difficult to create a PUF which behaves precisely according to some pre-specified $R$ and $T$.

In the rest of the paper we omit reference to the external degree of freedom. It will always be understood that Challenge and Transmitted states are moving away from Alice, and that Reflected states are moving towards her.

We assume that there is a lot of freedom in doing measurements and in preparing quantum states. More precisely, we assume that Alice can prepare basically any state $|\psi\rangle \in \mathcal{H}_{\text{int}}$. Similarly, there is a large number of different observables that Alice and Bob can measure. (In the case of photon polarization there is even a continuum of observables, namely the polarization component in any direction.) This freedom allows one to define a set of measurements that accurately characterize a PUF. More details are given in the full version [1].

[2] We can write $T = U^{-1}\Lambda U$ and $R = U^{-1}\Gamma U$, where $U$ is a unitary matrix and $\Lambda$ and $\Gamma$ are complex-valued diagonal matrices satisfying $|\Lambda_i|^2 + |\Gamma_i|^2 = 1$ for all $i \in [n]$. The basis vectors contained in $U$ are the eigenmodes of $S$.

We assume that a measurement of the internal state is able to discern whether or not there is a particle present at all. This statement is not trivial. Filter-based polarization measurements for instance do not see the difference between darkness and polarization perpendicular to the applied filter. Measurements using a polarization splitter and measurement of differential phase shift [16], on the other hand, *do* distinguish between presence and absence of a photon.

## 3 Remote Authentication of a QR-PUF

We present two protocols for remote authentication. The first one is applicable when every conceivable projection operator corresponds to an actual measurement that can be performed. The second one is geared to more restricted circumstances where Alice has only a limited number of projection measurements at her disposal. Both protocols are based on reflected states only. We consider the case $T = 0$.[3] In Section 4, a protocol will be discussed with $T \neq 0$.

### 3.1 Attack Model

Alice wants to verify if Bob has *real-time access* to a certain QR-PUF. QR-PUF enrollment occurs in a trusted environment. Alice has access to the enrollment data in a tamper-proof way (e.g. she is the enroller). The main attack is impersonation: someone may be trying to trick Alice into believing that he has access to the QR-PUF. There is a quantum channel between Alice and Bob. The attacker has physical access to challenge states as well as reflected states. He can destroy quantum states, perform measurements on them, and insert new states. However, he cannot clone a state. We allow $R$ and $T$ to be fully known to the attacker. However, as discussed in Section 1.1, it is assumed infeasible to create a new QR-PUF whose reflection matrix is $R$. It is also assumed infeasible to build an effective quantum computer that emulates the QR-PUF.

### 3.2 Authentication Protocol 1

It is assumed that Alice can perform a projection measurement onto any state in the Hilbert space.

Enrollment phase
A QR-PUF with identifier $I$ is accurately characterized. This yields the matrix $R$. The QR-PUF is given to Bob.

Authentication phase
Bob claims that he has access to the QR-PUF with identifier $I$. Alice fetches the enrollment data $R$ corresponding to $I$. She initializes counters $n_{\mathrm{ret}}$ and $n_{\mathrm{ok}}$ to zero. She then repeats the following procedure $m$ times:

[3] There is no loss of generality; transmitted states can always be re-routed to become part of the reflected state.

1. Alice prepares a state $|\psi\rangle$ uniformly at random and sends it to Bob.
2. Alice receives either nothing ('$\perp$') or a returning state $|\omega\rangle$. If she receives a state[4], then
   (a) She increases $n_{\text{ret}}$ by one.
   (b) If $|\psi\rangle$ was properly reflected by Bob's QR-PUF, then the returned state should be $|\omega_\psi\rangle := R|\psi\rangle$. Alice measures the projection of $|\omega\rangle$ onto $|\omega_\psi\rangle$, i.e. she performs a measurement of the operator $|\omega_\psi\rangle\langle\omega_\psi|$, obtaining either '0' or '1' as the outcome. If the outcome is 1, then she increases $n_{\text{ok}}$ by one.

If the fraction $n_{\text{ret}}/m$ is not consistent with the expected noise level, then the protocol is aborted. If $n_{\text{ok}} \geq (1-\varepsilon)n_{\text{ret}}$ then Alice is convinced that she has probed the QR-PUF with identifier $I$. Here $\varepsilon$ is a parameter denoting the tolerable fraction of wrong responses.

### 3.3 Security of Authentication Protocol 1

In the above protocol, and the other protocols in this paper, Alice waits for a returning state before sending her next challenge state. This considerably simplifies the security proofs, since in this setting it suffices to look at the security of an isolated round without having to worry about (entanglement) attacks on multiple challenge states. We leave more general protocols and their security proofs (e.g. using methods developed in [23]) as a subject for future work.

**Intercept-resend attacks.** We consider the type of attack where an impostor tries to convince Alice that he has the QR-PUF. A generic intercept-resend attack consists of the following steps. (a) For each of the $m$ rounds he picks a random orthonormal basis of the $n$-dimensional Hilbert space. The corresponding Hermitian operator is denoted as $B$, with eigenvalues $b_j$ and eigenstates $|b_j\rangle$ that form the basis. (b) He measures $B$, obtaining outcome $b_k$ for some $k \in [n]$. (c) He chooses a state $|\zeta\rangle$ such that $\langle b_k|\zeta\rangle = 0$ and a parameter $\alpha \in [0, \pi/2]$ according to some (possibly probabilistic) strategy $\mathcal{A}$ that may depend on $B$ and $k$. He computes $|\chi\rangle$,

$$|\chi\rangle = \cos\alpha|b_k\rangle + \sin\alpha|\zeta\rangle, \tag{4}$$

which is his guess for $|\psi\rangle$. Finally he prepares a state $|\omega\rangle$ and sends it to Alice,

$$|\omega\rangle = R|\chi\rangle. \tag{5}$$

**Theorem 1.** *In the intercept-resend attack described above, the impostor's probability $p_1$ of success in a single round of protocol 1 is bounded as*

$$p_1 \leq \frac{3}{n+2}.$$

*Proof.* See the full version [1] of the paper.

[4] A state that arrives too late is counted as $\perp$. This is a form of distance bounding. If Alice has a rough idea where Bob should be located, she knows what round-trip time to expect.

**Attack by quantum computer.** It is possible in theory to break the authentication scheme if one has [i] a sufficiently powerful quantum computer (QC) *and* [ii] a sufficiently fast way of transferring challenge states into the QC's qubits and transferring the computation result back to a response state.

We give a high-level description of the attack. First the challenge state $|\psi\rangle$ is transferred to the QC memory. This can be done without measuring $|\psi\rangle$, namely by quantum teleportation [3], in particular a form of teleportation that transfers states from one type of physical system to another [10,20]. Then the QC does a computation that has the effect of applying $R$. This is possible in theory since $R$ is known publicly, and applying $R$ is a unitary operation. The result is teleported to a response state and sent to Alice over the quantum channel.

Our assumption denoted as 'quantum-computational unclonability' states that it is either technically/financially too difficult to pull off the above given steps or, if all the hardware works, the attack is too slow.

It is not yet clear to us how to make quantitative statements about the difficulty of 'quantum' attacks. For instance, the challenge $\psi$ could comprise a random choice of photon wavelength $\lambda$, with the $R$-matrix depending on $\lambda$. How is the information about $\lambda$ transferred to the QC memory? How does the attacker know which $R$ to apply? Does he have to construct a big unitary operation that acts on all wavelengths simultaneously? That would certainly strain the QC hardware.

**Attack with an imperfect physical clone.** Consider the case of an attempted physical clone which is not quite equal to the original one.

**Theorem 2.** *Let $\delta > 0$ be a constant. Let the imperfect clone have a unitary reflection matrix $R'$. Let the eigenvalues of $R^{-1}R'$ be denoted as $\{e^{i\varphi_k}\}_{k\in[n]}$. Let these eigenvalues satisfy*

$$|\sum_{k\in[n]} e^{i\varphi_k}|^2 \leq n^2(1-\delta). \tag{6}$$

*Then the impostor's per-round probability of success is bounded by*

$$p_1 \leq 1 - \frac{n}{n+2}\delta. \tag{7}$$

*Proof.* See the full version [1] of the paper.

### 3.4 Authentication Protocol 2

The 2nd protocol also considers the case $T = 0$. The difference with protocol 1 is a restriction on the measurements available to Alice. She can no longer choose any projection measurement $|\psi\rangle\langle\psi|$. Instead she has a limited set of $s$ different observables $\{X_\alpha\}_{\alpha=1}^{s}$ at her disposal. These are Hermitian operators, and hence they have orthonormal sets of eigenvectors. The $i$'th eigenstate of $X_\alpha$ is denoted as $|\alpha i\rangle$, with $i \in [n]$. We make two assumptions about the set of observables:

- The $X_\alpha$ all have non-degenerate eigenvalues. This allows Alice to effectively turn measurement of $X_\alpha$ into a measurement of $|\alpha i\rangle\langle\alpha i|$ for some $i$.
- For $\alpha \neq \beta$ it holds that $\forall i, j \in [n] : \ |\langle\alpha i|\beta j\rangle| < D$, where $D \in [1/\sqrt{n}, 1)$.

The impostor has no restrictions on his choice of observable $B$. There are no restrictions on the state preparation, neither for Alice nor the impostor.

Enrollment phase
A QR-PUF with identifier $I$ is accurately characterized. This yields the matrix $R$. The QR-PUF is given to Bob.

Authentication phase
Bob claims that he has access to the QR-PUF with identifier $I$. Alice fetches the enrollment data $R$ corresponding to $I$. She initializes counters $n_{\text{ret}}$ and $n_{\text{ok}}$ to zero. She then repeats the following procedure $m$ times:

1. Alice draws $\alpha \in [s]$ and $i \in [n]$ uniformly at random. She prepares the state $R^{-1}|\alpha i\rangle$ and sends it to Bob.
2. Alice receives either nothing ('$\perp$') or a returning state $|\omega\rangle$. If it is a state,
   (a) She increases $n_{\text{ret}}$ by one.
   (b) She performs a measurement of the operator $|\alpha i\rangle\langle\alpha i|$, obtaining either '0' or '1' as the outcome. If the outcome is 1, then she increases $n_{\text{ok}}$.

If the fraction $n_{\text{ret}}/m$ of returned states is not consistent with the expected noise level, then the protocol is aborted. If $n_{\text{ok}} \geq (1-\varepsilon)n_{\text{ret}}$ then Alice is convinced that she has probed the QR-PUF with identifier $I$. Here $\varepsilon$ is a parameter denoting the tolerable fraction of wrong responses.

### 3.5 Security of Authentication Protocol 2

The intercept-resend attack is of the same kind as in Section 3.3. The impostor performs a measurement of some $B$ and obtains outcome $b_k$. He has some strategy $\mathcal{A}$ to choose a state $|\omega\rangle$ as a function of $B$ and $k$. He sends $|\omega\rangle$ to Alice.

**Theorem 3.** *In the above described intercept-resend attack on protocol 2, the per-round success probability $p_2$ is bounded by*

$$p_2 < \frac{1 + (s-1)D}{s}.$$

*Proof.* See the full version [1] of the paper.

Just as protocol 1, this protocol is vulnerable to a 'quantum' attack employing quantum teleportation and a quantum computer. The attack and the requirements are exactly the same as in Section 3.3.

### 3.6 Limitations on the State Preparation

If there are also restrictions on Alice's state preparation, it is still possible to construct an effective authentication scheme. Consider the most pessimistic case: Alice is only capable of preparing eigenstates of the observables $X_\alpha$. This is still sufficient for her to fully characterize Bob's QR-PUF. All she has to do is select random $\alpha$ and $j$, send $|\alpha j\rangle$ and do a measurement of $X_\alpha$ on the reflected state. After many repetitions, this procedure gives her all the matrix elements $\langle\alpha j|R|\alpha k\rangle$.

That is a perfectly viable method. The only drawback is proof-technical. Proving security cannot be done by finding a bound on a per-round success probability of spoofing; even if Bob's return states are correct, Alice's measurement results are stochastic, since in general $R|\alpha j\rangle$ is not an eigenstate of $X_\alpha$. Hence Alice cannot assign a label 'correct' to an isolated round.

## 4 Combining QR-PUF Authentication with QKE

We combine the authentication with a QKE protocol that is performed on the Transmitted states. The intuition is as follows. Since Alice knows $T$, she can prepare $|\psi\rangle \propto T^{-1}|\varphi\rangle$, for arbitrary $\varphi$, such that Bob receives $\varphi$ if transmission occurs. In this sense the QR-PUF is 'transparent' for the purpose of sending specific states to Bob, and they can run QKE *through the QR-PUF*. Some of Alice's challenges will be reflected back to Alice. She uses these to authenticate the QR-PUF. We assume that Alice can perform arbitrary projection measurements.

The protocol presented in this section authenticates Bob to Alice. If mutual authentication is required, the protocol can be run twice: first authenticating one party, then the other.

### 4.1 Attack Model

Bob claims that he possesses a QR-PUF with identifier $I$. Alice's goal is to authenticate QR-PUF $I$ and to generate a shared secret key with the party possessing that QR-PUF. Eve's goal is to learn the generated key. An impostor's goal is to convince Alice that he has real-time access to the QR-PUF even though he does not.

We make the following assumptions. Alice is honest. Bob may be acting in one of the following ways: (i) He is honest. (ii) He has access to the QR-PUF but does not hold it personally. He is in collusion with the party holding the QR-PUF. (iii) He has access to the QR-PUF but does not hold it personally. The party holding the QR-PUF is not cooperating with him. (iv) He does not have access to the QR-PUF.

In cases (i) and (ii) the protocol should result in authentication and a shared key, even though in case (ii) the secret key is shared between Alice and the PUF holder, while Bob may not even know the key. Case (iii) should result in authentication without a shared key. Case (iv) should not even result in authentication.

Eve has physical access to Challenge and Reflected states, but *not* to Transmitted states. She can destroy quantum states, perform measurements on them and insert new states. However, she cannot clone a state.

Again, we allow $R$ and $T$ to be public information. Creation or real-time emulation of a QR-PUF whose challenge-response matrix is $R$ is assumed infeasible. We make no such assumption about the $T$ matrix. Again, there is a tamper-proof source of enrollment data. We stress that the classical channel between Alice and Bob *does not have to be authenticated*.

### 4.2 Protocol Description

**System setup phase**

We make use of MACs with the *Key Manipulation Security* property (KMS-MAC) [7]. This is necessary since the classical messages are subject to manipulation attacks that influence the key. Use is made of two (publicly known) information-theoretically secure KMS-MACs $M_1 : \{0,1\}^{\ell_1} \times \{0,1\}^{m_1} \to \{0,1\}^{c_1}$ and $M_2 : \{0,1\}^{\ell_2} \times \{0,1\}^{m_2} \to \{0,1\}^{c_2}$. A MAC over message $x$ using key $k$ will be denoted as $M(k;x)$. A further public 'system parameter' is a universal [5] or almost-universal [25] hash function $F : \{0,1\}^N \times \{0,1\}^\sigma \to \{0,1\}^L$ which maps an $N$-bit string onto an $L$-bit string using $\sigma$ bits of randomness.

Finally there is an error-correcting code with messages in $\{0,1\}^N$ and codewords in $[n]^b$, which is chosen to have sufficient error-correcting capability to cope with the expected level of noise. The code word size $b$ is somewhat smaller than $m/2$, namely the expected number of rounds that yield a shared secret (corrected for particle loss). The encoding and decoding operations are denoted as `Code` and `Dec`. Addition modulo $n$ (with output in $[n]$) will be denoted as $\oplus$.

**Enrollment phase**

A QR-PUF is labeled with identifier $I$. The $R$ and $T$ matrix are accurately measured. The QR-PUF is given to Bob. Two observables $X_0$, $X_1$ are selected. They may be chosen depending on $R$ and $T$, or independently. The $k$th eigenstate of $X_\alpha$ is denoted as $|x_{\alpha k}\rangle$. The eigenstates satisfy $|\langle x_{0i}|x_{1j}\rangle|^2 = 1/n$ for all[5] $i, j \in [n]$. The $R$, $T$, $X_0$ and $X_1$ are public knowledge.

**Authentication and key exchange phase**

1. Alice fetches the enrollment data for PUF $I$. For $\alpha \in \{0,1\}$, $i \in [n]$ she initializes the following counters to zero: $m_{\alpha i}$ (for counting sent states), $g_{\alpha i}$ (returned states), $c_{\alpha i}$ (correct return states). She initializes a set $\mathcal{V}$ to $\emptyset$ (pointers to transmitted states). Alice and Bob both initialize a counter $t$ to zero.
2. The following steps are repeated $m$ times:
   (a) Alice and Bob increase $t$ by 1. Alice selects $\alpha \in \{0,1\}$ and $i \in [n]$ uniformly at random. She increases $m_{\alpha i}$. She stores $\alpha_t = \alpha$ and $i_t = i$. She prepares the state

$$|\psi_{\alpha i}\rangle := \frac{T^{-1}|x_{\alpha i}\rangle}{\sqrt{\langle x_{\alpha i}|(T^{-1})^\dagger T^{-1}|x_{\alpha i}\rangle}}$$

   and sends it to Bob. She performs a projection measurement $|\omega_{\alpha i}\rangle\langle\omega_{\alpha i}|$, with

$$|\omega_{\alpha i}\rangle := \frac{RT^{-1}|x_{\alpha i}\rangle}{\sqrt{\langle x_{\alpha i}|(T^{-1})^\dagger R^\dagger R T^{-1}|x_{\alpha i}\rangle}}. \tag{8}$$

[5] Such inner products can be achieved for instance by having

$$|x_{1k}\rangle = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} (e^{-i2\pi/n})^{kj} |x_{0j}\rangle \quad ; \quad |x_{0a}\rangle = \frac{1}{\sqrt{n}} \sum_{k=1}^{n} (e^{i2\pi/n})^{ka} |x_{1k}\rangle.$$

If she detects the existence of a return state, she increases $g_{\alpha i}$; else she adds $t$ to $\mathcal{V}$. If her projection measurement yields outcome '1', she increases $c_{\alpha i}$.

(b) Bob chooses a random bit $\beta$. He performs a measurement of $X_\beta$, obtaining either '$\perp$' (no transmitted state) or the $j$'th eigenvalue of $X_\beta$. If he gets $\perp$ then he stores $\beta_t = \perp$; else he stores $\beta_t = \beta$ and $j_t = j$.

3. Bob sends the vector $\boldsymbol{\beta}$ to Alice. Alice runs the following tests:
   (a) She checks how many states were lost ($t \in \mathcal{V} \wedge \beta_t = \perp$) and how many double counts occurred ($t \notin \mathcal{V} \wedge \beta_t \neq \perp$). If the number of either one of these occurrences is too high, then the noise level is considered too high and the protocol is aborted.
   (b) She checks if the transmission rates for all $\alpha, i$ were consistent with scattering by the QR-PUF. She does this by verifying if the vector $(1 - \frac{g_{01}}{m_{01}}, 1-\frac{g_{11}}{m_{11}}, \cdots, 1-\frac{g_{0n}}{m_{0n}}, 1-\frac{g_{1n}}{m_{1n}})$ is proportional to $(\tau_{01}, \tau_{11}, \cdots, \tau_{0n}, \tau_{1n})$, where the $\tau_{\alpha i}$ are the transmission probabilities,

$$\tau_{\alpha i} := \langle \psi_{\alpha i} | T^\dagger T | \psi_{\alpha i} \rangle = \frac{1}{\langle x_{\alpha i} | (T^\dagger T)^{-1} | x_{\alpha i} \rangle}. \tag{9}$$

   If there is a significant deviation then authentication has failed and the protocol is aborted.
   (c) For each $\alpha \in \{0,1\}$, $i \in [n]$ she checks if $c_{\alpha i}/g_{\alpha i} \geq (1-\varepsilon)$, where $\varepsilon$ is a small constant. If this is not the case, then authentication has failed and the protocol is aborted.
4. Alice compiles $\mathcal{Z} = \{t : t \in \mathcal{V} \wedge \alpha_t = \beta_t\}$ and randomly selects a subset $\mathcal{G} \subset \mathcal{Z}$ of size $b$. She generates random $a \in \{0,1\}^N$ and $y \in \{0,1\}^\sigma$. She computes $S = F(a, y)$ and parses $S$ as $S = k_1|k_2|S_{\text{rest}}$, with $k_1 \in \{0,1\}^{\ell_1}$ and $k_2 \in \{0,1\}^{\ell_2}$. She computes $w = \boldsymbol{\imath}_{\mathcal{G}} \oplus \texttt{Code}(a)$, $\mu_1 = M_1(k_1; \mathcal{G}, w, y)$, $\mu_2 = M_2(k_2; \boldsymbol{\beta})$. She sends $\mathcal{G}, w, y, \mu_1$.
5. Bob computes $a' = \texttt{Dec}(\boldsymbol{\jmath}_{\mathcal{G}} \oplus w)$ and $S' = F(a', y)$. He parses $S'$ as $S' = k_1'|k_2'|S'_{\text{rest}}$, with $k_1 \in \{0,1\}^{\ell_1}$ and $k_2' \in \{0,1\}^{\ell_2}$.
   He computes $\mu_1' = M_1(k_1'; \mathcal{G}, w, y)$, $\mu_2' = M_2(k_2'; \boldsymbol{\beta})$. He checks if $\mu_1 = \mu_1'$. If not, the protocol is aborted. He sends $\mu_2'$.
6. Alice checks if $\mu_2' = \mu_2$. If not, the protocol is aborted.

If the protocol does not abort then Alice and Bob have a noiseless shared secret $S_{\text{rest}} \in \{0,1\}^{L-\ell_1-\ell_2}$ about which Eve has negligible information, and Alice knows that she has generated this secret together with someone who has real-time access to the QR-PUF. Bob knows that he has a shared secret with someone who has sent challenges over the quantum channel, but has no further knowledge about this person.

### 4.3 Remarks

We have deliberately not specified what "too high" means in step 3a, and "significant deviation" in step 3b. We give no numbers for $m$, $b$, $N$, $L$, $\sigma$, $\ell_1$, $\ell_2$, $c_1$, $c_2$. These are design choices (threshold values etc.) that have been adequately

treated in the existing literature on QKE and KMS-MACs. We do not wish to further elaborate on such design choices in this paper.

The protocol may be modified in numerous ways without changing its essential properties. The classical communication between Alice and Bob may occur in a different order. The moment of checking the reflection rates may be shifted, etc.

As remarked before, if Alice and Bob want *mutual* authentication, the protocol can be run twice: first with Alice authenticating Bob's QR-PUF, then the other way round. The second time, the classical channel is already authenticated (there is a shared secret $S_{\text{rest}}$), which allows for ordinary information reconciliation and privacy amplification without the KMS-MACs. The two protocols may also be run intertwined, i.e. alternating their steps 2a,2b before proceeding to step 3. States may also be sent through both PUFs, but this will probably lead to more particle loss.

The KMS-MAC presented in [7] is secure against a 'linear' class of attacks on the MAC key, i.e. the attacker knows which change he is causing to the key, even though he does know the key itself. In our protocol an attack on the exchanged classical data gives no such knowledge to the attacker, since manipulation of the pointer sets $\mathcal{Z}$, $\mathcal{G}$ does not reveal $i_t$, $j_t$. Hence the construction in [7] is in fact overkill for our purposes.

We emphasize again that the authentication is in a sense reversed compared to 'standard' QKE: The quantum channel is authenticated without any use of the classical channel, and the data sent over the non-authenticated classical channel has to be checked for consistency with the exchanged quantum states.

