Atsuo Inomata
Kan Yasuda (Eds.)

LNCS 11049

Advances in Information and Computer Security

13th International Workshop on Security, IWSEC 2018
Sendai, Japan, September 3–5, 2018
Proceedings

Springer

Lecture Notes in Computer Science 11049

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

Editorial Board

More information about this series at http://www.springer.com/series/7410

Atsuo Inomata · Kan Yasuda (Eds.)

Advances in Information and Computer Security

13th International Workshop on Security, IWSEC 2018
Sendai, Japan, September 3–5, 2018
Proceedings

Springer

Editors
Atsuo Inomata
Tokyo Denki University
Tokyo
Japan

Kan Yasuda
Nippon Telegraph and Telephone Corporation
Tokyo
Japan

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-97915-1 ISBN 978-3-319-97916-8 (eBook)
https://doi.org/10.1007/978-3-319-97916-8

Library of Congress Control Number: 2018950103

LNCS Sublibrary: SL4 – Security and Cryptology

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The 13th International Workshop on Security, IWSEC 2018, was held at Sakura Hall of Tohoku University in Sendai, Japan, during September 3–5, 2018. The workshop was co-organized by ISEC (the Technical Committee on Information Security in Engineering Sciences Society of IEICE), CSEC (the Special Interest Group on Computer Security of IPSJ), and Cyberscience Center, Tohoku University.

This year, the workshop received 64 submissions, out of which two papers were withdrawn before the review process, and two other papers were withdrawn during the process. After extensive reviews and shepherding, we eventually accepted 18 regular papers and two short papers. Each submission was anonymously reviewed by four reviewers, and four (out of the 18) regular papers were accepted after revision under shepherding. These proceedings contain revised versions of the accepted papers.

The Best Paper Award was given to "Chosen Message Attack on Multivariate Signature ELSA at Asiacrypt 2017" by Yasufumi Hashimoto, Yasuhiko Ikematsu, and Tsuyoshi Takagi, and the Best Student Paper Award was given to "Estimated Cost for Solving Generalized Learning with Errors Problem via Embedding Techniques" by Weiyao Wang, Yuntao Wang, Atsushi Takayasu, and Tsuyoshi Takagi. In addition to the presentations of the accepted papers, the workshop also featured a poster session, SCIS/CSS invited talks, and two keynote talks. The keynote talks were given by Naofumi Homma and by Vasaka Visoottiviseth.

A number of people contributed to the success of IWSEC 2018. We would like to thank all authors for submitting their papers to the workshop, and we are also deeply grateful to the members of the Program Committee and to the external reviewers for their in-depth reviews and detailed discussions. We must mention that the selection of the papers was an extremely challenging task. Last but not least, we would like to thank the general co-chairs, Atsushi Fujioka and Masayuki Terada, for leading the Organizing Committee, and we would also like to thank the members of the Organizing Committee for ensuring the smooth running of the workshop.

June 2018

Atsuo Inomata
Kan Yasuda

IWSEC 2018
13th International Workshop on Security
Organization

Sendai, Japan, September 3–5, 2018

co-organized by
ISEC in ESS of IEICE
(Technical Committee on Information Security in Engineering Sciences Society of the Institute of Electronics, Information and Communication Engineers)
and
CSEC of IPSJ
(Special Interest Group on Computer Security of Information Processing Society of Japan)
and
Cyberscience Center, Tohoku University

General Co-chairs

Atsushi Fujioka