### 4.4 Security of the Authenticated QKE Protocol

**Intercept-resend attacks on the authentication.** An impostor has to overcome several hurdles. The first hurdle is the correct mimicking of the transmission rates while he does not know $\alpha$ and $i$ accurately. This is nontrivial if the rates $\tau_{\alpha i}$ are substantially different. As in Section 3, he chooses an observable $B$, does a measurement and obtains an outcome $b_k$. His first choice is whether to return a fake reflected state or not. From his viewpoint (only the knowledge that the challenge state projected onto $|b_k\rangle$) the probability distribution of $\alpha$ and $i$ is

$$\Pr[\alpha, i|B, k] = \frac{\Pr[\alpha, i, B, k]}{\Pr[B, k]} = \frac{\Pr[\alpha, i, B, k]}{\sum_{\alpha,i} \Pr[\alpha, i, B, k]} = \frac{|\langle b_k|\psi_{\alpha i}\rangle|^2}{\sum_{\alpha,i} |\langle b_k|\psi_{\alpha i}\rangle|^2}. \tag{10}$$

It is interesting to note that there actually exists a strategy that allows him to correctly mimic all the transmission rates $\tau_{\alpha i}$. In general his strategy consists of a set of probabilities

$$Q_{Bk} := \Pr[\text{transmit}|B, k]. \tag{11}$$

When Alice sends challenge $|\psi_{\alpha i}\rangle$ he will transmit with probability

$$\Pr[\text{transmit}|\alpha, i] = \mathbb{E}_B \sum_{k\in[n]} |\langle b_k|\psi_{\alpha i}\rangle|^2 Q_{Bk} = \langle\psi_{\alpha i}| \left( \mathbb{E}_B \sum_{k\in[n]} Q_{Bk} |b_k\rangle\langle b_k| \right) |\psi_{\alpha i}\rangle. \tag{12}$$

For all $\alpha, i$ he wants this to be exactly equal to

$$\tau_{\alpha i} = \langle\psi_{\alpha i}|T^\dagger T|\psi_{\alpha i}\rangle = \langle\psi_{\alpha i}|\left(\sum_{k\in[n]} t_k|t_k\rangle\langle t_k|\right)|\psi_{\alpha i}\rangle. \tag{13}$$

Here we use the notation $t_k \in [0,1]$ for the $k$'th eigenvalue of the Hermitian operator $T^\dagger T$. (Note that the $t_k$ and $|t_k\rangle$ are public knowledge.) Comparing (13) to (12), it is clear that equality is obtained by setting $B = T^\dagger T$ and $Q_{Bk} = t_k$.

If the impostor makes any other choice, then Alice will notice that the transmission rates are wrong. Next we show that a measurement of $T^\dagger T$ does not give him enough information to guess $\alpha$ and $i$. We first give two useful lemmas, and then prove a bound (Theorem 4) on the per-round success probability.

**Lemma 1.** *Let $q_0, q_1 \in [0,1]$ be constants. Let $|v_0\rangle$ and $|v_1\rangle$ be two normalized states. Let $\lambda_{\max}$ denote the function that returns the maximum eigenvalue of a matrix. Then it holds that*

$$\lambda_{\max}\left(q_0|v_0\rangle\langle v_0| + q_1|v_1\rangle\langle v_1|\right) = \frac{1}{2}\left(q_0 + q_1 + \sqrt{(q_1-q_0)^2 + 4q_0q_1|\langle v_0|v_1\rangle|^2}\right).$$

**Lemma 2.** *Let $\Delta > 0$ be a constant. Let $|\gamma_0\rangle$ and $|\gamma_1\rangle$ be normalized states with $|\langle\gamma_0|\gamma_1\rangle|^2 = \Delta$. Let $|k\rangle$ be an arbitrary orthonormal basis. Then it holds that*

$$\sum_{k\in[n]} \frac{|\langle k|\gamma_0\rangle|^2 \cdot |\langle k|\gamma_1\rangle|^2}{|\langle k|\gamma_0\rangle|^2 + |\langle k|\gamma_1\rangle|^2} \geq \frac{\Delta}{2}.$$

**Theorem 4.** *Let the impostor use $B = T^\dagger T$, $Q_{Bk} = t_k$ as his strategy for the intercept-resend attack. Then his per-round probability of success for the authentication, whenever he decides to send a state to Alice, is bounded by*

$$p \leq \frac{1}{2} + \frac{1}{4n}\sum_{ik}\sqrt{(|\langle t_k|\psi_{1i}\rangle|^2 - |\langle t_k|\psi_{0i}\rangle|^2)^2 + 4|\langle t_k|\psi_{1i}\rangle|^2 \cdot |\langle t_k|\psi_{0i}\rangle|^2 \cdot |\langle\omega_{0i}|\omega_{1i}\rangle|^2}. \tag{14}$$

**Corollary 1.** *Let $a, \Delta > 0$ be constants. Let the scattering matrix be such that for all $i$:*

$$|\langle\psi_{0i}|\psi_{1i}\rangle|^2 \geq \Delta \quad \text{and} \quad |\langle\omega_{0i}|\omega_{1i}\rangle|^2 < 1 - a. \tag{15}$$

*Then Theorem 4 implies that*

$$p < 1 - \frac{a\Delta}{4}.$$

For proofs we refer to the full version [1].

In summary, the impostor has no choice but to do a measurement that is essentially equal to $T^\dagger T$; otherwise Alice will notice that the reflection rates are wrong. But the knowledge obtained from measuring $T^\dagger T$ is not sufficient to learn

enough about the challenge state $|\psi_{\alpha i}\rangle$, and the result is a success rate $p$ that noticeably differs from 100%.

In principle (14) allows $p$ to come close to 1. However, for that situation to occur the scattering matrix must be quite pathological. The eigenvectors of $X_1$ are maximally removed from those of $X_0$. Something special has to happen in order to have $|\omega_{0i}\rangle \propto RT^{-1}|x_{0i}\rangle$ align with $|\omega_{1i}\rangle \propto RT^{-1}|x_{1i}\rangle$ for some $i$, let alone for *all* $i$. A randomly created PUF is extremely unlikely to behave like that; and it can be discarded if it is ever created at all.

Note that the bound in Corollary 1 is not tight. Hence in practice the success probability is even lower.

**Quantum computer attack on the authentication.** Again, the authentication is vulnerable to an attack by Quantum Computer (QC). The attack is an extension of the one in Section 3.3. It needs [i] fast quantum teleportation, [ii] fast quantum computation and [iii] fast measurement of the state of the QC. The third requirement is new.

The QC's memory consists of two parts: Mem1 for the transmission and Mem2 for the reflection degrees of freedom. The challenge state $|\psi_{\alpha i}\rangle$ is moved into Mem1 by teleportation. Mem2 is initialized to zero. Then a unitary operation is applied that is equivalent to scattering by the QR-PUF, i.e. an operation is done equivalent to applying the scattering matrix $S$ to $\begin{pmatrix} |\psi_{\alpha i}\rangle \\ 0 \end{pmatrix}$. The result is a superposition $T|\psi_{\alpha i}\rangle \otimes 0 + 0 \otimes R|\psi_{\alpha i}\rangle$. (The tensor product refers to Mem1 and Mem2.) Then a measurement is done that only detects 'in which memory the particle is'. If the outcome is '2', then the state of the QC is $0 \otimes R|\psi_{\alpha i}\rangle$; The state of Mem2 is teleported onto a response state and sent to Alice. If the outcome is '1', then the state in Mem1 is used for QKE (either by applying an equivalent of $X_0$ or $X_1$ directly in the QC, or by teleporting the state of Mem1 out and then measuring $X_{0/1}$).

This attack correctly reproduces all reflection rates, reflected states and transmitted states. It allows an impostor to pass authentication without possessing Bob's QR-PUF and to generate a shared secret key with Alice.

However, the attack requires two fast teleportations, a powerful QC and a fast QC state measurement, which together will present a serious technological hurdle for quite some time to come.

**Security of the key.** If there is no 'quantum' attack and if Bob's QR-PUF is successfully authenticated, then from Alice's point of view the key is secure. The argument is straightforward. The authentication can be spoofed only by a quantum attack. Given that there is no quantum attack, successful authentication implies that Bob really controls the QR-PUF. Hence Alice is truly generating a key with Bob. With impostors having been excluded, we only have to worry about eavesdropping. The transmitted states travel through the QR-PUF in a 'transparent' way,[6] i.e. Alice's $T^{-1}$ in the challenge states $T^{-1}|x_{\alpha i}\rangle$ undoes the

[6] The presence of the QR-PUF may introduce some additional noise. It is well known how to deal with noise.

effect of transmission through the QR-PUF, and effectively she sends eigenstates of $X_0$ or $X_1$ to Bob. Thus, the protocol rounds in which transmission occurs are completely equivalent to an ordinary QKE scheme, and all the standard security proofs for QKE apply.

**Security trade-offs for authenticated QKE.** Below we list the security properties of three authentication methods for QKE. We find it interesting that the QR-PUF authentication achieves an unusual type of trade-off: it has no need for the a priori sharing of a secret, but the security depends on physical assumptions.

| **Auth. method** | **Security assumption** | **Remarks** |
|---|---|---|
| MAC | unconditional | needs a priori shared secret |
| entangled state | unconditional | needs a priori shared secret state; unpractical because of decoherence. |
| QR-PUF | physical unclonability; no quantum emulation | needs trusted enrollment data (allowed to be public) |

## 5 Summary and Future Work

We have introduced a new security primitive, the Quantum Readout PUF (QR-PUF), which is a classical PUF that can be read out using quantum states. We have shown how QR-PUFs can be used to achieve remote authentication without a trusted remote reader device. The security is based on two well known physical assumptions, physical unclonability and uniqueness, and one new physical assumption, quantum-computational unclonability. The no-cloning theorem guarantees that intercept-resend attacks will be detected. Our authentication scheme is vulnerable to a three-step attack employing quantum teleportation and a quantum computer. For this reason we need the assumption of quantum-computational unclonability, which states that this kind of attack, while possible in theory, is infeasible in practice because of technical or financial issues. What makes a 'quantum' attack especially difficult is the fact that our protocol doubles as a distance bounding scheme; all steps of the attack have to be extremely fast.

We have sketched how QR-PUF authentication can be intertwined with Quantum Key Exchange. Reflected states are used for authentication, transmitted states for QKE. This combination achieves authenticated QKE without the need for an initial shared secret (such as a short MAC key or an entangled state). The sketched protocol has the unusual property that the quantum channel is authenticated, allowing for an un-authenticated classical channel. This reversal necessitates the use of KMS-MACs.

In our schemes Alice waits for a returning state before sending her next challenge state; this simplifies the security proofs considerably, since in this setting it suffices to look at the security of an isolated round without having to worry about (entanglement) attacks on multiple challenge states. We leave more general protocols and their security proof as a subject for future work.

We have not discussed implementations. The biggest question is how to construct a QR-PUF in the first place. The most practical option seems to be a partially transmissive optical PUF that is challengeable by single photon states or coherent states through an optical fiber (or open air). The main difficulty is to make sure that the uniqueness and physical unclonability properties hold, in spite of the limited number of challenges that can be passed through a fiber. Fibers carry a limited number of transversal modes, while such modes are the main way of challenging an ordinary speckle-producing optical PUF [21,22]. We are perhaps aided by the fact that multiple wavelengths are available as a challenge.

Another question is how to quantify the difficulty of the 'quantum' attack, the most serious threat to QR-PUFs. Here too the availability of different wavelengths seems to help us, increasing the required size of the quantum computer.

Our protocols can be extended in a number of obvious ways. For instance, EPR pairs can be used, as well as anti-eavesdropping countermeasures like 'decoy states' [15]. The QR-PUF can be used for Quantum Oblivious Transfer. Another option is transmitting states through more than one QR-PUF. It would also be interesting to see if one can construct a 'quantum PUF', i.e. a PUF that has actual quantum behaviour, resulting in nontrivial (even nonlinear) interaction between the challenge state and the PUF.

## Acknowledgements

We kindly thank Pim Tuyls, Berry Schoenmakers, Allard Mosk and Andrea Fiore for useful discussions.

## References

1. Full version, available at Cryptology ePrint Archive, Report 2009/369 (2009)
2. Bennett, C.H., Brassard, G.: Quantum cryptography: Public key distribution and coin tossing. In: IEEE Int. Conf. on Computers, Systems and Signal Processing, pp. 175–179 (1984)
3. Bennett, C.H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A., Wootters, W.K.: Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. Phys. Rev. Lett. 70, 1895–1899 (1993)
4. Buchanan, J.D.R., Cowburn, R.P., Jausovec, A., Petit, D., Seem, P., Xiong, G., Atkinson, D., Fenton, K., Allwood, D.A., Bryan, M.T.: Forgery: 'fingerprinting' documents and packaging. Nature, Brief Communications 436, 475 (2005)
5. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. J. of Computer and System Sciences 18(2), 143–154 (1979)
6. Chong, C.N., Jiang, D., Zhang, J., Guo, L.: Anti-counterfeiting with a random pattern. In: SECURWARE 2008, pp. 146–153. IEEE, Los Alamitos (2008)
7. Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 471–488. Springer, Heidelberg (2008)
8. DeJean, G., Kirovski, D.: Radio frequency certificates of authenticity. In: IEEE Antenna and Propagation Symposium – URSI (2006)

9. Dieks, D.: Phys. Lett. A, 92, 271 (1982)
10. Duan, L.M., Lukin, M., Cirac, J.I., Zoller, P.: Long-distance quantum communication with atomic ensembles and linear optics. Nature 414, 413–418 (2001)
11. Ekert, A.K.: Quantum cryptography based on Bells theorem. Phys. Rev. Lett. 67, 661–663 (1991)
12. Gassend, B., Clarke, D.E., van Dijk, M., Devadas, S.: Silicon physical unknown functions. In: ACM Conf. on Computer and Communications Security – CCS 2002, November 2002, pp. 148–160 (2002)
13. Gottesman, D., Preskill, J.: Secure quantum key exchange using squeezed states, arXiv:quant-ph/0008046v2 (2000)
14. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA intrinsic PUFs and their use for IP protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
15. Hwang, W.-Y.: Quantum key distribution with high loss: Toward global secure communication. Phys. Rev. Lett. 91, 057901 (2003)
16. Inoue, K., Waks, E., Yamamoto, Y.: Differential phase shift quantum key distribution. Phys. Rev. Lett. 89, 037902 (2002)
17. Kirovski, D.: Toward an automated verification of certificates of authenticity. In: ACM Conference on Electronic Commerce, pp. 160–169. ACM, New York (2004)
18. Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.J., Tuyls, P.: The Butterfly PUF: Protecting IP on every FPGA. In: HOST 2008, pp. 67–70. IEEE, Los Alamitos (2008)
19. Kursawe, K., Sadeghi, A.-R., Schellekens, D., Škorić, B., Tuyls, P.: Reconfigurable physical unclonable functions. In: HOST 2009 (2009)
20. Matsukevich, D.N., Kuzmich, A.: Quantum state transfer between matter and light. Science 306(5696), 663–666 (2004)
21. Pappu, R.: Physical One-Way Functions. PhD thesis, MIT (2001)
22. Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical One-Way Functions. Science 297, 2026–2030 (2002)
23. Renner, R.: Security of Quantum Key Distribution. PhD thesis, ETH Zurich (2005)
24. Shi, B.S., Li, J., Liu, J.M., Fan, X.F., Guo, G.C.: Quantum key distribution and quantum authentication based on entangled state. Phys. Lett. A 281, 83–87 (2001)
25. Stinson, D.R.: Universal hashing and authentication codes. Designs, Codes, and Cryptography 4, 369–380 (1994)
26. Tuyls, P., Schrijen, G.J., Škorić, B., van Geloven, J., Verhaegh, R., Wolters, R.: Read-proof hardware from protective coatings. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 369–383. Springer, Heidelberg (2006)
27. Tuyls, P., Škorić, B., Kevenaar, T.: Security with Noisy Data: Private Biometrics, Secure Key Storage and Anti-Counterfeiting. Springer, London (2007)
28. Tuyls, P., Škorić, B., Stallinga, S., Akkermans, A.H.M., Ophey, W.: Information-theoretic security analysis of physical uncloneable functions. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 141–155. Springer, Heidelberg (2005)
29. Škorić, B.: On the entropy of keys derived from laser speckle; statistical properties of Gabor-transformed speckle. Journal of Optics A: Pure and Applied Optics 10(5), 055304–055316 (2008)
30. Škorić, B., Bel, T., Blom, A.H.M., de Jong, B.R., Kretschman, H., Nellissen, A.J.M.: Randomized resonators as uniquely identifiable anti-counterfeiting tags. In: Secure Component and System Identification Workshop, Berlin (March 2008)
31. Wootters, W.K., Zurek, W.H.: A single quantum cannot be cloned. Nature 299, 802–803 (1982)

# Parallelizing the Camellia and SMS4 Block Ciphers

Huihui Yap[1,2], Khoongming Khoo[1,2], and Axel Poschmann[2]

[1] DSO National Laboratories, 20 Science Park Drive, Singapore 118230
[2] Division of Mathematical Sciences, School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
{yhuihui,kkhoongm}@dso.org.sg, aposchmann@ntu.edu.sg

**Abstract.** The $n$-cell GF-NLFSR (Generalized Feistel-NonLinear Feedback Shift Register) structure [8] is a generalized unbalanced Feistel network that can be considered as a generalization of the outer function $FO$ of the KASUMI block cipher. An advantage of this cipher over other $n$-cell generalized Feistel networks, e.g. SMS4 [11] and Camellia [5], is that it is parallelizable for up to $n$ rounds. In hardware implementations, the benefits translate to speeding up encryption by up to $n$ times while consuming less area and power. At the same time $n$-cell GF-NLFSR structures offer similar proofs of security against differential cryptanalysis as conventional $n$-cell Feistel structures. We also ensure that parallelized versions of Camellia and SMS4 are resistant against other block cipher attacks such as linear, boomerang, integral, impossible differential, higher order differential, interpolation, slide, XSL and related-key differential attacks.

**Keywords:** Generalized Unbalanced Feistel Network, GF-NLFSR, Camellia, SMS4.

## 1 Introduction

### 1.1 Background and Motivation

Two very important security properties of block cipher structures are low differential and linear probability bounds for protection against differential and linear cryptanalysis. Choy et al. [8] had proven that the "true" differential/linear probabilities of any $n$ rounds of the $n$-cell GF-NLFSR structure is $p^2$ if the differential/linear probability of the nonlinear function of each round is $p$. However, this result is applicable only if we use a nonlinear function with good provable differential/linear probability. One option is to use an S-box. However if the nonlinear function takes in 32-bit input, an S-box of this size would be infeasible to implement in terms of logic gates in hardware or as a look-up-table in memory. Other options would be to build a SDS (Substitution-Diffusion-Substitution) structure [20], use a Feistel structre [2] or even a nested Feistel structure for the nonlinear function [3], because there are provable bounds for the differential and linear probabilities of these structures.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 387–406, 2010.

However these nonlinear functions are too complex, and not suitable for space and speed efficient implementation. Therefore, the Substitution-then-Diffusion structure is usually implemented for the nonlinear functions. These structures are commonly called Substitution Permutation Networks (SPN) in the literature. Numerous examples of implementations where the SPN structure is used for the nonlinear functions of Feistel and Generalized Feistel Structures exist. They include DES [1], Camellia [5], SMS4 [11] and Clefia [21], to name a few. Motivated by these considerations, we would like to investigate the practical differential and linear probability bounds of the $n$-cell GF-NLFSR structure when the nonlinear function is a SPN structure.

As applications, we would like to parallelize some of the abovementioned ciphers, where we replace the (Generalized) Feistel structures by the parallelizable GF-NLFSR structures, while keeping the internal components like S-boxes and linear diffusion to be the same. This would make encryption speed faster by up to $n$ times. Two candidates which we find promising for parallelizing are the Camellia and SMS4 ciphers.

### 1.2 Related Works

In order to analyze the resistance of a block cipher against differential and linear cryptanalysis, we would like to lower bound the number of active S-boxes (S-boxes which contribute to the differential/linear probability) in any differential/linear characteristic path over a fixed number of rounds. Using such bounds, the cipher designer can choose a large enough number of rounds so that there are too many active S-boxes for differential/linear cryptanalysis to be successful.

Kanda [16] has proven that for a Feistel cipher with an SPN round function having branch number $\mathcal{B}$ (a measure of dispersion, please refer to Section 2 for the exact definition), the number of active S-boxes in any differential and linear characteristic path over every $4r$ rounds is at least $r\mathcal{B} + \lfloor \frac{r}{2} \rfloor$. Based on this lower bound, the authors of [5] designed the block cipher Camellia, which has practical provable security against differential and linear cryptanalysis.

### 1.3 Our Contribution

In Section 3, we provide a neat and concise proof of the result that for a $2nr$-round parallelizable $n$-cell GF-NLFSR structure with an SPN round function having branch number $\mathcal{B}$, the number of active S-boxes in any differential characteristic path is at least $r\mathcal{B} + \lfloor \frac{r}{2} \rfloor$. The result holds for any $n \geq 2$ in general, and we expect the result to be useful in the design and analysis of block cipher structures. For the case of a 2-cell GF-NLFSR structure, we have $r\mathcal{B} + \lfloor \frac{r}{2} \rfloor$ active S-boxes over every $4r$ rounds, which is the same as Kanda's result [16] for a conventional 2-cell Feistel structure. Motivated by this observation, we propose in Section 4 a parallelizable version of Camellia, p-Camellia, where we change the conventional Feistel structure to a 2-cell GF-NLFSR structure but keep all other components such as S-boxes and linear diffusion maps to be the same. We also ensure p-Camellia is secure against other cryptanalysis such as

linear, boomerang, integral, impossible differential, higher order differential, interpolation and slide attacks. In addition, we assess the advantages of hardware implementations. For this reason we briefly introduce design strategies for hardware implementations. We then show that especially for applications with high throughput requirements, a 2-cell GF-NLFSR such as p-Camellia offers significant advantages over a conventional 2-cell Feistel structure such as Camellia. In particular, we show that an implementation of p-Camellia that processes two rounds in parallel has a maximum frequency that is twice as high as it would be for Camellia while having lower area and power demands. We also show that for fully pipelined implementations a conventional 2-cell Feistel structure requires twice as many pipeline stages, and hence twice as many clock cycles delay, to achieve the same frequency as it is the case for a 2-cell GF-NLFSR.

In Section 5, we also apply a 4-cell GF-NLFSR structure to form a parallelizable version of SMS4 called p-SMS4. We change the generalized Feistel structure in both the main cipher and key schedule of SMS4 to a 4-cell GF-NLFSR structure but keep all other components such as S-boxes and linear diffusion maps to be the same. We first prove that p-SMS4 is secure against differential cryptanalysis. In [7], Biryukov et al. showed a powerful related-key differential attack on AES-256 which can recover the secret key with complexity $2^{131}$ using $2^{35}$ related keys. We give a proof through the p-SMS4 key schedule that p-SMS4 is resistant against this attack. We also ensure p-SMS4 is secure against other block cipher cryptanalysis such as boomerang, integral, impossible differential, higher order differential, interpolation, slide and XSL attacks. A 4-cell GF-NLFSR structure offers also implementation advantages for round-based and parallelized hardware architectures. We show that a round-based 4-cell GF-NLFSR structure has a shorter critical path, and hence a higher maximum frequency, than a conventional 4-cell Feistel structure. In parallelized implementations this advantage increases to a four times higher maximum frequency while having lower area and power demands. In general the advantage is dependent on the number of branches, hence an $n$-cell GF-NLFSR has an advantage of an $n$ times higher maximum frequency.

## 2 Definitions and Preliminaries

In this section, we will list some definitions and summarize the results of Kanda in [16]. He has proven the upper bounds of the maximum differential and linear characteristic probabilities of Feistel ciphers with bijective SPN round function. More explicitly, the round function $F$-function comprises the key addition layer, the $S$-function and the $P$-function. Here we neglect the effect of the round key since by assumption, the round key, which is used within one round, consists of independent and uniformly random bits, and is bitwise XORed with data. The $S$-function is a non-linear transformation layer with $m$ parallel $d$-bit bijective S-boxes whereas the $P$-function is a linear transformation layer. In particular, we have

$S : (GF(2^d)^m \rightarrow (GF(2^d)^m\ , X = (x_1, \cdots, x_m) \mapsto Z = S(X) = (s_1(x_1), \cdots, s_n(x_n)),$
$P : (GF(2^d)^m \rightarrow (GF(2^d)^m\ , Z = (z_1, \cdots, z_m) \mapsto Y = P(Z) = (y_1, \cdots, y_n),$
$F : (GF(2^d)^m \rightarrow (GF(2^d)^m\ , X \mapsto Y = F(X) = P(S(X)).$

**Definition 1.** *Let $x, z \in GF(2^d)$. Denote the differences and the mask values of $x$ and $z$ by $\Delta x$, $\Delta z$, and, $\Gamma x$, $\Gamma z$ respectively. The differential and linear probabilities of each S-box $s_i$ are defined as:*

$$DP^{s_i}(\Delta x \rightarrow \Delta z) = \frac{\#\{x \in GF(2^d) | s_i(x) \oplus s_i(x \oplus \Delta x) = \Delta z\}}{2^d},$$

$$LP^{s_i}(\Gamma z \rightarrow \Gamma x) = (2 \times \frac{\#\{x \in GF(2^d) | x \cdot \Gamma x = s_i(x) \cdot \Gamma z}{2^d} - 1)^2.$$

**Definition 2.** *The maximum differential and linear probabilities of S-boxes are defined as:*

$$p_s = \max_i \max_{\Delta x \neq 0, \Delta z} DP^{s_i}(\Delta x \rightarrow \Delta z),$$

$$q_s = \max_i \max_{\Gamma x, \Gamma z \neq 0} LP^{s_i}(\Gamma z \rightarrow \Gamma x).$$

*This means that $p_s, q_s$ are the upper bounds of the maximum differential and linear probabilities for all S-boxes.*

**Definition 3.** *Let $X = (x_1, x_2, \cdots, x_m) \in GF(2^d)^m$. Then the Hamming weight of $X$ is denoted by $H_w(X) = \#\{i | x_i \neq 0\}$.*

**Definition 4.** *[23] The branch number $\mathcal{B}$ of linear transformation $\theta$ is defined as follows:*

$$\mathcal{B} = \min_{x \neq 0}(H_w(x) + H_w(\theta(x)).$$

Consider Feistel ciphers with bijective SPN round functions as described previously. As mentioned in [16], for the differential case, $\mathcal{B}$ is taken to be the *differential* branch number, i.e. $\mathcal{B} = \min_{\Delta X \neq 0}(H_w(\Delta X) + H_w(\Delta Y))$, where $\Delta X$ is an input difference into the $S$-function and $\Delta Y$ is an output difference of the $P$-function. On the other hand, for the linear case, $\mathcal{B}$ is taken to be the *linear* branch number, i.e. $\mathcal{B} = \min_{\Gamma Y \neq 0}(H_w(P^*(\Gamma Y)) + H_w(\Gamma Y))$, where $\Gamma Y$ is an output mask value of the $P$-function and $P^*$ is a diffusion function of mask values concerning the $P$-function. Throughout this paper, $\mathcal{B}$ is used to denote differential or linear branch number, depending on the context.

**Definition 5.** *A differential active S-box is defined as an S-box given a non-zero input difference. Similarly, a linear active S-box is defined as an S-box given a non-zero output mask value.*

**Theorem 1.** *Let $\mathcal{D}^{(r)}$ and $\mathcal{L}^{(r)}$ be the minimum number of all differential and linear active S-boxes for a $r$-round Feistel cipher respectively. Then the maximum and linear characteristic probabilities of the $r$-round cipher are bounded by $p_s^{D^{(r)}}$ and $q_s^{L^{(r)}}$ respectively.*

Note that Theorem 1 applies to any block cipher in general.

**Theorem 2.** *[16] The minimum number of differential (and linear) active S-boxes $\mathcal{D}^{(4r)}$ for $4r$-round Feistel ciphers with SPN round function is at least $r\mathcal{B} + \lfloor \frac{r}{2} \rfloor$.*

## 3 Practical Security Evaluation of GF-NLFSR against Differential and Linear Cryptanalysis

GF-NLFSR was proposed by Choy et al. in [8]. It is an $n$-cell extension of the outer function $FO$ of the KASUMI block cipher which is a 2-cell structure [8].

Throughout this paper, we consider GF-NLFSR block ciphers with SPN (S-P) round function, as described in Section 1.2. In this paper, we assume that both the $S$-function and $P$-function are bijective.

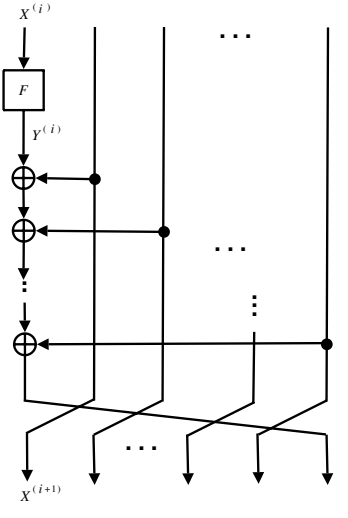



**Fig. 1.** $i$-th round of GF-NLFSR

With reference to Figure 1, let $X^{(i)}$ and $Y^{(i)}$ be the input and output data to the $i$-th round function respectively. Then the GF-NLFSR block cipher can be defined as

$$X^{(i+n)} = Y^{(i)} \oplus X^{(i+1)} \oplus X^{(i+2)} \oplus \cdots \oplus X^{(i+n-1)}, \text{ for } i = 1, 2, \cdots . \quad (1)$$

From equation (1), it can be shown almost immediately that there must be at least 2 differential active S-boxes over $(n+1)$-round of $n$-cell GF-NLFSR cipher.

**Proposition 1.** *The minimum number of differential active S-boxes for $(n+1)$-round $n$-cell GF-NLFSR cipher with bijective SPN round function satisfies $\mathcal{D}^{(n+1)} \geq 2$.*

*Proof.* Without loss of generality, we assume that the $n+1$ consecutive rounds run from the first round to the $(n+1)$-th round. Since the SPN round function is bijective, $\Delta Y^{(1)} = 0$ if and only if $\Delta X^{(1)} = 0$. From equation (1), we have

$$\Delta X^{(n+1)} = \Delta Y^{(1)} \oplus \Delta X^{(2)} \oplus \Delta X^{(3)} \oplus \cdots \oplus \Delta X^{(n)}, \tag{2}$$

from which it follows that there must exist at least two non-zero terms in equation (2) in order for equation (2) to hold. Therefore

$$\mathcal{D}^{(n+1)} = Hw(\Delta X^{(1)}) + \cdots + Hw(\Delta X^{(n+1)}) \geq 2. \qquad \square$$

**Lemma 1.** *Let* $X = (x_1, x_2, \cdots, x_m)$ *and* $X' = (x'_1, x'_2, \cdots, x'_m) \in GF(2^d)^m$. *Then*

$$H_w(X \oplus X') \leq H_w(X) + H_w(X').$$

*Proof*

$$\begin{aligned}
&H_w(X \oplus X') \\
&= \#\{s | x_s \neq 0 \text{ and } x'_s = 0\} + \#\{t | x_t = 0 \text{ and } x'_t \neq 0\} \\
&+ \#\{u | x_u \neq 0 \text{ and } x'_u \neq 0 \text{ and } x_u \neq x'_u\} \\
&\leq H_w(X) + \#\{t | x_t = 0 \text{ and } x'_t \neq 0\} \\
&\leq H_w(X) + H_w(X') \qquad \square
\end{aligned}$$

Lemma 2 is a straightforward generalization of Lemma 1.

**Lemma 2.** *Let* $X_1, X_2, \cdots, X_k \in GF(2^d)^m$. *Then*

$$H_w(X_1 \oplus X_2 \oplus \cdots \oplus X_k) \leq H_w(X_1) + H_w(X_2) + \cdots + H_w(X_k).$$

As stated in Theorem 1, to investigate the upper bound of the maximum differential characteristic probability of the GF-NLFSR cipher, we need to find a lower bound for $\mathcal{D}^{(r)}$, the number of differential active S-boxes for $r$ consecutive rounds of the cipher. Then the differential characteristic probability of the $r$-round GF-NLFSR cipher is at most $p_s^{\mathcal{D}^{(r)}}$.

**Lemma 3.** *For n-cell GF-NLFSR cipher, the minimum number of differential active S-boxes in any* $2n$ *consecutive rounds satisfies* $\mathcal{D}^{(2n)} \geq \mathcal{B}$.

*Proof.* Without loss of generality, we assume that the $2n$ consecutive rounds run from the first round to the $2n$-th round. For $j = 1, \cdots, n$, note that at least one of $\Delta X^{(j)} \neq 0$. Let $i$ be the smallest integer such that $\Delta X^{(i)} \neq 0$, where $1 \leq i \leq n$. Then

$$\begin{aligned}
\mathcal{D}^{(2n)} &= H_w(\Delta X^{(1)}) + H_w(\Delta X^{(2)}) + \cdots + H_w(\Delta X^{(2n)}) \\
&\geq H_w(\Delta X^{(i)}) + H_w(\Delta X^{(i+1)}) \cdots + H_w(\Delta X^{(i+n)}) \\
&\geq H_w(\Delta X^{(i)}) + H_w(\Delta X^{(i+1)} \oplus \cdots \oplus \Delta X^{(i+n)}), \text{ by Lemma 2,} \\
&= H_w(\Delta X^{(i)}) + H_w(\Delta Y^{(i)}) \\
&\geq \mathcal{B}. \qquad \square
\end{aligned}$$

*Remark 1.* From the above proof, we see that with probability $1 - \frac{1}{M}$, where $M$ is the size of each cell, i.e. most of the time, we have $\Delta X^{(1)} \neq 0$. In that case, we are able to achieve at least $\mathcal{B}$ number of differential active S-boxes over $(n+1)$-round of $n$-cell GF-NLFSR cipher.

As a consequence of Lemma 3 and using a similar approach as [16], we have the following result.

**Theorem 3.** *The minimum number of differential active S-boxes for* $2nr$*-round* $n$*-cell GF-NLFSR cipher with bijective SPN round function satisfies*

$$\mathcal{D}^{(2nr)} \geq r\mathcal{B} + \lfloor \frac{r}{2} \rfloor.$$

In particular, when $n = 2$, the minimum number of differential active S-boxes for $4r$-round 2-cell GF-NLFSR cipher with bijective SPN round function is at least $r\mathcal{B} + \lfloor \frac{r}{2} \rfloor$. Hence we see that 2-cell GF-NLFSR cipher with bijective SPN round function has similar practical security against differential cryptanalysis as Feistel ciphers with bijective SPN round functions. Moreover, 2-cell GF-NLFSR has an added advantage that it realizes parallel computation of round functions, thus providing strong motivation for parallelizing ciphers with SPN round functions, as described in the next section. To conclude this section, we shall investigate the practical security of 2-cell GF-NLFSR cipher against linear cryptanalysis. Again from Theorem 1, to investigate the upper bound of the maximum linear characteristic probability of the 2-cell GF-NLFSR cipher, we need to find a lower bound for $\mathcal{L}^{(r)}$, the number of linear active S-boxes for $r$ consecutive rounds of the cipher. Then the linear characteristic probability of the $r$-round cipher is at most $q_s^{\mathcal{L}^{(r)}}$.

**Lemma 4.** *For* 2*-cell GF-NLFSR cipher with bijective SPN round function and linear branch number* $\mathcal{B} = 5$*, the minimum number of linear active S-boxes in any* 4 *consecutive rounds satisfies* $\mathcal{L}^{(4)} \geq 3$*.*

*Proof.* Let the input and output mask values to the $i$-th round $F$ function be $\Gamma X^{(i)}$ and $\Gamma Y^{(i)}$ respectively. Note that since the $F$ function is bijective, $\Gamma X^{(i)} = 0$ if and only if $\Gamma Y^{(i)} = 0$. Without loss of generality, we assume that the 4 consecutive rounds run from the first round to the 4th round. Thus the minimum number of linear active S-boxes over 4 consecutive rounds is given by

$$\mathcal{L}^{(4)} = H_w(\Gamma Y^{(1)}) + H_w(\Gamma Y^{(2)}) + H_w(\Gamma Y^{(3)}) + H_w(\Gamma Y^{(4)}).$$

From the duality between differential characteristic and linear approximation, we have

$$\Gamma Y^{(i+1)} = \Gamma X^{(i-1)} \oplus \Gamma X^{(i)},$$

for $i = 2$ and 3. We consider all cases as follows, where $\mathcal{L}_i^{(r)}$ denotes the number of linear active S-boxes over $r$ rounds for case $i$:

Case 1: $\Gamma Y^{(1)} = 0$
This implies that $\Gamma Y^{(2)} \neq 0$ and $\Gamma Y^{(3)} = \Gamma X^{(2)}$. Hence $\mathcal{L}_1^{(3)} \geq H_w(\Gamma Y^{(2)}) + H_w(\Gamma Y^{(3)}) = H_w(\Gamma Y^{(2)}) + H_w(\Gamma X^{(2)}) \geq \mathcal{B} = 5 \geq 3$. Thus $\mathcal{L}_1^{(4)} \geq \mathcal{L}_1^{(3)} \geq 3$.

Case 2: $\Gamma Y^{(1)} \neq 0$ and $\Gamma Y^{(2)} = 0$
This implies that $\Gamma Y^{(3)} = \Gamma X^{(1)}$. Hence $\mathcal{L}_2^{(3)} \geq H_w(\Gamma Y^{(1)}) + H_w(\Gamma Y^{(3)}) = H_w(\Gamma Y^{(1)}) + H_w(\Gamma X^{(1)}) \geq \mathcal{B} = 5 \geq 3$. Thus $\mathcal{L}_2^{(4)} \geq \mathcal{L}_2^{(3)} \geq 3$.

Case 3: $\Gamma Y^{(1)} \neq 0$, $\Gamma Y^{(2)} \neq 0$ and $\Gamma Y^{(3)} = 0$
This implies that $\Gamma Y^{(4)} = \Gamma X^{(2)}$. Hence $\mathcal{L}_3^{(4)} \geq H_w(\Gamma Y^{(1)}) + H_w(\Gamma Y^{(2)}) + H_w(\Gamma Y^{(4)}) = H_w(\Gamma Y^{(1)}) + H_w(\Gamma Y^{(2)}) + H_w(\Gamma X^{(2)}) \geq 1 + \mathcal{B} = 6 \geq 3$.

Case 4: $\Gamma Y^{(1)} \neq 0$, $\Gamma Y^{(2)} \neq 0$, $\Gamma Y^{(3)} \neq 0$ and $\Gamma Y^{(4)} = 0$
Then we obtain $\mathcal{L}_4^{(4)} \geq H_w(\Gamma Y^{(1)}) + H_w(\Gamma Y^{(2)}) + H_w(\Gamma Y^{(3)}) \geq 1 + 1 + 1 = 3$.

Case 5: $\Gamma Y^{(1)} \neq 0$, $\Gamma Y^{(2)} \neq 0$, $\Gamma Y^{(2)} \neq 0$ and $\Gamma Y^{(4)} \neq 0$
Then we obtain $\mathcal{L}_5^{(4)} = H_w(\Gamma Y^{(1)}) + H_w(\Gamma Y^{(2)}) + H_w(\Gamma Y^{(3)}) + H_w(\Gamma Y^{(3)}) \geq 1 + 1 + 1 + 1 = 4 \geq 3$ .
Therefore $\mathcal{L}^{(4)} \geq 3$. □

**Theorem 4.** *For 2-cell GF-NLFSR cipher with bijective SPN round function and linear branch number $\mathcal{B} = 5$, we have*

(1) $\mathcal{L}^{(8)} \geq 7$,
(2) $\mathcal{L}^{(12)} \geq 11$,
(3) $\mathcal{L}^{(16)} \geq 15$.

*Proof.* Without loss of generality, we begin from the first round.

(1) From the proof of Lemma 4, over 8 rounds, we only need to check the case for $\Gamma Y^{(1)} \neq 0$, $\Gamma Y^{(2)} \neq 0$, $\Gamma Y^{(3)} \neq 0$ and $\Gamma Y^{(4)} = 0$. (In all remaining cases, there will be at least 7 linear active S-boxes over 8 rounds.) However $\Gamma Y^{(3)} \neq 0$ and $\Gamma Y^{(4)} = 0$ correspond to Case 1 of Lemma 4 for the four consecutive rounds that begin from the 4th round and end after the 7th round. Hence there will be at least $3+5 = 8$ linear active S-boxes. Therefore $\mathcal{L}^{(8)} \geq 7$.
(2) From (i), over 12 rounds, we only need to consider the case for $\Gamma Y^{(i)} \neq 0$ for $i = 1, \cdots, 7$ and $\Gamma Y^{(8)} = 0$. Following a similar argument to (i), we are definitely ensured of at least $7 + 5 = 12$ linear active S-boxes. Hence $\mathcal{L}^{(12)} \geq 11$.
(3) The proof is similar to that of (i) and (ii). □

## 4 Application 1: Parallelizing Camellia

### 4.1 Brief Description of Camellia

Camellia was jointly developed by NTT and Mitsubishi Electric Corporation. According to [5], Camellia uses an 18-round Feistel structure for 128-bit key,

and a 24-round Feistel structure for 192-bit and 256-bit keys, with additional input/output whitenings and logical functions called the $FL$-function and $FL^{-1}$-function inserted every 6 rounds. Its $F$-function uses the SPN (Substitution-Permutation Network) structure, whereby the non-linear layer comprises eight S-boxes in parallel while the linear layer can be represented using only bytewise exclusive-ORs. Note that the $F$-function is bijective.

For security against differential and linear cryptanalysis, the branch number of the linear layer should be optimal, i.e. branch number = 5. In addition, the S-boxes adopt functions affine equivalent to the inversion function in $GF(2^8)$ which achieves the best known of the maximum differential and linear probabilities $2^{-6}$ [5].

The key schedule of Camellia is slightly different for the 128-bit key version and the 192-bit/256-bit key version. Despite the slight differences, the key schedule is relatively simple and consists of two main steps. One (or two) 128-bit subkey materials are first derived from the secret key via some Feistel network. The round keys are then generated by rotating the secret key itself and the derived subkeys by various amounts.

For more details of the structure of Camellia, readers are referred to [4].

### 4.2 Parallelizing Camellia: p-Camellia

In this section, we propose another version of the existing Camellia block cipher, which we call p-Camellia (*"parallelizable" Camellia*). As described previously, Camellia uses a Feistel network structure. For the encryption procedure of p-Camellia, we shall replace the Feistel network with the 2-cell GF-NLFSR block cipher structure instead, as depicted in Figure 4 of Appendix. Other components such as number of rounds, $S$-function, $P$-function and the key schedule etc remain unchanged. In addition, similar to Camellia, there are input/output whitenings which are represented by the XOR symbols at the beginning/end of p-Camellia cipher in Figure 4 of Appendix.

### 4.3 Differential and Linear Cryptanalysis of p-Camellia

Following the same approach in [4], denote the maximum differential and linear characteristic probabilities of p-Camellia reduced to 16-round by $p$ and $q$ respectively. Recall that since both p-Camellia and Camellia use the same $F$-function, in the case of p-Camellia, the maximum differential and linear probability of the S-boxes are $2^{-6}$. From [4], the differential branch numbers is equal to 5. By considering the $P^*$-function of Camellia as in [16], the linear branch number is verified to be 5.

Over 16 rounds, there are four 4-round blocks. By virtue of Theorem 3, where $n = 2$ and $r = 4$, we have

$$p \leq (2^{-6})^{4\times 5+2} = 2^{-132} < 2^{-128}.$$

By Theorem 4, we obtain $q \leq (2^{-6})^{15} = 2^{-90}$. This implies that an attacker needs to collect at least $2^{90}$ chosen/known plaintexts to mount an attack, which

is not feasible in practice. Also, as remarked in [4], both Theorems 3 and 4 apply to 2-cell GF-NLFSR ciphers in general and so we expect p-Camellia to be more secure in practice, and this will be verified through computer stimulation with the results provided in an extended version of this paper.

This implies that there is no effective differential or linear characteristic for p-Camellia reduced to more than 15 rounds. In other words, p-Camellia offers sufficient security against differential and linear attack.

### 4.4 Other Attacks on p-Camellia

In this section, we briefly examine the protection of p-Camellia against various known attacks. Since p-Camellia uses the same components as Camellia, we expect that p-Camellia offers similar level of protection against most of the attacks, as compared to Camellia.

**Boomerang Attack.** To perform boomerang attack, the cipher is split into two shorter ciphers $E_0$ and $E_1$ such that the differential probability of each part is known to be large. Suppose an adversary split 16 rounds into $E_0$ and $E_1$ with $r$ and $16 - r$ rounds respectively. By Theorem 3, the characteristic differential probability of each sub-ciphers would be bounded by $p_0 \leq (2^{-30})^{\lfloor r/4 \rfloor}$ and $p_1 \leq (2^{-30})^{\lfloor (16-r)/4 \rfloor}$. (Note that we ignore the last term in the upper bound of Theorem 3 for ease of calculation.) It can be easily verified that $\lfloor r/4 \rfloor + \lfloor (16 - r)/4 \rfloor \geq 3$ for $r = 1, \ldots, 15$. Consequently,

$$p_0^2 \times p_1^2 \leq 2^{-60 \times 3} = 2^{-180} < 2^{-128},$$

and thus p-Camellia is secure against boomerang attack.

**Impossible Differential Attack.** Impossible differential attack is a chosen plaintext attack and is an extension of differential cryptanalysis. The main idea of this attack is to construct an impossible differential characteristic which is then used to filter wrong key guesses. According to [8], there is at least one 4-round impossible differential in the 2-cell GF-NLFSR, namely $(\alpha, 0) \nrightarrow_4 (\beta, \beta)$, where $\alpha$ and $\beta$ are non-zero fixed differences. We have not found impossible differentials with more than 4 rounds. As explained in [5], we expect that the presence of the $FL$- and $FL^{-1}$ functions will greatly increase the difficulty of performing impossible differential attack on p-Camellia since the functions change the differential paths depending on key values.

**Integral Attack.** In an integral attack, the attacker studies the propagation of multisets of chosen plaintexts of which part is held constant, and another part varies through all possibilities (also said to be *active*) through the cipher. There is a 4-round integral distinguisher of 2-cell GF-NLFSR [8], namely $(A, C) \rightarrow (S_0, S_1)$, where $C$ is constant, $A$ is active and $S_0 \oplus S_1$ is active. We have not found integral distinguishers with more than 4 rounds. An adversary can extend an integral attack distingusher by at most three rounds. That means he would need to extend the integral attack distinguisher from 4 to $18 - 3 = 15$ rounds which seems unlikely.

**Slide Attack.** The slide attack works on ciphers with cyclical structures over a few rounds. According to [5], the $FL$- and $FL^{-1}$- functions are inserted between every 6 rounds to provide non-regularity across rounds. In addition, different subkeys are used for every round, making slide attack unlikely.

We now proceed to examine the protection of p-Camellia against higher order differential attack and interpolation attack. We will adopt a similar approach as [5], which is somewhat heuristic but adequate for us to have a comprehensive and insightful discussion.

**Higher Order Differential Attack.** Higher order differential attack was introduced by Knudsen in [17]. This attack works especially well on block ciphers with components of low algebraic degree such as the KN-Cipher [13], whereby the ciphers can be represented as Boolean polynomials of low degree in terms of the plaintext. The attack requires $O(2^{t+1})$ chosen plaintext when the cipher has degree $t$.

p-Camellia uses exactly the same S-boxes as Camellia and it was confirmed in [5] that the degree of the Boolean polynomial of every output bit of the S-boxes is 7 by finding Boolean polynomial for every outpit bit of the S-boxes. Hence, similar to Camellia, the degree of an intermediate bit in the encryption process should increase as the data passes through many S-boxes. Indeed, let $(\alpha_i, \beta_i)$ be the input to the $(i+1)$-th round of p-Camellia. Suppose $\deg(\alpha_0) = \deg(\beta_0) = 1$. After the first round, $\deg(\alpha_1) = \deg(\beta_0) = 1$ while $\deg(\beta_1) = \deg(F(\alpha_0) \oplus \beta_0) = 7$. Continuing this process, we see that the degrees of $\alpha_i$ and $\beta_i$ for $i = 0, 1, 2, \cdots$, increases as follows: $(1,1), (1,7), (7,7), (7,49), (49,49), (49,127), (127,127), \cdots$

That is, the degrees increase exponentially as the number of rounds increase and reach the maximum degree of 127 after the 6th round, implying that it is highly unlikely that higher order differential attack will work.

**Interpolation Attack.** The interpolation attack [14] works on block ciphers that can be expressed as an equation in $GF(2^d)$ with few monomials. p-Camellia uses the same components as Camellia and it was shown in [5] that as the data passes through many S-boxes and the $P$-function, the cipher became a complex function which is a sum of many multi-variate monomials over $GF(2^8)$. Hence we also expect p-Camellia to be secure against interpolation attack.

### 4.5 Implementation Advantages

Before we discuss the implementation advantages of p-Camellia we briefly introduce hardware implementation strategies for block ciphers that consist of a round-function that is iterated several times. While software implementations have to process single operations in a serial manner, hardware implementations offer more flexibility for parallelization. Generally speaking there exist three major architecture strategies for the implementation of block ciphers: *serialized*, *round-based*, and *parallelized*. In a *serialized* architecture only a fraction of a single round is processed in one clock cycle. These lightweight implementations allow reduction in area and power consumption at the cost of a rather

long processing time. If a complete round is performed in one clock cycle, we have a *round-based* architecture. This implementation strategy usually offers the best time-area product and throughput per area ratio. A *parallelized* architecture processes more than one round per clock cycle, leading to a rather long critical path. A longer critical path leads to a lower maximum frequency but also requires the gates to drive a higher load (fanout), which results in larger gates with a higher power consumption. By inserting intermediate registers (a technique called *pipelining*), it is possible to split the critical path into fractions, thus increasing the maximum frequency. Once the pipeline is filled, a complete encryption can be performed in one clock cycle with such an architecture. Consequently, this implementation strategy yields the highest throughput at the cost of high area demands. Furthermore, since the pipeline has to be filled, each pipelining stage introduces a delay of one clock cycle.

From a lightweight perspective, *i.e.* if we consider serialized architectures, it is no wonder that area, power and timing demands stay the same for Camellia and p-Camellia, since no operation was introduced or removed. Also a round-based p-Camellia implementation is as efficient as a round-based Camellia implementation. However, if we consider applications that require high throughput, p-Camellia has significant advantages. If we consider an architecture that implements two rounds in one clock cycle (see Figure 2), Camellia's critical path involves two F-functions and two 2-input XOR gates, compared to only one F-function and one 3-input XOR gate for p-Camellia. Since Camellia inserts every six rounds the $FL$ and $FL^{-1}$ functions, it is advantageous to parallelize this fraction of Camellia/p-Camellia. In this case the critical path of Camellia consists of six F-functions, six 2-input XOR gates and the delay of $FL/FL^{-1}$ while

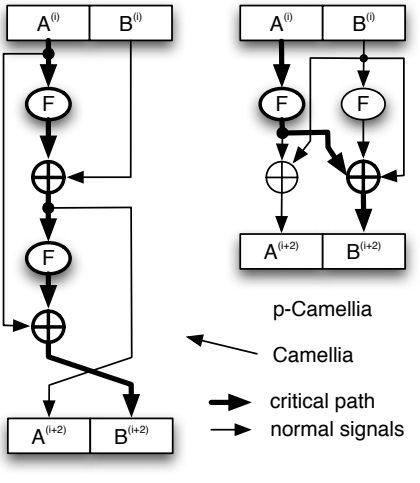



**Fig. 2.** Possible hardware architecture of two rounds of Camellia (left) and p-Camellia (right)

p-Camellia's critical path only consists of three F-functions, three 3-input XOR gates, and the delay of $FL/FL^{-1}$. Given the fact that the F-function consists of a 2-input XOR gate (key addition), several combinatorial gates (S-box) and an extensive XOR network (P-function), the delay difference between a 2-input and a 3-input XOR gate is negligible. Hence p-Camellia can achieve a maximum frequency that is twice as high as it would be for Camellia while having lower area and power demands. In case pipelining is applied, Camellia requires twice as much pipelining stages as p-Camellia to achieve the same maximum frequency, resulting in a delay that is twice as high.

## 5 Application 2: Parallelizing SMS4

### 5.1 Brief Description of SMS4

According to [11], SMS4 takes in a 128-bit key and uses a 32-round generalized Feistel structure to transform the plaintext to the ciphertext. Each round of the generalized Feistel transformation transforms four 32-bit words $X_i$, $i = 0, 1, 2, 3$, as follows:

$$(X_0, X_1, X_2, X_3, rk) \mapsto (X_1, X_2, X_3, X_0 \oplus T(X_1 \oplus X_2 \oplus X_3 \oplus rk)), \quad (3)$$

where $rk$ denotes the round key. In each round, the nonlinear function $T$ does the following operations in sequence: 32-bit subkey addition, S-box Subsitution (layer of four 8-bit S-boxes) and lastly, a 32-bit linear transformation $L$.

It is well-known that the S-boxes adopt functions affine equivalent to the inversion function in $GF(2^8)$ [15,10], which achieves the best known maximum differential and linear probabilities of $2^{-6}$. Furthermore, it can be verified that the branch number of the linear transformation $L$ is $\mathcal{L}_d = 5$. This gives optimal spreading effect which increases the number of active S-boxes for protection against differential and linear cryptanalysis.

The key schedule of SMS4 XORs the secret key $MK$ with a constant $FK$ and passes it through a nearly-identical 32-round structure as the main SMS4 cipher. The only difference is that the 32-bit linear transformation $L$ is replaced by a simpler linear transformation $L'$, which can be verified to have branch number $\mathcal{L}'_d = 4$. The 32-bit nonlinear output of the $i$-th round of the key schedule is taken to be the $i$-th round subkey of the main cipher. For more details, please refer to [11].

### 5.2 Parallelizing SMS4: p-SMS4

In this section, we propose another version of the existing SMS4 block cipher, which we call p-SMS4 (*"parallelizable" SMS4*). As described previously, SMS4 uses a generalized Feistel network structure described by equation (3). For the encryption procedure of p-SMS4, we shall replace the generalized Feistel network with the 4-cell GF-NLFSR block cipher structure described by:

$$(X_0, X_1, X_2, X_3, rk) \mapsto (X_1, X_2, X_3, X_1 \oplus X_2 \oplus X_3 \oplus T(X_0 \oplus rk)). \quad (4)$$

Other components such as number of rounds and the $T$-function, which consists of four S-boxes and a $L$-function, remain the same as SMS4. One round of p-SMS4 corresponds to a 4-cell version of the structure in Figure 1, where the nonlinear function $F(\cdot)$ is the $T$-function used in SMS4.

The key schedule of p-SMS4 XORs the secret key $MK$ with a constant $FK$ and passes it through an identical 32-round structure as the main cipher of p-SMS4 described by equation (4). The constant $FK$, S-box and the linear transformation $L'$ of the key schedule remain the same as SMS4. We need the key schedule to have the same structure as the main cipher so that it is also parallelizable in hardware, and thus can be made "on-the-fly".

### 5.3 Differential Cryptanalysis of p-SMS4

Su et al. proved bounds for the differential characteristic probability of the SMS4 cipher in [18]. One of the results they proved was that in every 7 rounds of the SMS4 cipher, there are at least 5 active S-boxes. However, there are currently no known bounds on the linear characteristic probability of SMS4 to the best of our knowledge.

Similarly for the p-SMS4 cipher, we can easily compute the differential characteristic bound by Theorem 3. Denote the maximum differential probability of p-SMS4 reduced to 29-round by $p$ (we assume a minus-3 round attack where the attacker guesses three subkeys with complexity $2^{96}$).

Recall that both p-SMS4 and SMS4 use the same $T$-function. In the case of p-SMS4, the maximum differential probability of the S-boxes is $2^{-6}$ and $\mathcal{L}_d = 5$. By virtue of Theorem 3 with $n = 4$ and $r = 5$, the first 24 rounds has $5 \times 3 + \lfloor 3/2 \rfloor = 16$ active S-boxes. Over the next 5 rounds, we have 2 active S-boxes by Proposition 1. Therefore the differential characteristic probability over 29 rounds satisfies:

$$p \leq (2^{-6})^{16} \times (2^{-6})^2 = 2^{-108}.$$

This implies that an attacker needs to collect at least $2^{108}$ chosen plaintext-ciphertext pairs to launch an attack. This is not feasible in practice. Moreover by Remark 1, for random input differences, we have at least 5 active S-boxes every 5 rounds with probability $1 - 2^{-32}$. Only $2^{-32}$ of the time do we need 8 rounds to ensure at least 5 active S-boxes. Thus we expect the bound for the differential characteristic probability to be even lower. In summary, we have shown that p-SMS4 offers sufficient security against differential cryptanalysis.

*Remark 2.* Like the SMS4 cipher, we currently do not have a bound for the characteristic linear probability of p-SMS4. This shall be studied in an extended version of this paper.

### 5.4 Related-Key Differential Attack on p-SMS4

Related-key differential attacks have been shown to have the devastating effect of recovering the secret key of AES-256 with a complexity of $2^{131}$ using $2^{35}$ related keys in [7]. In related-key differential attack, there are non-zero differential

inputs into both the cipher and the key schedule. The adversary tries to find a differential characteristic path in the key schedule with probability $p_k$ and a differential characteristic path in the main cipher with probability $p_{c|k}$ that holds, on the condition that the key schedule differential path is true. The attacker can then launch the attack with complexity $O(1/(p_k \times p_{c|k}))$ where he can tweak the secret key $1/p_k$ times to get that many related keys. In AES-256, we have $p_k = 2^{-35}$ and $p_{c|k} = 2^{-93}$.

Because the p-SMS4 key schedule uses a 4-cell GF-NLFSR structure, we can try to bound the probability $p_k$ of a differential characteristic path in the key schedule by Theorem 3. However, Theorem 3 cannot be directly applied to the main cipher to derive the differential characteristic probability $p_{c|k}$ because there are subkey differential input into every round.

We use the fact that the key schedule uses the inversion S-box with differential probability $2^{-6}$ and that the linear transform $L'$ has branch number $\mathcal{L'}_d = 4$. By Theorem 3 with $n = 4$ and $r = 4$, every 24 rounds of the key schedule has $4 \times 3 + \lfloor 3/2 \rfloor = 13$ active S-boxes. With a computation similar to Section 5.3, we have another 2 active S-boxes over the next 5 rounds giving:

$$p_k \leq (2^{-6})^{13} \times (2^{-6})^2 = 2^{-90}.$$

over 29 rounds of the key schedule. That means the complexity of any minus-3 round related-key differential attack is at least $O(2^{90})$ and uses at least $2^{90}$ related keys, which is not feasible in practice. Again, by a similar explanation as in Section 5.3 based on Remark 1, most of the time we have 5 active S-boxes per 5 rounds and we expect $p_k$ to be lower and the attack complexity to be higher.

In [6], a related-key boomerang attack on AES-256 with a complexity of $2^{119}$ using 4 related keys is presented but it assumes a more powerful adverserial model. In a similar way, we can show through the p-SMS4 key schedule differential structure that related-key boomerang attack is infeasible.

### 5.5 Other Attacks on p-SMS4

**Boomerang Attack.** Suppose an adversary performs a minus-3 round attack on 29 rounds of p-SMS4. He would need to split 29 rounds into two subciphers $E_0$, $E_1$ with $r$ and $29 - r$ rounds respectively, where $r = 1, \cdots, 28$. By Proposition 1 and Theorem 3, $p_0 \leq (2^{-6})^{5\times\lfloor \frac{r}{8} \rfloor + 2\times\lfloor \frac{r \bmod 8}{5} \rfloor}$ and $p_1 \leq (2^{-6})^{5\times\lfloor \frac{29-r}{8} \rfloor + 2\times\lfloor \frac{(29-r) \bmod 8}{5} \rfloor}$. (Note that we ignore the last term in the upper bound of Theorem 3 for ease of calculation.) For $r = 1, \cdots, 28$, let $n_8 = \lfloor \frac{r}{8} \rfloor + \lfloor \frac{29-r}{8} \rfloor$ and $n_5 = \lfloor \frac{r \bmod 8}{5} \rfloor + \lfloor \frac{(29-r) \bmod 8}{5} \rfloor$. It can be easily checked that there are only three combinations of values that $n_8$ and $n_5$ can take, as summarized in the Table 1.

Now $p_0 \times p_1 \leq (2^{-6})^{5n_8+2n_5}$. This implies that

$$p_0^2 \times p_1^2 \leq (2^{-12})^{5n_8+2n_5}.$$

The upper bounds of $p_0^2 \times p_1^2$ for each combination of $n_8$ and $n_5$ are also given in Table 1. From Table 1, we see that $p_0^2 \times p_1^2 < 2^{-128}$. Hence p-SMS4 is secure against boomerang attack.

**Table 1.** Values of $n_8$, $n_5$ and upper bounds of $p_0^2 \times p_1^2$ for $r = 1, \cdots, 28$

| $n_8$ | $n_5$ | $r$ | $p_0^2 \times p_1^2$ |
|---|---|---|---|
| 3 | 0 | $1, \cdots, 4, 9, \cdots, 12, 17, \cdots, 20, 25, \cdots, 28$ | $\leq (2^{-12})^{15} = 2^{-180}$ |
| 3 | 1 | 5, 8, 13, 16, 21, 24 | $\leq (2^{-12})^{15+2} = 2^{-204}$ |
| 2 | 2 | 6, 7, 14, 15, 22, 23 | $\leq (2^{-12})^{10+4} = 2^{-168}$ |

**Impossible Differential Attack.** According to [8,19,22], there is at least one 18-round impossible differential distinguisher in the 4-cell GF-NLFSR, which results in a 25-round impossible differential attack with complexity $2^{123}$ and uses $2^{115}$ chosen plaintext encryptions. An identical attack is applicable to 25-round p-SMS4 with the same complexity. However, that attack is unlikely to work on the full p-SMS4 cipher, which has 32 rounds.

**Integral Attack.** According to [8,19], there is at least one 16-round integral attack distinguisher in the 4-cell GF-NLFSR starting with one active 32-bit word. A naive key guessing attack can extend this distinguisher by at most 3 rounds at the end (guessing more rounds of keys may make the complexity too close to $2^{128}$). An adversary may extend the attack by 4 rounds in front, starting with 3 active words and using the method of [12]. Using these means, we expect a $4 + 16 + 3 = 23$ round attack on p-SMS4 and the full 32 rounds will be secure against integral attack.

**Slide Attack.** The slide attack works on ciphers with cyclical structures over a few rounds. However the subkeys used in every round are nonlinearly derived from the previous subkey. Thus the subkeys are all distinct and there is no simple linear relation between them, making slide attack unlikely.

**XSL Attack.** In [15], Ji and Hu showed that the eprint XSL attack on SMS4 embedded in $GF(2^8)$ can be applied with complexity $2^{77}$. A similar analysis can be applied on p-SMS4 to show that the complexity of the eprint XSL attack on p-SMS4 embedded in $GF(2^8)$ is also $2^{77}$. However, it was shown in [10] by Choy et al. that Ji and Hu's analysis might be too optimistic and the actual complexity of the compact XSL attack on embedded SMS4 is at least $2^{216.58}$. We can use an analysis identical to the ones used in [10] to show that the complexity of the compact XSL attack on p-SMS4 is also at least $2^{216.58}$.

Using a similar approach as [5], we discuss the protection of p-SMS4 against higher order differential attack and interpolation attack in the remaining of this section.

**Higher Order Differential Attack.** As mentioned previously, higher order differential attack is generally applicable to ciphers that can be represented as Boolean polynomials of low degree in terms of the plaintext. The attack requires $O(2^{t+1})$ chosen plaintext when the cipher has degree $t$.

p-SMS4 uses exactly the same S-boxes as SMS4 where the degree of the Boolean polynomial of every output bit of the S-boxes is 7. Making the assumption that when we compose two *randomly chosen* S-boxes $F, G$ of degree $t_1, t_2$, $F \circ G$ should have degree $t_1 t_2$. We expect the degree of an intermediate bit in the encryption process to increase exponentially as the data passes through many S-boxes.

Indeed, by the $4^{th}$ round, every output bit will have degree 7. By the $8^{th}$ round, every output bit will have degree $7^2 = 49$. By the $12^{th}$ round, every output bit will have degree $min(7^3, 127) = 127$ in terms of the plaintext bits. Therefore p-SMS4 is secure against higher order differential attack.

**Interpolation Attack.** The interpolation attack works on block ciphers that can be expressed as an equation in $GF(2^d)$ with few monomials. p-SMS4 uses the same components as SMS4 and as the data passes through many S-boxes and L-functions, the cipher will became a complex function which is a sum of exponentially many multi-variate monomials over $GF(2^8)$. Hence we expect p-SMS4 to be secure against interpolation attack.

### 5.6 Implementation Advantages

Similar to p-Camellia we will assess the implementation advantages of p-SMS4 over SMS4 with respect to serialized, round-based and parallelized architectures. In case of SMS4 the XOR sum of three branches forms the input to the F-function and its output is XORed to the last branch while p-SMS4 uses one branch as the input for the F-function and XORs its output to the remaining three branches. This difference allows more flexible implementations of p-SMS4 compared to SMS4, because the XOR sum of four signals can be achieved by either using three 2-input XOR gates or combining a 3-input XOR gate with a 2-input XOR gate. The first option is faster (0.33 ns vs. 0.45 ns) while the second option requires less area (256 GE vs. 235 GE), which is an advantage for lightweight implementations. Beside this flexibility, p-SMS4 has similar characteristics as SMS4 for a serialized implementation. The critical path of a round-based p-SMS4 implementation is shorter than that of SMS4, since it consists of the F-function and a 2-input XOR gate compared to a 3-input XOR gate, the F-function and a 2-input XOR gate for SMS4.

For parallelized implementations p-SMS4 offers even greater advantages. If we consider an implementation that processes four rounds in one clock cycle (see figure 3), the critical path of p-SMS consists only of the F-function and two 2-input XOR gates while SMS4's critical path consists of four F-functions, four 2-input XOR gates and four 3-input XOR gates. Hence, the maximum frequency and thus the maximum throughput that can be achieved with p-SMS4 using such an architecture is more than four times higher while the area and power consumption are lower compared to a corresponding SMS4 implementation. A similar frequency can be achieved for SMS4 by inserting three pipelining stages, which significantly increases the area and power consumption and introduces a delay of three clock cycles.

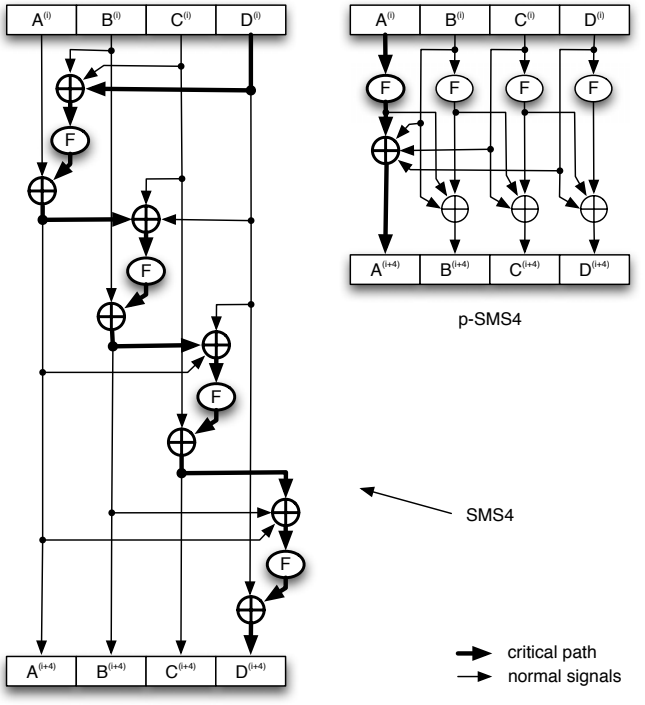



**Fig. 3.** Possible hardware architecture of four rounds of SMS4 (left) and p-SMS4 (right)

From these estimates it becomes clear that the implementation advantages of our newly proposed parallelizable Feistel-structure becomes even larger with a growing number of branches. In fact, an $n$-cell GF-NLFSR can be implemented using $n$ rounds in parallel while having the same critical path as for a single round implementation. This translates to an $n$ times higher maximum frequency while the area and power consumption are less then for a conventional Feistel structure.

## 6 Conclusion

In this paper we proposed the use of $n$-cell GF-NLFSR structure to parallelize (Generalized) Feistel structures. We used two examples, p-Camellia and p-SMS4, and showed that they offer sufficient security against various known existing attacks. At the same time, as compared to their conventional Feistel structure counterparts Camellia and SMS4, their hardware implementations achieve a maximum frequency that is $n$ times higher, where $n$ is the number of Feistel branches, while having lower area and power demands. These estimates indicate that of $n$-cell GF-NLFSRs are particularly well suited for applications that require a high throughput.

## References


1. National Bureau of Standards, Data Encryption Standard, FIPS-Pub.46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C. (January 1977)
2. SKIPJACK and KEA Algorithm Specifications, http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf
3. Universal Mobile Telecommunications System (UMTS); Specification of the 3GPP confidentiality and integrity algorithms; Document 2: Kasumi specification, http://www.etsi.org/website/document/algorithms/ts_135202v070000p.pdf
4. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Specification of Camellia - A 128-Bit Block Cipher (2000), http://info.isl.ntt.co.jp/camellia/
5. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: *Camellia*: A 128-Bit Block Cipher Suitable for Multiple Platforms, Design and Analysis. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 39–56. Springer, Heidelberg (2001)
6. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256, IACR eprint server, 2009/317 (June 2009), http://eprint.iacr.org/2009/317
7. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and Related-Key Attack on the Full AES-256 (Extended Version), IACR eprint server, 2009/241 (June 2009), http://eprint.iacr.org/2009/241
8. Choy, J., Chew, G., Khoo, K., Yap, H.: Cryptographic Properties and Application of a Generalized Unbalanced Feistel Network Structure (Revised Version), Cryptology Eprint Archive, Report 2009/178 (July 2009) (Revision of [9])
9. Choy, J., Chew, G., Khoo, K., Yap, H.: Cryptographic Properties and Application of a Generalized Unbalanced Feistel Network Structure. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 73–89. Springer, Heidelberg (2009)
10. Choy, J., Yap, H., Khoo, K.: An Analysis of the Compact XSL Attack on BES and Embedded SMS4. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 103–118. Springer, Heidelberg (2009)
11. Diffe, W., Ledin, G.: SMS4 Encryption Algorithm for Wireless Networks, Cryptology ePrint Archive: Report 2008/329 (2008)
12. Hwang, K., Lee, W., Lee, S., Lee, S., Lim, J.: Saturation Attacks on Reduced Round Skipjack. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 100–111. Springer, Heidelberg (2002)
13. Jakobsen, T., Knudsen, L.R.: The Interpolation Attack on Block Ciphers. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 28–40. Springer, Heidelberg (1997)
14. Jakobsen, T., Knudsen, L.R.: Attacks on Block Ciphers of Low Algebraic Degree. Journal of Cryptology 14, 197–210 (2001)
15. Ji, W., Hu, L.: New Description of SMS4 by an Embedding over $GF(2^8)$. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 238–251. Springer, Heidelberg (2007)
16. Kanda, M.: Practical Security Evaluation against Differential and Linear Cryptanalysis for Feistel Ciphers with SPN Round Function. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 324–338. Springer, Heidelberg (2001)
17. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
18. Su, B., Wu, W., Zhang, W.: Differential Cryptanalysis of SMS4 Block Cipher, Cryptology Eprint Archive, Report 2010/062 (February 2010)

19. Li, R., Sun, B., Li, C.: Distinguishing Attack on a Kind of Generalized Unbalanced Feistel Network, Cryptology Eprint Archive, Report 2009/360 (July 2009)
20. Park, S., Sung, S., Lee, S., Lim, J.: Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 247–260. Springer, Heidelberg (2003)
21. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit Blockcipher CLEFIA (Extended Abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
22. Wu, W., Zhang, L., Zhang, L., Zhang, W.: Security Analysis of the GF-NLFSR Structure and Four-Cell Block Cipher, Cryptology Eprint Archive, Report 2009/346 (July 2009)
23. Rijmen, V., Daemon, J., Preneel, B., Bosselaers, A., Win, E.D.: The cipher SHARK. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 99–111. Springer, Heidelberg (1996)

## A Figure of p-Camellia

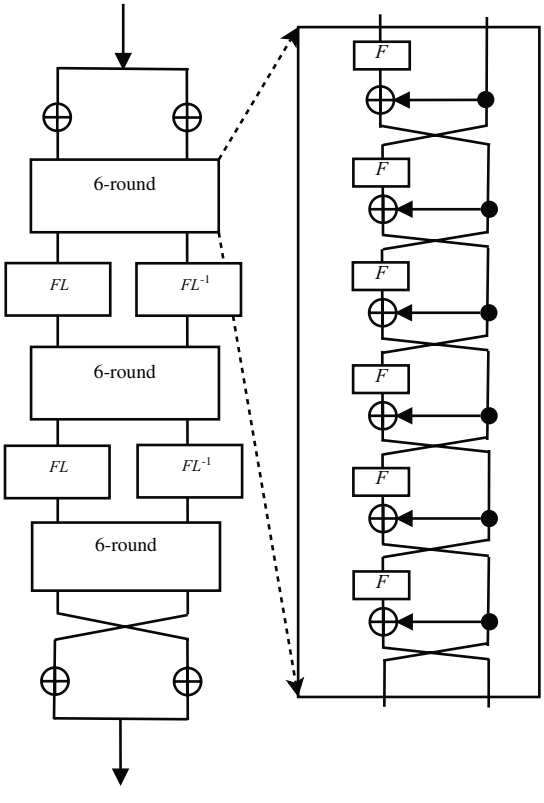



**Fig. 4.** p-Camellia block cipher structure

# Improved Linear Differential Attacks on CubeHash

Shahram Khazaei[1], Simon Knellwolf[2], Willi Meier[2], and Deian Stefan[3]

[1] EPFL, Switzerland
[2] FHNW, Switzerland
[3] The Cooper Union, USA

**Abstract.** This paper presents improved collision attacks on round-reduced variants of the hash function CubeHash, one of the SHA-3 second round candidates. We apply two methods for finding linear differential trails that lead to lower estimated attack complexities when used within the framework introduced by Brier, Khazaei, Meier and Peyrin at ASIACRYPT 2009. The first method yields trails that are relatively dense at the beginning and sparse towards the end. In combination with the condition function concept, such trails lead to much faster collision attacks. We demonstrate this by providing a real collision for CubeHash-5/96. The second method randomizes the search for highly probable linear differential trails and leads to significantly better attacks for up to eight rounds.

**Keywords:** hash function, differential attack, collision, linearization, SHA-3, CubeHash.

## 1 Introduction

Hash functions are important cryptographic primitives that transform arbitrary-length input messages into fixed-length message digests. They are used in many applications, notably in commitment schemes, digital signatures and message authentication codes. To this end they are required to satisfy different security properties, one of them is collision resistance. Informally, a hash function is collision resistant if it is practically infeasible to find two distinct messages $m_1$ and $m_2$ that produce the same message digest.

Chabaud and Joux [7] presented the first differential collision attack on SHA-0. Using a linearized model of the hash function, they found message differences that lead to a collision of the original hash function with a higher probability than the birthday bound. Similar strategies were later used by Rijmen and Oswald [12] on SHA-1 and by Indesteege and Preneel [8] on EnRUPT.

Pramstaller *et al.* [11] related the problem of finding highly probable linear differences to the problem of finding low weight codewords of a linear code. A recent work of Brier *et al.* [4] more precisely analyzed this relation for hash functions whose non-linear operations only consist in modular additions. They reformulate the problem of finding message pairs that conform to a linear differential trail to that of finding preimages of zero of a condition function. The

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 407–418, 2010.

search for such preimages is accelerated by the implicit use of message modification techniques. Given a linear differential trail, the concept further allows to estimate the corresponding complexity of the collision attack.

The success of the attack essentially depends on finding appropriate linear trails that lead to a low complexity of the attack. Such trails must not only have low weight, but the distribution of the weights along the trail must also be considered. A trail that is dense at the beginning and sparse towards the end yields a lower attack complexity than an arbitrary trail of comparable weight. This is due to freedom degrees use: initial conditions can be more easily satisfied than those towards the end of the trail.

**Contribution of this Paper.** We apply two different methods for finding appropriate trails for variants of the SHA-3 second round candidate CubeHash. For several round parameters $r$ and message block sizes $b$ we present better collision attacks on CubeHash-$r/b$ than those presented so far. Specifically, we find collisions of CubeHash-5/96 and give a theoretical attack of CubeHash-8/96 with estimated complexity of $2^{80}$ compression function calls. This improves over the generic attack with complexity of about $2^{128}$ and is the first collision attack on more than seven rounds.

**Previous Results on CubeHash.** We refer to part 2. B. 5 of [2] for a complete survey of cryptanalytic results on CubeHash. The currently best collision attacks on CubeHash-$r/b$ for message block sizes $b = 32, 64$ were presented in [4]. For $b = 32$ they present attacks of complexity $2^{54.1}$ and $2^{182.1}$ for four and six rounds, respectively. For $b = 64$ an attack of complexity $2^{203}$ for seven rounds is given. No collision attack for more than seven rounds was presented so far. Generic attacks are discussed by Bernstein in the appendix of [1].

**Organization.** Section 2 reviews the linearization framework and the concept of condition function presented in [4]. In Section 3 we describe the hash function CubeHash and define an appropriate compression function. Section 4 details how to find linear differential trails that, in combination with the concept of condition function, lead to successful collision attacks. In Section 5 we analyze CubeHash-5/96 and present a real collision. We conclude in Section 6.

## 2 Linearization Framework and Condition Function

In this section we fix notations and briefly review the general framework for collision attacks presented in [4] (see [5] for an extended version).

### 2.1 Fixed-Input-Length Compression Function

To any hash function we can attribute a fixed-input-length compression function $\mathsf{Compress} : \{0,1\}^m \rightarrow \{0,1\}^h$, with $m \geq h$, such that a collision for the compression function directly translates to a collision of the hash function[1] (e.g., we

[1] Note that this notion of compression function does not coincide with the frequently used compression function in the context of Merkle-Damgård and other iterated constructions.

can just restrict the domain of the hash function). We suppose that the only non-linear operations of the compression function consist of modular additions of $w$-bit words. For any input $M$ to the compression function denote $\mathbf{A}(M)$ and $\mathbf{B}(M)$ the concatenation of all left, and, respectively right addends that are added in the course of the computation of $\mathsf{Compress}(M)$. Analogously define $\mathbf{C}(M)$ as the concatenation of the corresponding carry words. Thus, if $n_a$ is the number of additions effected in the course of one evaluation of the compression function, each of $\mathbf{A}(M), \mathbf{B}(M)$ and $\mathbf{C}(M)$ contains $n_a w$ bits.

### 2.2 Linearization and Raw Probability

Let $\mathsf{Compress}_{\mathsf{lin}}$ be the linear function obtained by replacing all modular additions of $\mathsf{Compress}$ by XORs. For an input $\Delta$ to this linearized compression function denote $\boldsymbol{\alpha}(\Delta)$ and $\boldsymbol{\beta}(\Delta)$ the concatenation of the left, and, respectively right addends that are XORed in the course of the computation of $\mathsf{Compress}_{\mathsf{lin}}(\Delta)$, setting their most significant bits to zero[2].

We say that a message $M$ *conforms to the trail of* $\Delta$ if for all $i = 0, \ldots, n_a - 1$

$$((A^i \oplus \alpha^i) + (B^i \oplus \beta^i)) \oplus (A^i + B^i) = \alpha^i \oplus \beta^i,$$

where $A^i, B^i, \alpha^i$ and $\beta^i$ respectively denote the $i$th $w$-bit word of $\mathbf{A}(M)$, $\mathbf{B}(M)$, $\boldsymbol{\alpha}(\Delta)$ and $\boldsymbol{\beta}(\Delta)$. According to Lemma 2 in [4], the probability that a randomly chosen $M$ conforms to the trail of $\Delta$ is given by

$$p_\Delta = 2^{-\mathrm{wt}(\boldsymbol{\alpha}(\Delta) \vee \boldsymbol{\beta}(\Delta))},$$

where $\mathrm{wt}(\cdot)$ denotes the Hamming weight. If $\Delta$ lies in the kernel of the linearized compression function, $p_\Delta$ is a lower bound for the probability that the message pair $(M, M \oplus \Delta)$ is a collision of the compression function. We call $p_\Delta$ the *raw probability* of $\Delta$ and $y = -\log_2(p_\Delta)$ the number of conditions imposed by $\Delta$.

### 2.3 The Condition Function

Let $\Delta$ be in the kernel of the linearized compression function. The condition function $\mathsf{Condition}_\Delta$ has the same domain as the compression function, but outputs $Y$ of lengths $y = \mathrm{wt}(\boldsymbol{\alpha}(\Delta) \vee \boldsymbol{\beta}(\Delta))$. To shorten the notation we omit the argument $M$ to $\mathbf{A}, \mathbf{B}, \mathbf{C}$, and $\Delta$ to $\boldsymbol{\alpha}, \boldsymbol{\beta}$. Additionally, we use subscripts to denote bit positions, e.g., $\mathbf{A}_i$ is the $i$th bit of $\mathbf{A}$. Let $i_0, \ldots, i_{y-1}$ be the bit positions of the $y$ non-zero bits in $\boldsymbol{\alpha} \vee \boldsymbol{\beta}$. Define the condition function $Y = \mathsf{Condition}_\Delta(M)$ by

$$Y_j = (\boldsymbol{\alpha}_{i_j} \oplus \boldsymbol{\beta}_{i_j})\mathbf{C}_{i_j} \oplus \boldsymbol{\alpha}_{i_j}\mathbf{B}_{i_j} \oplus \boldsymbol{\beta}_{i_j}\mathbf{A}_{i_j} \oplus \boldsymbol{\alpha}_{i_j}\boldsymbol{\beta}_{i_j} \text{ for } j = 0, \ldots, y-1.$$

By Proposition 1 in [4], the problem of finding a message $M$ that conforms to the trail of $\Delta$ is equivalent to the problem of finding $M$ such that $\mathsf{Condition}_\Delta(M) = 0$.

Suppose an ideal situation where we are given partitions $\bigcup_{i=1}^{\ell} \mathcal{M}_i = \{0, \ldots, m-1\}$ and $\bigcup_{i=0}^{\ell} \mathcal{Y}_i = \{0, \ldots, y-1\}$ such that for $j = 0, \ldots, \ell$ the output bits

[2] The most significant bits of each addition are linear.

with position indices in $\mathcal{Y}_j$ only depend on the input bits with position indices in $\bigcup_{i=1}^{j} \mathcal{M}_i$. Then we expect to find an $M$ such that $\mathsf{Condition}_\Delta(M) = 0$ after

$$c_\Delta = \sum_{i=0}^{\ell} 2^{q_i}$$

evaluations of the condition function, where $q_i = |\mathcal{Y}_i| + \max(0, q_{i+1} - |\mathcal{M}_{i+1}|)$ for $i = \ell - 1, \ldots, 0$ and $q_\ell = |\mathcal{Y}_\ell|$. We call $c_\Delta$ the *theoretical complexity* of $\Delta$.

We refer to [5] for a method to approximately determine such partitions and suitable probabilities $2^{-p_i}$ to keep track of the non-ideality of these partitions. Theoretical complexities in this paper are computed including the probabilities $2^{-p_i}$. We further refer to [5] for instructions on implementing the search algorithm with negligible memory requirement.

## 3 Description of CubeHash

CubeHash [2] is a second-round candidate of the SHA-3 competition [10] of the National Institute of Standards and Technology. The function is designed with parameters $r, b$, and $h$ which are the number of rounds, the number of bytes per message block, and the hash output length (in bits), respectively. We denote the parametrized function as CubeHash-$r/b$. The initial proposal of CubeHash-8/1 was tweaked to CubeHash-16/32 which is about 16 times faster and is now the official proposal for all digest lengths $h = 224, 256, 384$ or 512. Third-party cryptanalysis with larger values of $b$ and fewer number of rounds $r$ is explicitly encouraged.

### 3.1 Algorithm Specification

CubeHash operates on 32-bit words. It maintains a 1024-bit internal state $X$ which is composed of 32 words $X_0, \ldots, X_{31}$. The algorithm is composed of five steps:

1. Initialize the state $X$ to a specified value that depends on $(r, b, h)$.
2. Pad the message to a sequence of $b$-byte input blocks.
3. For every $b$-byte input block:
   - XOR the block into the first $b$-bytes of the state.
   - Transform the state through $r$ identical rounds.
4. Finalize the state: XOR 1 into $X_{31}$ and transform the state through $10r$ identical rounds.
5. Output the first $h$ bits of the state.

A round consists of the following steps:

- Add $X_i$ into $X_{i \oplus 16}$, for $0 \leq i \leq 15$.
- Rotate $X_i$ to the left by seven bits, for $0 \leq i \leq 15$.

– Swap $X_i$ and $X_{i\oplus 8}$, for $0 \le i \le 7$.
– XOR $X_{i\oplus 16}$ into $X_i$, for $0 \le i \le 15$.
– Swap $X_i$ and $X_{i\oplus 2}$, for $i \in \{16, 17, 20, 21, 24, 25, 28, 29\}$.
– Add $X_i$ into $X_{i\oplus 16}$, for $0 \le i \le 15$.
– Rotate $X_i$ to the left by eleven bits, for $0 \le i \le 15$.
– Swap $X_i$ and $X_{i\oplus 4}$, for $i \in \{0, 1, 2, 3, 8, 9, 10, 11\}$.
– XOR $X_{i\oplus 16}$ into $X_i$, for $0 \le i \le 15$.
– Swap $X_i$ and $X_{i\oplus 1}$, for $i \in \{16, 18, 20, 22, 24, 26, 28, 30\}$.

In this paper we consider the variants CubeHash-$r/b$ with $b = 32, 64$ and $96$, always assuming $h = 512$.

### 3.2 Defining the Compression Function

Following [4] we define a fixed-input-length compression function Compress for CubeHash. This compression function is parametrized by a 1024-bit initial value $V$ and compresses $t$ ($t \ge 1$) $b$-byte message blocks $M = M^0 \| \cdots \| M^{t-1}$. The output $H = \mathsf{Compress}(M, V)$ consists of the last $1024 - 8b$ bits of the internal state after $tr$ round transformations processing $M$.

A colliding message pair $(M, M \oplus \Delta)$ for Compress directly extends to a collision of CubeHash by appending a pair of message blocks $(M^t, M^t \oplus \Delta^t)$ such that $\Delta^t$ erases the difference in the first $8b$ bits of the internal state. The difference $\Delta^t$ is called *erasing block difference*.

When searching for collisions of Compress, the parameter $V$ is not restricted to be the initial value of CubeHash. Specifically, $V$ can be the state after processing some message prefix $M^{\mathrm{pre}}$. Thus, a pair of colliding messages for the hash function then has the general form

$$(M^{\mathrm{pre}} \| M \| M^t \| M^{\mathrm{suff}}, M^{\mathrm{pre}} \| M \oplus \Delta \| M^t \oplus \Delta^t \| M^{\mathrm{suff}})$$

for an arbitrary message suffix $M^{\mathrm{suff}}$.

## 4 Constructing Linear Differentials

We linearize the compression function of CubeHash to find message differences that can be used for a collision attack as described in Section 2. Specifically, we are interested in finding differences with low theoretical complexity. As a first approach one can search for differences with a high raw probability.

### 4.1 Searching for High Raw Probability

Let $\mathsf{Compress}_{\mathsf{lin}}$ be the linearization of Compress obtained by replacing all modular additions in the round transformation with XORs and setting $V = 0$. Using the canonical bases, $\mathsf{Compress}_{\mathsf{lin}}$ can be written as a matrix $\mathcal{H}$ of dimension $(1024 - 8b) \times 8bt$. Let $\tau$ be the dimension of its kernel. As noted in [4], the matrix $\mathcal{H}$ does not have full rank for many parameters $r/b$ and $t$, and one can find

differences with high a raw probability (imposing a small number of conditions) in the set of linear combinations of at most $\lambda$ kernel basis vectors, where $\lambda \geq 1$ is chosen such that the set can be searched exhaustively. The results heavily depend on the choice of the kernel basis. Table 1 compares the minimal number of conditions for $\lambda = 3$ for two different choices of the kernel basis. The results in the first three rows are obtained using the same algorithm as in [4] to determine the bases. The results in the last three rows are obtained using the more standard procedure implemented for example in the Number Theory Library of Shoup [13].

**Table 1.** Minimal number of conditions found for $\lambda = 3$ using two different algorithms to determine the kernel bases

| $b/r$ | 4 | 5 | 6 | 7 | 8 | 16 |
|---|---|---|---|---|---|---|
| 32 | 156 | 1244 | 400 | 1748 | 830 | 2150 |
| 64 | 130 | 205 | 351 | 447 | 637 | 1728 |
| 96 | 62 | 127 | 142 | 251 | 266 | 878 |
| 32 | 189 | 1952 | 700 | 2428 | 830 | 2150 |
| 64 | 189 | 1514 | 700 | 1864 | 637 | 1728 |
| 96 | 67 | 128 | 165 | 652 | 329 | 928 |

The inverse raw probability is an upper bound of the theoretical complexity, and as such, we expect that differences with a high raw probability have a low theoretic complexity. However, a higher raw probability does not always imply a lower theoretic complexity. There are differences with lower raw probability that lead to a lower theoretic complexity than that of a difference with a higher raw probability. Hence, when searching for minimal complexity of the collision attack, simply considering the number of conditions imposed by a difference is not sufficient.

### 4.2 Searching for Sparsity at the End

As previously observed in [3,7,9,11,14], conditions in early steps of the computation can be more easily satisfied than those in later steps. This is due to message modifications, (probabilistic) neutral bits, submarine modifications and other freedom degrees use. Similar techniques are used implicitly when using a dependency table to find a preimage of the condition function (and thus a collision for the compression function). This motivates the search for differences $\Delta$ such that $\boldsymbol{\alpha}(\Delta) \vee \boldsymbol{\beta}(\Delta)$ is sparse at the end. In general, however, this is not the case for trails found using the above method and, in contrast, most are sparse in the beginning and dense at the end. This is due to diffusion of the linearized compression function.

We note that the linearized round transformation of CubeHash is invertible and let $\mathsf{Compress}^{\mathsf{b}}_{\mathsf{lin}}$ be defined in the same way as $\mathsf{Compress}_{\mathsf{lin}}$ but with inverse linearized round transformations. Suppose that $\Delta' = \Delta^0 \| \cdots \| \Delta^{t-1}$ lies in the

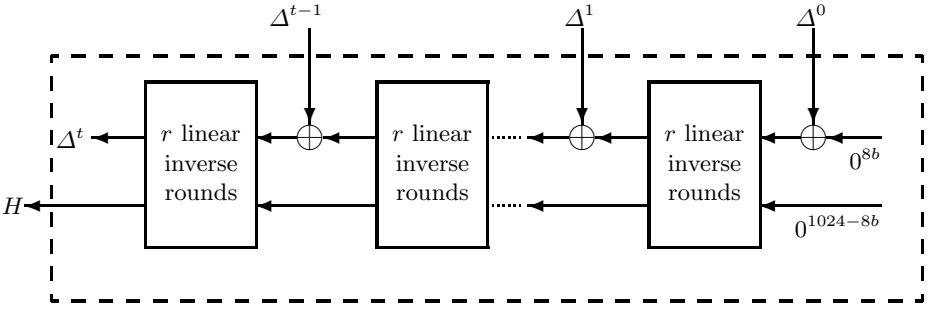



**Fig. 1.** Computation of $\mathsf{Compress}^{\mathsf{b}}_{\mathsf{lin}}$ on input $\Delta' = \Delta^0 \| \cdots \| \Delta^{t-1}$. If $\Delta'$ lies in the kernel, $H = 0$ and $\Delta = \Delta^t \| \cdots \| \Delta^1$ lies in the kernel of $\mathsf{Compress}_{\mathsf{lin}}$.

kernel of $\mathsf{Compress}^{\mathsf{b}}_{\mathsf{lin}}$ and $\Delta^t$ equals the (discarded) first $8b$ bits of the state after the $tr$ linear inverse round transformations processing $\Delta'$ as shown in Fig. 1. Then, the difference $\Delta = \Delta^t \| \cdots \| \Delta^1$ lies in the kernel of $\mathsf{Compress}_{\mathsf{lin}}$ and $\Delta^0$ is the corresponding erasing block difference. As for the linearized compression function we determine a basis of the kernel of $\mathsf{Compress}^{\mathsf{b}}_{\mathsf{lin}}$ and exhaustively search for linear combinations of at most $\lambda$ kernel basis vectors of high raw probability. Due to the diffusion of the inverse transformation, these trails tend to be dense at the beginning and sparse at the end.

### 4.3 Randomizing the Search

The kernel of $\mathcal{H}$ contains $2^\tau$ different elements. The above method finds the best difference out of a subset of $\sum_{i=1}^{\lambda} \binom{\tau}{i}$ elements. We may find better results by increasing $\lambda$ or by repeating the search for another choice of the basis. Using ideas from [11] we propose an alternative search algorithm, that works well for many variants of CubeHash and does not decisively depend on the choice of the kernel basis.

Let $\Delta_0, \ldots, \Delta_{\tau-1}$ be a kernel basis of $\mathsf{Compress}_{\mathsf{lin}}$ and denote $\mathcal{G}$ the matrix whose $\tau$ rows consist of the binary vectors $\Delta_i \| \boldsymbol{\alpha}(\Delta_i) \| \boldsymbol{\beta}(\Delta_i)$ for $i = 0, \ldots, \tau - 1$. Elementary row operations on $\mathcal{G}$ preserve this structure, that is, the rows always have the form $\Delta \| \boldsymbol{\alpha}(\Delta) \| \boldsymbol{\beta}(\Delta)$ where $\Delta$ lies in the kernel of $\mathsf{Compress}_{\mathsf{lin}}$ and its raw probability is given by the Hamming weight of $\boldsymbol{\alpha}(\Delta) \vee \boldsymbol{\beta}(\Delta)$. For convenience, we call this the raw probability of the row (instead of the raw probability of the first $|\Delta|$ bits of the row). Determine $i_{\max}$, the index of the row with the highest raw probability. Then iterate the following steps:

1. Randomly choose a column index $j$ and let $i$ be the smallest row index such that $\mathcal{G}_{i,j} = 1$ (choose a new $j$ if no such $i$ exists).
2. For all row indices $k = i + 1, \ldots, \tau - 1$ such that $\mathcal{G}_{k,j} = 1$:
   - add row $i$ to row $k$,
   - set $i_{\max} = k$ if row $k$ has higher raw probability than row $i_{\max}$.
3. Move row $i$ to the bottom of $\mathcal{G}$, shifting up rows $i + 1, \ldots, \tau - 1$ by one.

**Table 2.** Minimal number of conditions found with the randomized search. Values in boldface improve over the values in Table 1.

| $b/r$ | 4 | 5 | 6 | 7 | 8 | 16 |
|---|---|---|---|---|---|---|
| 32 | 156 | 1244 | **394** | 1748 | 830 | 2150 |
| 64 | 130 | 205 | **309** | 447 | 637 | 1728 |
| 96 | **38** | 127 | **90** | 251 | **151** | **709** |

*Remark.* A more specific choice of the column index $j$ in the first step does not lead to better results. In particular, we tried to prioritize choosing columns towards the end, or for every chosen column in $\boldsymbol{\alpha}(\Delta)$ to also eliminate the corresponding column in $\boldsymbol{\beta}(\Delta)$.

Table 2 shows the best found raw probabilities after 200 trials of 600 iterations. Estimating the corresponding theoretic complexities as described in Section 2.3 yields the improved collision attacks presented in Table 3.

**Table 3.** Logarithmic theoretical complexities of improved collision attacks

| $b/r$ | 4 | 5 | 6 | 7 | 8 | 16 |
|---|---|---|---|---|---|---|
| 32 | | | 180 | | | |
| 64 | | | 132 | | | |
| 96 | 7 | | 51 | | 80 | |

For CubeHash-$r/b$ there is a generic collision attack with complexity of about $2^{512-4b}$. For $b > 64$ this is faster than the generic birthday attack on hash functions with output length $h = 512$. For $b = 96$, specifically, the generic attack has a complexity of about $2^{128}$. Our attacks clearly improve over this bound.

## 5 Collision for CubeHash-5/96

This section illustrates the findings of Section 4.2 and provides a collision for CubeHash-5/96.

### 5.1 Linear Differentials

We consider two linear differences found by the methods of Section 4.1 and 4.2 respectively. Both consist of two 96-byte blocks, a first block that lies in the kernel of the linearized compression function and a second one that is the corresponding erasing block difference. They are given by

$$
\begin{aligned}
\Delta_a^0 = {} & \texttt{40000000 00000000 40000000 00000000 00000000 00000000}\\
& \texttt{00000000 00000000 00200000 00000000 00000000 00000000}\\
& \texttt{00000000 00000000 00000000 00000000 00000000 00000040}\\
& \texttt{00000000 00000040 00000000 00020000 00000000 00000000},\\
\Delta_a^1 = {} & \texttt{01000111 01000111 00000000 00000000 8008002A 00000000}\\
& \texttt{08000022 00000000 00000000 00000000 00000000 00000000}\\
& \texttt{00000000 00000000 11040000 00000000 40000101 01000111}\\
& \texttt{00000000 00000000 00002208 00000000 08002000 00000000}
\end{aligned}
$$

and

$$
\begin{aligned}
\Delta_b^0 = {} & \texttt{08000208 08000208 00000000 00000000 40000100 00000000}\\
& \texttt{00400110 00000000 00000000 00000000 00000000 00000000}\\
& \texttt{00000000 00000000 0800A000 00000000 08000888 08000208}\\
& \texttt{00000000 00000000 40011000 00000000 00451040 00000000},\\
\Delta_b^1 = {} & \texttt{80000000 00000000 80000000 00000000 00000000 00000000}\\
& \texttt{00000000 00000000 00400000 00000000 00000000 00000000}\\
& \texttt{00000000 00000000 00000000 00000000 00000000 00000080}\\
& \texttt{00000000 00000080 00000000 00040000 00000000 00000000}.
\end{aligned}
$$

The number of conditions imposed by $\Delta_a$ and $\Delta_b$ are 127 and 134, respectively. Despite its lower raw probability, $\Delta_a$ has a theoretical complexity of $2^{31.9}$ compression function calls, which is much less than $2^{68.7}$ for $\Delta_b$. As discussed above, this is due to the different structure of $\boldsymbol{\alpha} \vee \boldsymbol{\beta}$. We recall that $y$ denotes the Hamming weight of $\boldsymbol{\alpha} \vee \boldsymbol{\beta}$ and let

$$y_i = \sum_{k=32i}^{32(i+1)} \mathrm{wt}(\alpha^k \vee \beta^k).$$

That is, $y_i$ is the number of conditions imposed by a difference at round $i$. Table 4 compares the values $y_i$ for $\Delta_a$ and $\Delta_b$. The conditions imposed in the first two rounds can easily be satisfied by appropriate message modifications, and thus, do not significantly increase the complexity of the attack — contrary to the conditions imposed in the last two rounds.

Due to its low theoretical complexity, we can use $\Delta_b$ to find a collision and to confirm empirically the estimated theoretical complexity.

**Table 4.** Number of conditions per round theoretical complexities

| | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y$ | $\log_2(c_\Delta)$ |
|---|---|---|---|---|---|---|---|
| $a$ | 14 | 17 | 23 | 30 | 43 | 127 | 68.7 |
| $b$ | 44 | 36 | 25 | 17 | 12 | 134 | 31.9 |

**Table 5.** Partition sets corresponding to the trail $\Delta_b$ for CubeHash-5/96. Numbers in $\mathcal{M}_i$ are byte indices, whereas numbers in $\mathcal{Y}_i$ are bit indices.

| $i$ | $\mathcal{M}_i$ | $\mathcal{Y}_i$ | $q_i$ |
|---|---|---|---|
| 0 | $\emptyset$ | $\emptyset$ | 0.00 |
| 1 | $\{2, 6, 66\}$ | $\{1, 2\}$ | 2.00 |
| 2 | $\{10, 1, 9, 14, 74, 5, 13, 65, 17, 70\}$ | $\{5\}$ | 1.35 |
| 3 | $\{73, 7, 16, 19, 18, 78, 25, 37, 41\}$ | $\{23, 24\}$ | 2.00 |
| 4 | $\{69, 77, 24, 33\}$ | $\{21, 22\}$ | 2.00 |
| 5 | $\{50, 89\}$ | $\{12, 13\}$ | 2.00 |
| 6 | $\{20, 27, 45, 88\}$ | $\{11\}$ | 1.15 |
| 7 | $\{57, 4\}$ | $\{38\}$ | 1.00 |
| 8 | $\{80\}$ | $\{7, 8\}$ | 2.00 |
| 9 | $\{38, 40, 81, 3, 28, 32\}$ | $\{34\}$ | 1.24 |
| 10 | $\{49\}$ | $\{41\}$ | 1.00 |
| 11 | $\{58\}$ | $\{19, 20, 42, 43\}$ | 4.00 |
| 12 | $\{91\}$ | $\{16, 17\}$ | 2.00 |
| 13 | $\{23, 34, 44, 83\}$ | $\{29, 30\}$ | 2.07 |
| 14 | $\{90\}$ | $\{14\}$ | 1.07 |
| 15 | $\{15, 26\}$ | $\{15\}$ | 1.07 |
| 16 | $\{36\}$ | $\{37, 55\}$ | 2.31 |
| 17 | $\{42, 46, 48\}$ | $\{25, 26\}$ | 2.12 |
| 18 | $\{56\}$ | $\{18, 31, 40\}$ | 3.01 |
| 19 | $\{59\}$ | $\{48, 79\}$ | 2.00 |
| 20 | $\{84, 92, 0\}$ | $\{35\}$ | 1.00 |
| 21 | $\{82\}$ | $\{9, 10, 27, 28, 32, 33\}$ | 6.04 |
| 22 | $\{31, 51\}$ | $\{44, 56, 64\}$ | 3.03 |
| 23 | $\{71\}$ | $\{6\}$ | 1.00 |
| 24 | $\{11, 54, 67\}$ | $\{3\}$ | 1.00 |
| 25 | $\{75\}$ | $\{78\}$ | 1.00 |
| 26 | $\{21, 55\}$ | $\{46, 59\}$ | 2.00 |
| 27 | $\{63\}$ | $\{50\}$ | 1.00 |
| 28 | $\{79\}$ | $\{45, 49, 65, 70\}$ | 4.00 |
| 29 | $\{12\}$ | $\{71\}$ | 1.06 |
| 30 | $\{22\}$ | $\{58, 67, 81, 82, 83\}$ | 5.00 |
| 31 | $\{29, 62\}$ | $\{63\}$ | 1.03 |
| 32 | $\{87, 95\}$ | $\{53, 54, 74, 76, 85\}$ | 5.01 |
| 33 | $\{39, 47\}$ | $\{39\}$ | 1.01 |
| 34 | $\{53, 8\}$ | $\{69, 88, 89\}$ | 3.30 |
| 35 | $\{30\}$ | $\{77, 86, 94, 98\}$ | 5.04 |
| 36 | $\{60, 61\}$ | $\{62, 91, 101, 102\}$ | 4.35 |
| 37 | $\{35, 52\}$ | $\{61, 90, 103\}$ | 4.22 |
| 38 | $\{43\}$ | $\{36, 57, 60, 104, 111\}$ | 5.77 |
| 39 | $\{64\}$ | $\{0\}$ | 1.33 |
| 40 | $\{68\}$ | $\{4\}$ | 2.03 |
| 41 | $\{72\}$ | $\{97, 100, 121\}$ | 8.79 |
| 42 | $\{76\}$ | $\{66, 80, 92, 93\}$ | 13.39 |
| 43 | $\{85\}$ | $\{47, 112\}$ | 16.92 |
| 44 | $\{93\}$ | $\{51, 52, 68, 72, 75, 87, 95\}$ | 22.91 |
| 45 | $\{86, 94\}$ | $\{73, 84, 96, 99, 105, \ldots, 110, 113, \ldots, 132, 133\}$ | 31.87 |

### 5.2 Finding the Collision

According to Section 3.2, we construct a collision for the hash function out of a collision for the compression function. Using a dependency table at byte-level, we obtained a partition of the condition function attributed to $\Delta_b^0$ (see Table 5). Then, using the tree-based backtracking algorithm proposed in [4], we found several collisions after $2^{22.41}$ to $2^{32.25}$ condition function calls. One of them, found after $2^{29.1}$ condition function calls, is given by

$$
\begin{aligned}
M^{\text{pre}} = {} & \texttt{F06BB068 487C5FE1 CCCABA70 0A989262 801EDC3A 69292196} \\
& \texttt{8848F445 B8608777 C037795A 10D5D799 FD16C037 A52D0B51} \\
& \texttt{63A74C97 FD858EEF 7809480F 43EB264C D6631863 2A8CCFE2} \\
& \texttt{EA22B139 D99E4888 8CA844FB ECCE3295 150CA98E B16B0B92}, \\
M^0 = {} & \texttt{3DB4D4EE 02958F57 8EFF307A 5BE9975B 4D0A669E E6025663} \\
& \texttt{8DDB6421 BAD8F1E4 384FE128 4EBB7E2A 72E16587 1E44C51B} \\
& \texttt{DA607FD9 1DDAD41F 4180297A 1607F902 2463D259 2B73F829} \\
& \texttt{C79E766D 0F672ECC 084E841B FC700F05 3095E865 8EEB85D5}.
\end{aligned}
$$

For $M^1 = 0$, the messages $M^{\text{pre}} \| M^0 \| M^1$ and $M^{\text{pre}} \| M^0 \oplus \Delta_b^0 \| M^1 \oplus \Delta_b^1$ collide to the same digest

$$
\begin{aligned}
H = {} & \texttt{C2E51517 C503746E 46ECD6AD 5936EC9B} \\
& \texttt{FF9B74F9 2CEA4506 624F2B0B FE584D2C} \\
& \texttt{56CD3E0E 18853BA8 4A9D6D38 F1F8E45F} \\
& \texttt{2129C678 CB3636D4 D865DE13 410E966C}
\end{aligned}
$$

under CubeHash-5/96. Instead of $M^1 = 0$, any other $M^1$ can be chosen and, moreover, the colliding messages can be extended by an arbitrary message suffix $M^{\text{suff}}$.

## 6 Conclusion

In this paper we used two methods for finding improved linear differential trails for CubeHash. The method of backward computation lead to the first practical collision attack on CubeHash-5/96. The randomized search yielded new highly probable differential trails which lead to improved collision attacks for up to eight rounds. Both methods may also apply to collision attacks on other hash functions.

Our analysis did not lead to an attack on the official CubeHash-16/32.

## Acknowledgements

This work was partially supported by European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. The second author is

supported by the Hasler Foundation www.haslerfoundation.ch under project number 08065. The third author is supported by GEBERT RÜF STIFTUNG under project number GRS-069/07.

## References

1. Bernstein, D.J.: Cubehash. Submission to NIST (2008)
2. Bernstein, D.J.: Cubehash. Submission to NIST, Round 2 (2009)
3. Biham, E., Chen, R.: Near-Collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
4. Brier, E., Khazaei, S., Meier, W., Peyrin, T.: Linearization Framework for Collision Attacks: Application to CubeHash and MD6. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 560–577. Springer, Heidelberg (2009)
5. Brier, E., Khazaei, S., Meier, W., Peyrin, T.: Linearization Framework for Collision Attacks: Application to CubeHash and MD6 (extended version). Cryptology ePrint Archive, Report 2009/382 (2009), http://eprint.iacr.org
6. Brier, E., Peyrin, T.: Cryptanalysis of CubeHash. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 354–368. Springer, Heidelberg (2009)
7. Chabaud, F., Joux, A.: Differential collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)
8. Indesteege, S., Preneel, B.: Practical Collisions for EnRUPT. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 246–259. Springer, Heidelberg (2009)
9. Naito, Y., Sasaki, Y., Shimoyama, T., Yajima, J., Kunihiro, N., Ohta, K.: Improved Collision Search for SHA-0. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 21–36. Springer, Heidelberg (2006)
10. National Institute of Standards and Techonolgy. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithms (SHA-3) Family. Federal Register, 72 (2007)
11. Pramstaller, N., Rechberger, C., Rijmen, V.: Exploiting coding theory for collision attacks on SHA-1. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 78–95. Springer, Heidelberg (2005)
12. Rijmen, V., Oswald, E.: Update on SHA-1. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 58–71. Springer, Heidelberg (2005)
13. Shoup, V.: NTL: A Library for doing Number Theory. Version 5.5.2, http://www.shoup.net/ntl
14. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

# Cryptanalysis of the 10-Round Hash and Full Compression Function of SHAvite-3-512*

Praveen Gauravaram[1], Gaëtan Leurent[2], Florian Mendel[3], María Naya-Plasencia[4], Thomas Peyrin[5], Christian Rechberger[6], and Martin Schläffer[3]

[1] Department of Mathematics, DTU, Denmark
[2] ENS, France
[3] IAIK, TU Graz, Austria
[4] FHNW Windisch, Switzerland
[5] Ingenico, France
[6] ESAT/COSIC, K.U.Leuven and IBBT, Belgium
martin.schlaeffer@iaik.tugraz.at

**Abstract.** In this paper, we analyze the SHAvite-3-512 hash function, as proposed and tweaked for round 2 of the SHA-3 competition. We present cryptanalytic results on 10 out of 14 rounds of the hash function SHAvite-3-512, and on the full 14 round compression function of SHAvite-3-512. We show a second preimage attack on the hash function reduced to 10 rounds with a complexity of $2^{497}$ compression function evaluations and $2^{16}$ memory. For the full 14-round compression function, we give a chosen counter, chosen salt preimage attack with $2^{384}$ compression function evaluations and $2^{128}$ memory (or complexity $2^{448}$ without memory), and a collision attack with $2^{192}$ compression function evaluations and $2^{128}$ memory.

**Keywords:** hash function, cryptanalysis, collision, (second) preimage.

## 1 Introduction

With the advent of new cryptanalysis [6,20] of the FIPS 180-2 standard hash function SHA-1 [14], NIST has initiated an open hash function competition [15]. SHAvite-3 [3,4], a hash function designed by Biham designed by Biham and Dunkelman, is a second round candidate in the NIST's SHA-3 hash function competition [16]. It is an iterated hash function based on the HAIFA hash function framework [2]. In this framework, the compression functions also accepts a

* This work was supported by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy). Parts of this work were carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme and while authors were participating the ECRYPT2 workshop "Hash$^3$: Proofs, Analysis and Implementation" in November 2009. The first author is supported by the Danish Council for Independent Research (FTP and FNU) grant 274-09-0096.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 419–436, 2010.

salt and counter input in addition to the chaining value and message block. The mixing of the salt with the message aims to increase the security of hash-then-sign digital signatures against offline collision attacks [9,10], whereas the counter aims to thwart any attempts to mount some generic attacks on the iterated hash functions [1,8,11].

The SHAvite-3 compression function consists of a generalized Feistel structure whose round function is based on the AES. SHAvite-3 proposes two different instances called SHAvite-3-256 and SHAvite-3-512 with 12 and 14 rounds respectively. The first round version of SHAvite-3 has been tweaked for the second round of the SHA-3 competition due to a chosen salt and chosen counter collision attack [18] on the compression function. Recently, Bouillaguet et al. [5] have proposed the cancellation property to analyse hash functions based on a generalized Feistel structure. They have applied their method to find second preimages for 9 rounds of the SHAvite-3-512 hash function.

In this paper, we further analyze SHAvite-3-512 by improving the previous analysis of Bouillageut et al. [5]. We first present a chosen counter, chosen salt collision and preimage attack for the full compression function of SHAvite-3-512. The complexity for the preimage attack is $2^{384}$ compression function evaluations and $2^{128}$ memory (or complexity $2^{448}$ without memory), and for the collision attack we get $2^{192}$ compression function evaluations and $2^{128}$ memory. We then propose a second preimage attack on the hash function reduced to 10 rounds with a complexity of $2^{497}$ compression function evaluations and $2^{16}$ memory.

The paper is organised as follows: In Section 2, we briefly describe the SHAvite-3-512 hash function and in Section 3, we provide the fundamental ideas used in our attacks. In Section 4, we provide a preimage and collision attacks for the full 14 round compression function. In Section 5, we present a second preimage attack on the 10 round hash function and we conclude in Section 6.

## 2 The SHAvite-3-512 Hash Function

SHAvite-3-512 is used for the hash sizes of $n = 257, \ldots, 512$ bits. First, the message $M$ is padded and split into $\ell$ 1024-bit message blocks $M_1 \| M_2 \| \ldots \| M_\ell$. Then, each message block is iteratively processed using the 512-bit compression function $C_{512}$ and finally, the output is truncated to the desired hash size as follows:

$$\begin{aligned} h_0 &= IV \\ h_i &= C_{512}(h_{i-1}, M_i, salt, cnt) \\ hash &= trunc_n(h_i) \end{aligned}$$

The 512-bit compression function $C_{512}$ of SHAvite-3-512 consists of a 512-bit block cipher $E^{512}$ used in Davies-Meyer mode. The input of the compression function $C_{512}$ consists of a 512-bit chaining value $h_{i-1}$, a 1024-bit message block $M_i$, a 512-bit *salt* and a 128-bit counter (*cnt*) to denote the number of bits processed by the end of the iteration. The output of the compression function $C_{512}$ is given by (+ denotes an XOR addition):

$$h_i = C_{512}(h_{i-1}, M_i, salt, cnt) = h_{i-1} + E^{512}(M_i \| salt \| cnt, h_{i-1})$$

### 2.1 State Update

The state update of the compression function consists of a 14-round generalized Feistel structure. The input $h_{i-1}$ is divided into four 128-bit chaining values $(A_0, B_0, C_0, D_0)$. In each round $i = 0, \ldots, 13$, these chaining values are updated using the non-linear round functions $F_i$ and $F'_i$ by the Feistel structure given as follows (also see Figure 1):

$$(A_{i+1}, B_{i+1}, C_{i+1}, D_{i+1}) = (D_i, A_i + F_i(B_i), B_i, C_i + F'_i(D_i))$$

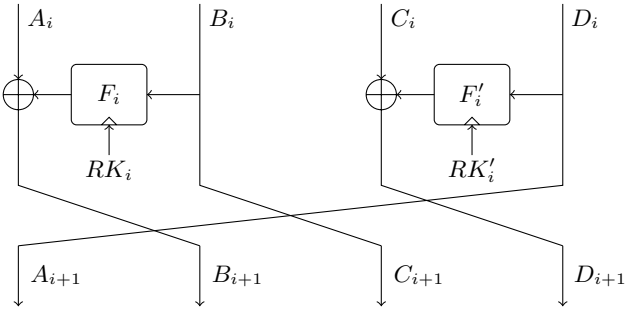



**Fig. 1.** Round $i$ of the state update of SHAvite-512

The non-linear functions $F_i$ and $F'_i$ are keyed by the 512-bit round keys $RK_i = (k^3_{0,i}, k^2_{0,i}, k^1_{0,i}, k^0_{0,i})$ and $RK'_i = (k^3_{1,i}, k^2_{1,i}, k^1_{1,i}, k^0_{1,i})$ respectively. Each round function is composed of four AES rounds with subkeys $k^0_{0,i}$ and $k^0_{1,i}$ used as a key whitening before the first AES round and an all zero-key $0^{128}$ for the last internal round. Hence, the round functions $F_i$ and $F'_i$ are defined as:

$$F_i(x) = AES(0^{128}, AES(k^3_{0,i}, AES(k^2_{0,i}, AES(k^1_{0,i}, k^0_{0,i} + x)))) \quad (1)$$

$$F'_i(x) = AES(0^{128}, AES(k^3_{1,i}, AES(k^2_{1,i}, AES(k^1_{1,i}, k^0_{1,i} + x)))) \quad (2)$$

### 2.2 Message Expansion

The message expansion of $C_{512}$ (the key schedule of $E^{512}$) takes as input a 1024-bit message block, a 512-bit salt and 128-bit counter. The 1024-bit message block $M_i$ is represented as an array of 8 128-bit words $(m_0, m_1, \ldots, m_7)$, the 512-bit salt as an array of 16 32-bit words $(s_0, s_1, \ldots, s_{15})$ and the counter as an array of 4 32-bit words $(cnt_0, cnt_1, cnt_2, cnt_3)$.

The subkeys for the odd rounds of the state update are generated using parallel AES rounds and a subsequent linear expansion step. The AES rounds are keyed by the salt words. The subkeys for the even rounds are computed using only a linear layer. One out of $r = 0, \ldots, 7$ rounds of the message expansion is shown in Figure 2. The first subkeys of round $r = 0$ are initialized with the message block:

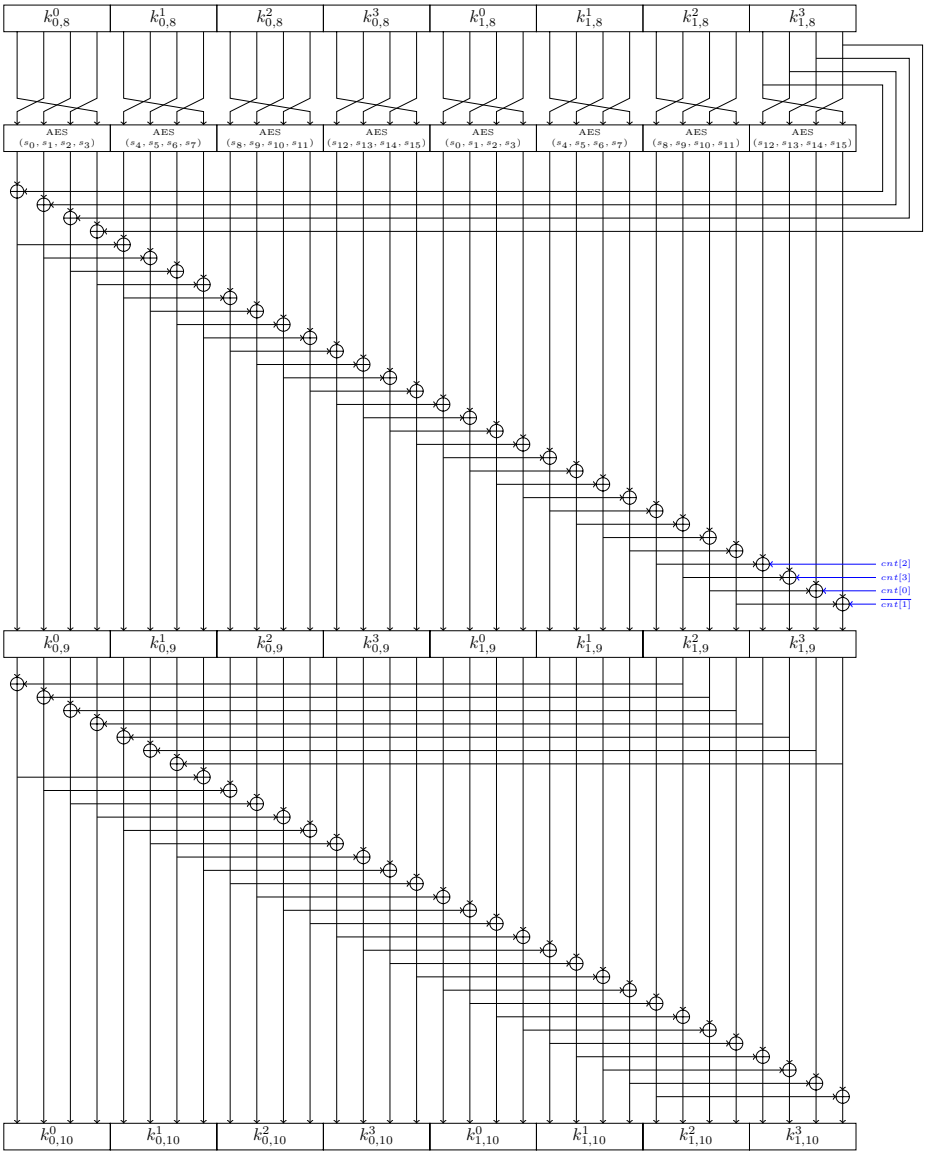



**Fig. 2.** One round ($r = 2$) of the message expansion (key schedule) of SHAvite-512 with the counter values ($cnt_2 \| cnt_3 \| cnt_0 \| \overline{cnt_1}$) added prior to subkey $k^3_{1,9}$

$$(k^0_{0,0}, k^1_{0,0}, k^2_{0,0}, k^3_{0,0}, k^0_{1,0}, k^1_{1,0}, k^2_{1,0}, k^3_{1,0}) = (m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7)$$

Note that in rounds $r = 0, 2, 4, 6$ of the message expansion, the counter value (with one word inverted) is added to the subkeys:

$$k^0_{0,1} = k^0_{0,1} + (cnt_0 \| cnt_1 \| cnt_2 \| \overline{cnt_3})$$
$$k^1_{0,5} = k^1_{0,5} + (cnt_3 \| cnt_2 \| cnt_1 \| \overline{cnt_0})$$
$$k^3_{1,9} = k^3_{1,9} + (cnt_2 \| cnt_3 \| cnt_0 \| \overline{cnt_1})$$
$$k^2_{1,13} = k^2_{1,13} + (cnt_1 \| cnt_0 \| cnt_3 \| \overline{cnt_2})$$

We refer to [4] for additional details on the message expansion.

## 3 Basic Attack Strategy

We first show how we can keep one chaining value of the state update constant using the cancellation property. Then, we interleave the cancellation property to simplify the resulting conditions and to keep the chaining value constant for a larger number of rounds. These properties can be used to construct partial preimages of the compression function. Note that we define a partial preimage attack as the task to find a preimage for only parts of the target hash value. Then, we extend this partial preimage attack to a collision or preimage attack of the compression function, or a (second) preimage of the hash function.

### 3.1 The Cancellation Property

The cancellation property was published by Bouillaguet et al. in the analysis of the SHA-3 candidates Lesamta and SHAvite-3 [5]. Using the cancellation property, a disturbance introduced in one state variable cancels itself 2 rounds later again. In our attacks on SHAvite-3-512, the cancellation property is used to ensure that $B_i = B_{i+4}$ in the state update. This is the case if and only if $F_{i+3}(B_{i+3}) = F'_{i+1}(D_{i+1})$ (see Table 1). Hence, a disturbance $F'_{i+1}(D_{i+1})$ introduced in round $i+1$ cancels itself in round $i+3$ (also see Figure 3).

Using $B_{i+3} = D_{i+1} + F_{i+2}(B_{i+2})$ and Equations (1) and (2) we get:

$$AES(0^{128}, AES(k^3_{0,i+3}, AES(k^2_{0,i+3}, AES(k^1_{0,i+3}, k^0_{0,i+3} + D_{i+1} + F_{i+2}(B_{i+2}))))) =$$
$$AES(0^{128}, AES(k^3_{1,i+1}, AES(k^2_{1,i+1}, AES(k^1_{1,i+1}, k^0_{1,i+1} + D_{i+1})))).$$

This equation and thus, the characteristic of Table 1, is fulfilled under the following conditions:

$$(k^3_{0,i+3}, k^2_{0,i+3}, k^1_{0,i+3}) = (k^3_{1,i+1}, k^2_{1,i+1}, k^1_{1,i+1}) \tag{3}$$

**Table 1.** We use the cancellation property to ensure that $B_i = B_{i+4}$. This is the case if $F_{i+3}(B_{i+3})$ and $F'_{i+1}(D_{i+1})$ are equal.

| $i$ | $A_i$ | $B_i$ | $C_i$ | $D_i$ | *condition* |
|---|---|---|---|---|---|
| $i$ | ? | $B_i$ | ? | ? | |
| $i+1$ | ? | ? | $B_i$ | $D_{i+1}$ | |
| $i+2$ | $D_{i+1}$ | $B_{i+2}$ | ? | $B_i + F'_{i+1}(D_{i+1})$ | $F_{i+3}(B_{i+3}) = F'_{i+1}(D_{i+1})$ |
| $i+3$ | $B_i + F'_{i+1}(D_{i+1})$ | $B_{i+3}$ | $B_{i+2}$ | ? | |
| $i+4$ | ? | $B_i$ | ? | ? | |

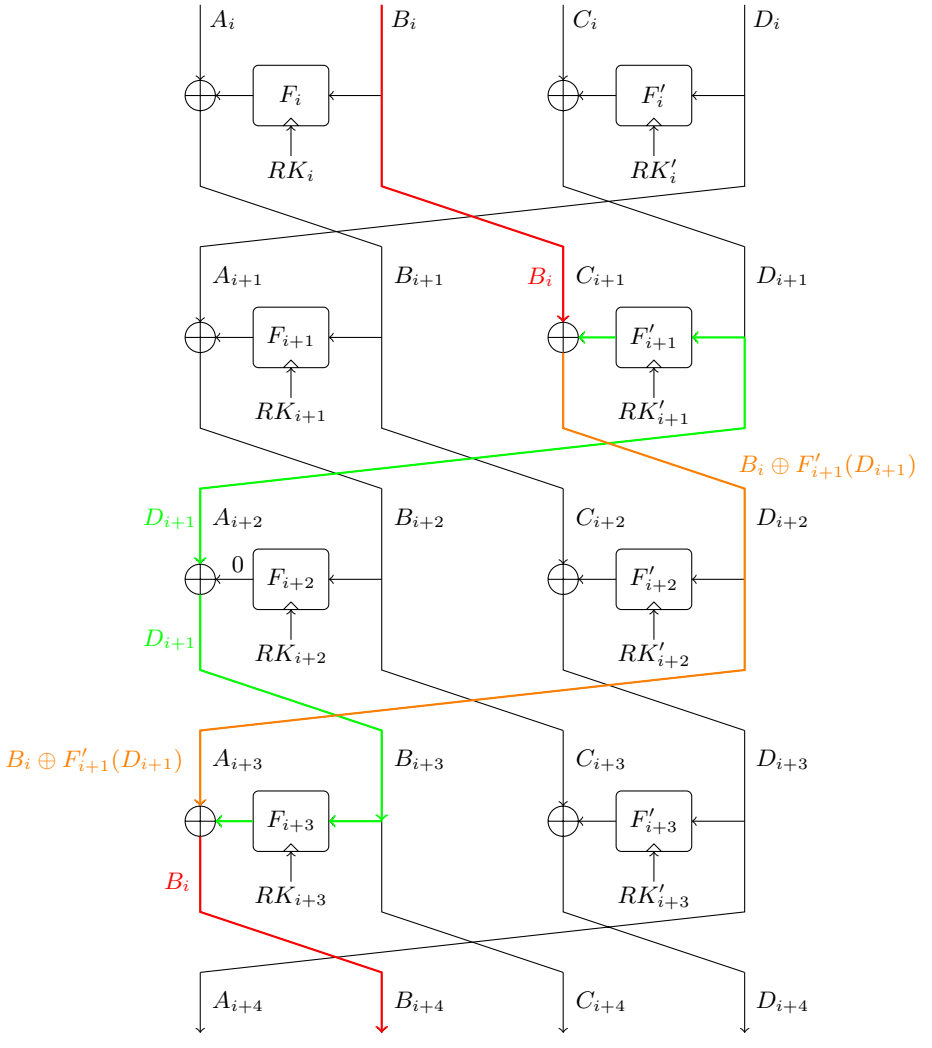



**Fig. 3.** We need to ensure that $B_i = B_{i+4}$ for a number of times. This is the case if $F_{i+3}(B_{i+3}) = F'_{i+1}(D_{i+1})$. However, we use the following sufficient conditions which can be fulfilled more easily using interleaving: $F_{i+2}(B_{i+2}) = 0$ and $RK_{i+3} = RK'_{i+1}$.

and

$$k^0_{0,i+3} + F_{i+2}(B_{i+2}) = k^0_{1,i+1}. \tag{4}$$

Hence, using the cancellation property we always get $B_i = B_{i+4}$, no matter which value $D_{i+1}$ has. Also differences in $D_{i+1}$ are cancelled in round $i + 4$ again. Of course, we can repeat the cancellation property for other rounds as long as we can fulfill the resulting conditions. The cancellation property is used

twice in the 9-round hash attack on SHAvite-3-512 by Bouillaguet et al. [5] and in the 10-round hash attack in Section 5, and 4 times in the 13 and 14-round attack of the compression function in Section 4.

### 3.2 Interleaving

In this work, we extend the 9 round attack on SHAvite-3-512 by several rounds. This is possible by interleaving the cancellation property such that the conditions can be fulfilled more easily. Note that equation (4) depends on the chaining value $B_{i+2}$. In our attack, we use the following sufficient conditions, which allow us to fulfill the conditions on the keys and chaining values independently:

$$F_{i+2}(B_{i+2}) = 0 \tag{5}$$

$$RK_{i+3} = RK'_{i+1}. \tag{6}$$

Figure 3 shows that if the output of $F_{i+2}(B_{i+2})$ is zero, the same input $D_{i+1}$ enters the round functions $F_{i+3}$ and $F'_{i+1}$. Hence, if the keys of these non-linear functions are equal, any disturbance introduced by $D_{i+1}$ cancels itself two rounds later. Note that we get equivalent conditions if we continue with $B_{i+4} = B_{i+8}$ and interleave with $B_{i+2} = B_{i+6}$. Hence, we get

$$\begin{aligned} B_i &= B_{i+4} = B_{i+8} = \dots \\ B_{i+2} &= B_{i+6} = B_{i+10} = \dots \end{aligned}$$

if the following conditions on the chaining values are fulfilled:

$$\begin{aligned} &F_{i+2}(B_{i+2}) = 0, \;\; F_{i+6}(B_{i+2}) = 0, \;\; F_{i+10}(B_{i+2}) = 0, \dots \\ &F_{i+4}(B_i) = 0, \;\; F_{i+8}(B_i) = 0, \;\; F_{i+12}(B_i) = 0, \dots \end{aligned}$$

Since $F$ is an invertible function, all these conditions are fulfilled if we choose $B_i = F^{-1}_{i+4}(0)$ and $B_{i+2} = F^{-1}_{i+2}(0)$, and the keys of the respective functions are the same. Hence, we get the following conditions only on the keys:

$$RK_{i+2} = RK_{i+6} = RK_{i+10} = \dots \tag{7}$$

$$RK_{i+4} = RK_{i+8} = RK_{i+12} = \dots \tag{8}$$

$$RK_{i+3} = RK'_{i+1}, \;\; RK_{i+5} = RK'_{i+3}, \;\; RK_{i+7} = RK'_{i+5}, \dots \tag{9}$$

An example starting with $B_3$ is given in Table 2.

### 3.3 From Partial Preimages to Preimages and Collisions

In the following attacks on SHAvite-3-512, we show how to fix one 128-bit output word $H_i$ of the (reduced) compression function $C_{512}$ to some predefined value:

$$H_0 \| H_1 \| H_2 \| H_3 = C_{512}(A_0 \| B_0 \| C_0 \| D_0, msg, salt, cnt)$$

**Table 2.** Interleaving

| $i$ | $A_i$ | $B_i$ | $C_i$ | $D_i$ | *conditions* |
|---|---|---|---|---|---|
| 3 | ? | $B_3$ | ? | ? | |
| 4 | ? | ? | $B_3$ | $D_4$ | |
| 5 | $D_4$ | $B_5$ | ? | $B_3 + F'_4(D_4)$ | $F_5(B_5) = 0$ |
| 6 | $B_3 + F'_4(D_4)$ | $D_4$ | $B_5$ | $D_6$ | $RK_6 = RK'_4$ |
| 7 | $D_6$ | $B_3$ | $D_4$ | $B_5 + F'_6(D_6)$ | $F_7(B_3) = 0$ |
| 8 | $B_5 + F'_6(D_6)$ | $D_6$ | $B_3$ | $D_8$ | $RK_8 = RK'_6$ |
| 9 | $D_8$ | $B_5$ | $D_6$ | $B_3 + F'_8(D_8)$ | $RK_9 = RK_5$ |
| 10 | $B_3 + F'_8(D_8)$ | $D_8$ | $B_5$ | $D_{10}$ | $RK_{10} = RK'_8$ |
| 11 | $D_{10}$ | $B_3$ | $D_8$ | $B_5 + F'_{10}(D_{10})$ | $RK_{11} = RK_7$ |
| ... | ... | ... | ... | ... | ... |

Let's assume, we are able to construct a partial preimage on $H_0$ with a complexity of $2^x < 2^{128}$. Then, this partial preimage can be extended to construct a preimage or collision for the compression function below the generic complexity.

In a preimage attack on the compression function, we have to find some input values $A_0 \| B_0 \| C_0 \| D_0, msg, salt, cnt$ to the compression function for a given output $H_0 \| H_1 \| H_2 \| H_3$. For example, by repeating a partial preimage attack on $H_0$ about $2^{384}$ times, we expect to find a chaining value where also $H_1$, $H_2$ and $H_3$ are correct. In other words, we can find a preimage for the (reduced) compression function of SHAvite-512 with a complexity of about $2^{384+x}$.

Similarly, a collision for the compression function can be constructed. If we can find $2^{192}$ inputs $A_0 \| B_0 \| C_0 \| D_0, msg, salt, cnt$ such that all produce the same output value $H_0$, two of the inputs also lead to the same values $H_1$, $H_2$, and $H_3$ due to the birthday paradox. Hence, we can construct a collision for the compression function of SHAvite-512 with a complexity of $2^{192+x}$.

Further, by using an arbitrary first message block and a standard meet-in-the-middle attack we can turn the preimage attack on the compression function into a (second) preimage attack on the hash function. Note that in this case the *salt* and *cnt* values need to be the same for all partial preimages.

## 4 Attacks on the Compression Function

In this section, we present preimage and collision attacks for the SHAvite-3-512 compression function reduced to 13 and 14 rounds. We first give an outline of the attack and describe the characteristic used to find partial preimages. Then, we show how to find a message, salt and counter value according to the conditions of this characteristic. Finally, we extend the attack to find many message, salt and counter values such that the partial preimage attack can be extended to find a collision and preimage for 13 and 14 rounds. Note that the given preimage attack is an s-Pre (enhanced preimage) attack, as defined by Reyhanitabar et al. [19]. However, the attacks on the compression fucntion can not be extended to the hash function, since in the attack we have to choose the salt and counter value.

### 4.1 Outline of the Attack

In the attack on 13 and 14 rounds of the compression function, we use the same idea as in the 9-round attack in [5], but we extend it to more rounds by interleaving more cancellations. In detail, we use the cancellation property four times at rounds 6, 8, 10 and 12. This requires several 128-bit equalities on the key-schedule. We can satisfy the equalities using the degrees of freedom an attacker has in the choice of the message, salt, counter and chaining value in an attack on the compression function. Both, the 13 and 14 round attacks use the characteristic given in Table 3. For the 13 round attack we omit the last round.

**Table 3.** Characteristic for the attack on 13 and 14 rounds of the compression function. We can keep the value $Z$ constant as long as the conditions are fulfilled.

| $i$ | $A_i$ | $B_i$ | $C_i$ | $D_i$ | *conditions* |
|---|---|---|---|---|---|
| 0 | ? | ? | ? | ? | |
| 1 | ? | ? | ? | ? | |
| 2 | ? | $X$ | ? | ? | |
| 3 | ? | $Z$ | $X$ | ? | |
| 4 | ? | $Y$ | $Z$ | $D_4$ | |
| 5 | $D_4$ | $Z$ | $Y$ | $Z + F'_4(D_4)$ | $F_5(Z) = 0$ |
| 6 | $Z + F'_4(D_4)$ | $D_4$ | $Z$ | $D_6$ | $RK_6 = RK'_4$ |
| 7 | $D_6$ | $Z$ | $D_4$ | $Z + F'_6(D_6)$ | $RK_7 = RK_5$ |
| 8 | $Z + F'_6(D_6)$ | $D_6$ | $Z$ | $D_8$ | $RK_8 = RK'_6$ |
| 9 | $D_8$ | $Z$ | $D_6$ | $Z + F'_8(D_8)$ | $RK_9 = RK_7$ |
| 10 | $Z + F'_8(D_8)$ | $D_8$ | $Z$ | $D_{10}$ | $RK_{10} = RK'_8$ |
| 11 | $D_{10}$ | $Z$ | $D_8$ | $Z + F'_{10}(D_{10})$ | $RK_{11} = RK_9$ |
| 12 | $Z + F'_{10}(D_{10})$ | $D_{10}$ | $Z$ | ? | $RK_{12} = RK'_{10}$ |
| 13 | ? | $Z$ | $D_{10}$ | ? | $RK_{13} = RK_{11}$ |
| 14 | ? | ? | $Z$ | ? | |

We start the attack using $X, Z, Y, Z$ for $B_2, B_3, B_4, B_5$. Choosing $B_3 = B_5 = Z$ gives slightly simpler and more uniform conditions for the key schedule which can be fulfilled easier (see Section 4.2 and 4.3). The characteristic requires that the outputs of the functions $F_5$, $F_7$, $F_9$ and $F_{11}$ are zero and we get:

$$F_5(Z) = F_7(Z) = F_9(Z) = F_{11}(Z) = 0 \ .$$

As already mentioned in the previous section, the best way to guarantee that the above conditions hold is to ensure that the four subkeys $RK_5, RK_7, RK_9, RK_{11}$ and hence, the functions $F_5, F_7, F_9, F_{11}$ are equal. In this case $Z$ can be easily computed by $Z = F_5^{-1}(0)$ and we get the following conditions for the key-schedule:

$$RK_5 = RK_7 = RK_9 = RK_{11} \ .$$

Next, we need to ensure that the output of $F'_4(D_4)$ and $F_6(D_4)$ is equal to one another such that $B_7 = Z + F'_4(D_4) + F_6(D_4) = Z$ after round 7. In total, the characteristic specifies that:

- $F'_4(D_4) = F_6(D_4)$ such that $B_7 = Z$
- $F'_6(D_6) = F_8(D_6)$ such that $B_9 = Z$
- $F'_8(D_8) = F_{10}(D_8)$ such that $B_{11} = Z$
- $F'_{10}(D_{10}) = F_{12}(D_{10})$ such that $B_{13} = Z$

and we get the following conditions on the key schedule for $i = 4, 6, 8, 10$ (also see Table 3):

$$RK'_i = RK_{i+2} \ .$$

If we can find a message, salt and counter value, such that all conditions on the state values and key-schedule are fulfilled, the characteristic of Table 3 is followed from round $i = 5$ to the end. Then, we simply compute backwards to get the inputs $(A_0, B_0, C_0, D_0)$ of the compression function as a function of $X$, $Y$ and $Z$ (see also Table 4). Note that we can choose only values for $X$ and $Y$ since $Z$ is fixed by the attack:

$$A_0 = Z + F_4(Y) + F'_2(Y + F_3(Z)) + F_0(Y + F_3(Z) + F'_1(Z + F_2(X))) \quad (10)$$
$$B_0 = Y + F_3(Z) + F'_1(Z + F_2(X)) \quad (11)$$
$$C_0 = Z + F_2(X) + F'_0(X + F_1(Z + F_4(Y) + F'_2(Y + F_3(Z)))) \quad (12)$$
$$D_0 = X + F_1(Z + F_4(Y) + F'_2(Y + F_3(Z))) \quad (13)$$

**Table 4.** We get the input of the 14-round characteristic as a function of $X$,$Y$ and $Z$ by computing backwards from round $i = 4$. We only show the updated chaining values $A_i$ and $C_i$ since $B_i = C_{i+1}$ and $D_i = A_{i+1}$.

| $i$ | $A_i$ | $C_i$ |
|---|---|---|
| 0 | ? | $Z + F_2(X) + F'_0(X + F_1(Z + \ldots + F_4(Y) + F'_2(Y + F_3(Z))))$ |
| 1 | $X + F_1(Z + F_4(Y) + F'_2(Y + F_3(Z)))$ | $Y + F_3(Z) + F'_1(Z + F_3(X))$ |
| 2 | $Z + F_2(X)$ | $Z + F_4(Y) + F'_2(Y + F_3(Z))$ |
| 3 | $Y + F_3(Z)$ | $X$ |
| 4 | $Z + F_4(Y)$ | $Z$ |
| 5 | $D_4$ | $Y$ |
| 6 | $Z + F'_4(D_4)$ | $Z$ |
| 7 | $D_6$ | $D_4$ |
| 8 | $Z + F'_6(D_6)$ | $Z$ |
| 9 | $D_8$ | $D_6$ |
| 0 | $Z + F'_8(D_8)$ | $Z$ |
| 11 | $D_{10}$ | $D_8$ |
| 12 | $Z + F'_{10}(D_{10})$ | $Z$ |
| 13 | ? | $D_{10}$ |
| 14 | ? | $Z$ |

### 4.2 Finding the Message

An easy solution to fulfill all the conditions on the subkeys is to ensure that all keys are equal. This was possible for the round 1 version of SHAvite-512. By setting the message to the all zero value, each byte of the salt to `0x52` and the counter to 0, all subkeys are equal to zero [18]. In this case the required conditions are trivially fulfilled.

For the second round of the SHA-3 competition, SHAvite-512 has been tweaked and some counter words are inverted to prevent all zero keys. However, by choosing the counter to be

$$(cnt_3, cnt_2, cnt_1, cnt_0) = (0, 0, \overline{0}, 0),$$

the value $(cnt_2, cnt_3, cnt_0, \overline{cnt_1})$ added in round 5 of the key schedule is zero (see Figure 2). In contrast to the attack on the round 1 version, this results in a valid counter value. If we choose each byte of the salt to be `0x52` and the subkeys of round 5 to be zero, a large part of the subkeys will remain zero until non-zero values are added by the counter again. This happens in round 3 and round 7 of the key schedule. We only require that subkeys with round index $i = 4, \ldots, 13$ are equal. Table 5 shows that indeed all required subkeys can be forced to zero. By computing backwards we get the message which is given in Appendix A. Since the key $RK_5 = 0$, we further get for $Z = F_5^{-1}(0) =$ `0x1919...19`.

**Table 5.** Subkeys in SHAvite-512 where '0' denotes an all-zero subkey $k^j_{0,i}$ or $k^j_{1,i}$, and '?' denotes a subkey which is not zero. The counter values are XORed prior to the subkeys marked by $^*$.

| $i$ | $RK_i$ | | | | $RK'_i$ | | | | $r$ |
|---|---|---|---|---|---|---|---|---|---|
| | $k^0_{0,i}$ | $k^1_{0,i}$ | $k^2_{0,i}$ | $k^3_{0,i}$ | $k^0_{1,i}$ | $k^1_{1,i}$ | $k^2_{1,i}$ | $k^3_{1,i}$ | |
| 0 | ? | ? | ? | ? | ? | ? | ? | ? | $M$ |
| 1 | ?$^*$ | ? | ? | ? | ? | ? | ? | 0 | 1 |
| 2 | 0 | ? | ? | ? | ? | 0 | 0 | 0 | |
| 3 | 0 | ? | ? | ? | 0 | 0 | 0 | 0 | 2 |
| 4 | 0 | ? | 0 | 0 | 0 | 0 | 0 | 0 | |
| 5 | 0 | 0$^*$ | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 9 | 0 | 0 | 0 | 0$^*$ | 0 | 0 | 0 | 0 | 5 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | ?$^*$ | ? | 7 |

### 4.3 Using Different Salt Values

Actually, we can relax the previous condition that all required subkeys are zero. This allows us to use many different salt values. Instead of trying to get the

**Table 6.** Subkeys in SHAvite-512 with periodic round keys $RK_i$ or $RK'_i$

| $i$ | $RK_i$ | | | | $RK'_i$ | | | |
|---|---|---|---|---|---|---|---|---|
| | $k^0_{0,i}$ | $k^1_{0,i}$ | $k^2_{0,i}$ | $k^3_{0,i}$ | $k^0_{1,i}$ | $k^1_{1,i}$ | $k^2_{1,i}$ | $k^3_{1,i}$ |
| $i$ | $a$ | $b$ | $c$ | $d$ | $a$ | $b$ | $c$ | $d$ |
| $i+1$ | $e$ | $f$ | $g$ | $h$ | $e$ | $f$ | $g$ | $h$ |
| $i+2$ | $a$ | $b$ | $c$ | $d$ | $a$ | $b$ | $c$ | $d$ |

subkeys to be zero, we try to find subkeys which are periodic (see Table 6). Due to the nature of the key schedule, we will then get $RK_{i+2} = RK'_{i+2} = RK_i = RK'_i$ as long as the counter does not interfere. This will be enough to apply the cancellation property, and to interleave it.

We can find salts and messages giving such a periodic expanded message by solving a linear system of equations. The message expansion alternates linear and non-linear layers, where the non-linear layer is one AES round with the salt used as the key. Note that the 512-bit salt is used twice in the 1024-bit message expansion. Hence, if we look for solutions with equal left and right halves, both halfs will still be equal after the non-linear layer. To find solutions, we construct a linear system with 1024 binary variables (or 32 32-bit variables), corresponding to the inputs and outputs of the non-linear layer. Hence, we get 1024 binary equations stating that the output of the linear layer must be equal to the input of the non-linear layer, such that the key-schedule will be periodic.

Each solution to this system gives an input and output of the non-linear layer. The resulting output and input of the single non-linear AES round can easily be connected by computing the according salt value. Then, we compute the message expansion backwards to get the message that will give a good expanded message for the given salt. Surprisingly, the system has many solutions, with a kernel of dimension 9, given by the following basis:

$$\begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&1&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&0&1&0&0&0&0&1
\end{bmatrix}$$

This means that we can generate $2^{9 \cdot 32} = 2^{288}$ different salt values and corresponding message blocks such that the conditions on the subkeys are fulfilled. An example of a random salt and message block in this class is given below:

```
Salt:   7f  51  e2  fb  ca  a5  95  ac  04  42  40  19  30  0f  17  82
        6d  31  01  30  86  30  0d  18  05  dc  db  90  96  2f  2a  78
        32  71  59  03  e3  bb  97  17  ee  41  bc  97  a3  b2  5c  18
        ce  af  fd  90  d6  8d  bf  fd  ab  11  9d  62  6a  11  13  b6
```

```
Message: e1 c1 44 35 67 84 1b 18 ca ad ac f8 13 4d ea c9
         b1 a0 79 d1 e7 a9 11 ca 75 eb 96 82 cc 81 50 4f
         7a 69 43 38 8f 5d e8 e4 e0 ce 3f 6b 55 1a 27 4e
         8d e6 2d 9a 63 76 78 73 67 10 8e d2 38 02 45 90
         12 16 0b cc 6f ab c8 1a ca 0e c8 b7 5b e7 33 93
         58 87 01 5f 09 b3 64 c3 a9 a2 5a 15 c9 70 a8 cb
         a0 80 ff a8 c7 6d 24 60 09 8e 9d 15 0b b1 af 8d
         5c 37 11 26 17 df d7 eb 9f bc f0 82 2d ad 98 e0
```

### 4.4 Collision and Preimages for 13 Rounds

If all the conditions on the key-schedule of Table 3 are fulfilled, the characteristic is followed and we get $B_{13} = Z$ after 13 rounds. For each value of $X$ and $Y$ ($Z$ is fixed in the attack), we can compute backward to get the input of the compression function according to Equation (11) (also see Table 4).

After applying the feed-forward to $B_0$, we get for the output word $H_1$:

$$H_1 = B_0 + B_{13} = Z + Y + F_3(Z) + F_1'(Z + F_2(X))$$

For any value of $H_1$ we can simply choose a value $X$ and compute $Y$. Hence, we can construct a partial preimage on 128 bits (on $H_1$) for the compression function of SHAvite-3-512 reduced to 13 rounds. The complexity is about one compression function evaluation. In Appendix A, we give 3 examples for inputs (chaining values) of the compression function of SHAvite-512 leading to outputs with $H_1$ equal to zero. Note that we can repeat this attack for all choices of $X$ and hence, $2^{128}$ times.

To construct a collision and preimage for the whole output of the compression function, we need to repeat the partial preimage algorithm $2^{192}$ and $2^{384}$ times, respectively. Note that we can construct up to $2^{128+288} = 2^{416}$ partial preimages using the freedom in $X$ and the message expansion.

### 4.5 Collision and Preimages for 14 Rounds

The 14-round attack follows the characteristic of Table 3. In order to extend the 13-round attack to 14 rounds, we simply add one round at the end. In this case, the conditions on the message, salt and counter value are the same as for 13 rounds. Again, we compute backwards to get the input of the compression function as a function of $X$, $Y$ and $Z$ (see Equation (12) and Table 4). Notice that after applying the feed-forward we get for the output word $H_2$:

$$H_2 = C_0 + C_{14} = F_2(X) + F_0'(X + F_1(Z + F_4(Y) + F_2'(Y + F_3(Z))))$$

which can be rewritten as:

$$F_0'^{-1}(H_2 + F_2(X)) + X = F_1(Z + F_4(Y) + F_2'(Y + F_3(Z))) \qquad (14)$$

This equation is nicely split using $X$ only on the left-hand side, and $Y$ only on the right-hand side. Any solution $X, Y$ to the equation gives a 128-bit partial

preimage for any chosen value of $H_2$. Note that we can also shift the characteristic by one round to get a partial preimage for $H_1$.

One easy way to find solutions for Equation (14) is to use a collision-finding algorithm: if we compute the left-hand side with $2^{64}$ random values for $X$, and the right-hand side with $2^{64}$ random values for $Y$, we expect to find one solution due to the birthday paradox with complexity $2^{64}$. Note that this can be done with a memoryless collision-finding algorithm. The complexity can be reduced at the cost of higher memory requirements. By first saving $2^{128}$ candidates for $X$ and then computing $2^{128}$ candidates for $Y$ we get $2^{128}$ solutions for Equation (14) with a complexity of $2^{128}$. Hence, we get an amortized cost of 1 computation per solution.

More generally, we can make a trade-off between time and memory using a distinguished point based collision-finding algorithm, as given in [17, Section 4.2]. Using $2^k$ bits of memory ($k \leq 128$) and $2^l$ processors, we can generate $2^{128}$ solutions choosing 128 bits of the output of the compression with complexity $2^{192-k/2-l}$. If we repeat this with several salts, we obtain the following attacks on the compression function:

- a collision attack in time $2^{256-k/2-l}$
- a preimage attack in time $2^{448-k/2-l}$

## 5 Attack on 10 Rounds of the Hash Function

The 10-round attack on the SHAvite-3-512 hash function is an extension of the 9-round attack from [5]. The extension is the same as the extension from 13 to 14 rounds of the compression function. Since the attack does not require freedom from the salt or counter values, it is a real attack on the 10-round SHAvite-3-512.

### 5.1 Extending the 9 Round Attack

We extend the 9-round attack by adding one round at the beginning according to the characteristic of Table 7. In order to satisfy the conditions in round 6 and 8, it is enough to have:

- $(k^1_{0,4}, k^2_{0,4}, k^3_{0,4}) = (k^1_{1,6}, k^2_{1,6}, k^3_{1,6})$ and $k^0_{0,4} + k^0_{1,6} = F_5(Z_5)$
- $(k^1_{0,6}, k^2_{0,6}, k^3_{0,6}) = (k^1_{1,8}, k^2_{1,8}, k^3_{1,8})$ and $k^0_{0,6} + k^0_{1,8} = F_7(Z_7)$

The condition on the keys $(k^1_{0,4}, k^2_{0,4}, k^3_{0,4}) = (k^1_{1,6}, k^2_{1,6}, k^3_{1,6})$ and $(k^1_{0,6}, k^2_{0,6}, k^3_{0,6}) = (k^1_{1,8}, k^2_{1,8}, k^3_{1,8})$ will be satisfied by carefully building a suitable message, following the algorithm given in [5]. Then, $Z_5$ and $Z_7$ are computed to satisfy the remaining conditions as follows: $Z_5 = F_5^{-1}(k^0_{0,4} + k^0_{1,6})$ and $Z_7 = F_7^{-1}(k^0_{0,6} + k^0_{1,8})$.

Then, we start with the state $B_2 = X$, $B_3 = Z_7$, $B_4 = Y$, and $B_5 = Z_5$ and compute backward to the input of the compression function, similar to Equation (12) and Table 4 of Section 4.1. In particular, we get for the input $C_0$:

$$C_0 = Z_7 + F_2(X) + F'_0(X + F_1(Z_5 + F_4(Y) + F'_2(Y + F_3(Z_7))))$$

**Table 7.** Characteristic for the attack on 10 of the hash function. We can fix the output $C_i = Z_5$ as long as the conditions are fulfilled.

| $i$ | $A_i$ | $B_i$ | $C_i$ | $D_i$ | *condition* |
|---|---|---|---|---|---|
| 0 | ? | ? | ? | ? | |
| 1 | ? | ? | ? | ? | |
| 2 | ? | $X$ | ? | ? | |
| 3 | ? | $Z_7$ | $X$ | ? | |
| 4 | ? | $Y$ | $Z_7$ | $D_4$ | |
| 5 | $D_4$ | $Z_5$ | $Y$ | $Z_7 + F'_4(D_4)$ | |
| 6 | $Z_7 + F'_4(D_4)$ | $D_4 + F_5(Z_5)$ | $Z_5$ | $D_6$ | $F_6(D_4 + F_5(Z_5)) = F'_4(D_4)$ |
| 7 | $D_6$ | $Z_7$ | ? | $Z_5 + F'_6(D_6)$ | |
| 8 | $Z_5 + F'_6(D_6)$ | $D_6 + F_7(Z_7)$ | $Z_7$ | ? | $F_8(D_6 + F_7(Z_7)) = F'_6(D_6)$ |
| 9 | ? | $Z_5$ | ? | ? | |
| 10 | ? | ? | $Z_5$ | ? | |

and after applying the feed-forward, we get for the output word $H_2$:

$$H_2 = C_0 + C_{10} = Z_5 + Z_7 + F_2(X) + F'_0(X + F_1(Z_5 + F_4(Y) + F'_2(Y + F_3(Z_7))))$$

Just like in the 14-round attack, we can rewrite this equation such that one side depends only on $X$, and one side depends only on $Y$:

$$F'^{-1}_0(H_2 + Z_5 + Z_7 + F_2(X)) + X = F_1(Z_5 + F_4(Y) + F'_2(Y + F_3(Z_7)))$$

To extend this partial preimage to a full preimage attack on the hash function we repeat the following steps:

- find a suitable message (cost: $2^{224}$);
- find about $2^{128}$ solutions $X$ and $Y$ using a memoryless collision-search algorithm (cost: $2^{196}$).

This generates $2^{128}$ inputs of the compression function such that 128 bits of the output are chosen. We need to repeat these steps $2^{256}$ times to get a full preimage on the compression function reduced to 10 rounds with a cost of $2^{480}$ and negligible memory requirements.

### 5.2 Second Preimage Attack

Using a first message block and by applying a generic unbalanced meet-in-the-middle attack, we can extend the preimage attack on the compression function to a second preimage attack on the hash function reduced to 10 rounds. The complexity is about $2^{497}$ compression function evaluations and $2^{16}$ memory. By using a tree based approach [7,12,13], the complexity of the attack can be reduced at the cost of higher memory requirements. Note that a preimage attack is not possible as we cannot ensure a correctly padded message block.

## 6 Conclusion

SHAvite-3-512 as considered during round 1 of the SHA-3 competition was shown to be subject to a chosen salt, chosen counter (pseudo) collision attack on the compression function. As a result, the compression function was tweaked by the designers. The tweaked SHAvite-3-512, as considered during round 2 of the SHA-3 competition, is here shown to still be succeptible to attacks in the same model, albeit at a higher cost. Although these attacks on the compression function do not imply an attack on the full hash function, they violate the collision resistance reduction proof of HAIFA. This illustrates that great care most be taken when salt and counter inputs are added to a compression function design. Furthermore, our analysis also illustrates that more than 70% (10 out of 14) of the rounds are not enough to defend the hash function SHAvite-3-512 against second preimage attacks.

## Acknowledgements

We would like to thank Hitachi, Ltd. for supporting a predecessor of this work.

## References

1. Andreeva, E., Bouillaguet, C., Fouque, P.A., Hoch, J.J., Kelsey, J., Shamir, A., Zimmer, S.: Second preimage attacks on dithered hash functions. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 270–288. Springer, Heidelberg (2008)
2. Biham, E., Dunkelman, O.: A Framework for Iterative Hash Functions - HAIFA. Cryptology ePrint Archive, Report 2007/278 (2007), http://eprint.iacr.org/2007/278 (Accessed on 10/1/2010)
3. Biham, E., Dunkelman, O.: The SHAvite-3 Hash Function. Submission to NIST (2008), http://ehash.iaik.tugraz.at/uploads/f/f5/Shavite.pdf (Accessed on 10/1/2010)
4. Biham, E., Dunkelman, O.: The SHAvite-3 Hash Function. Second round SHA-3 candidate (2009), http://ehash.iaik.tugraz.at/wiki/SHAvite-3 (Accessed on 10/1/2010)
5. Bouillaguet, C., Dunkelman, O., Leurent, G., Fouque, P.A.: Attacks on Hash Functions based on Generalized Feistel - Application to Reduced-Round Lesamnta and SHAvite-3-512. Cryptology ePrint Archive, Report 2009/634 (2009), http://eprint.iacr.org/2009/634 (Accessed on 10/1/2010)
6. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006), http://dx.doi.org/10.1007/11935230_1
7. De Cannière, C., Rechberger, C.: Preimages for Reduced SHA-0 and SHA-1. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 179–202. Springer, Heidelberg (2008)
8. Dean, R.D.: Formal Aspects of Mobile Code Security. Ph.D. thesis, Princeton University (1999)

9. Gauravaram, P., Knudsen, L.R.: On Randomizing Hash Functions to Strengthen the Security of Digital Signatures. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 88–105. Springer, Heidelberg (2009)
10. Halevi, S., Krawczyk, H.: Strengthening Digital Signatures Via Randomized Hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)
11. Kelsey, J., Schneier, B.: Second Preimages on $n$-bit Hash Functions for Much Less than $2^n$ Work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
12. Leurent, G.: MD4 is Not One-Way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
13. Mendel, F., Rijmen, V.: Weaknesses in the HAS-V Compression Function. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 335–345. Springer, Heidelberg (2007)
14. NIST: FIPS PUB 180-2-Secure Hash Standard (August 2002), http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf (Accessed on 10/1/2010)
15. NIST: Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Docket No: 070911510-7512-01 (November 2007)
16. NIST: Second Round Candidates. Official notification from NIST (2009), http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/submissions_rnd2.html (Accessed on 8/1/2010)
17. van Oorschot, P.C., Wiener, M.J.: Parallel Collision Search with Cryptanalytic Applications. J. Cryptology 12(1), 1–28 (1999)
18. Peyrin, T.: Chosen-salt, chosen-counter, pseudo-collision on SHAvite-3 compression function (2009), http://ehash.iaik.tugraz.at/uploads/e/ea/Peyrin-SHAvite-3.txt (Accessed on 10/1/2010)
19. Reyhanitabar, M.R., Susilo, W., Mu, Y.: Enhanced Security Notions for Dedicated-Key Hash Functions: Definitions and Relationships. In: Hong, S., Iwata, T. (eds.) FSE 2010, LNCS. Springer, Heidelberg (2010) (to appear)
20. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005), http://dx.doi.org/10.1007/11535218_2

## A Partial Preimage for 13 Rounds of the Compression of SHAvite-512

We give 3 examples for chaining inputs that all lead to a partial (128-bit) preimage of 0 for the compression function of SHAvite-512 reduced to 13 rounds. Note that the message block, the salt and counter values are equal in these examples.

```
counter:    00 00 00 00 FF FF FF FF 00 00 00 00 00 00 00 00
salt:       52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52
            52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52
            52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52
            52 52 52 52 52 52 52 52 52 52 52 52 52 52 52 52
```

```
message block M:  d1 58 6a 59 5f 1e ac c3 89 02 6a 23 8b 18 3d 35
                  a3 7b a6 8d 26 62 da 9a a6 8d 25 50 da 67 1e 62
                  0d fa 2b 8f a0 08 a4 97 b2 9b 25 0a 3e c3 6d c0
                  0b f7 12 3b d5 92 dd dc cf fa 79 ec 05 83 6e 9e
                  94 97 dd 03 4e e7 c1 07 8b f4 3d 9a df da 97 72
                  cc 24 50 90 0c 0a 0a b3 7c 58 d5 5d 7c 4d f9 ed
                  41 72 19 1a 8a ce 36 db ed fa 2e 40 23 66 8b d3
                  fa 1e 72 00 7b 8a 00 23 d3 00 49 88 00 96 79 19

chaining value 1: 73 ae 12 97 3f 8f 59 33 83 e5 b8 79 9f 39 3f d6
                  19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
                  3f e8 9e 31 8c 13 5b 51 05 f6 26 2f ab 50 d0 2f
                  7e d4 37 2c 7e b3 6f e2 a3 8c 10 c1 30 cb 43 1f
output 1:         f5 bb 28 52 27 67 80 b5 8d 68 2d 1b 66 f2 0c 1e
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                  94 49 61 0d cc ed ea 7b 89 2a 90 ee e4 cc 49 0c
                  9c 3e 2e 17 78 f2 60 44 f5 f9 95 6c c0 dd 70 4f
chaining value 2: 4d 96 cb 1f d7 26 9b f1 b8 84 e7 37 69 20 85 ee
                  19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
                  9c 0a 66 73 3f 9d 8e 4f 7d 15 85 71 6a cd fb 07
                  14 e6 c4 31 41 26 44 15 3a f8 a6 db b7 06 9a 4f
output 2:         2a bf 6d c0 ef f7 78 b2 29 88 60 cc 04 63 22 6d
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                  8b 45 69 a9 7f 67 5e 20 e4 8d 9b 01 d6 74 a9 dd
                  d3 9c 37 d1 ae ed 12 4d 47 d1 7c 28 72 26 1e 97
chaining value 3: b3 56 96 56 1a 43 91 1e 7b 0c 3f 99 9c f2 6b be
                  19 19 19 19 19 19 19 19 19 19 19 19 19 19 19 19
                  70 04 fd 88 dd b1 28 f2 03 6a 04 9c c2 65 b4 7b
                  2c d9 e6 74 aa 0b c5 78 85 e0 0c 21 89 ba 7f 8e
output 3:         31 a4 76 86 fa 16 f4 41 7a 93 6b 68 33 2d 46 c9
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                  b1 a3 15 94 bc 41 2d fc 69 83 82 13 76 76 17 92
                  af 7e d8 93 c5 06 13 8e 05 2b 31 ab 65 cd 2a 51
```

# Author Index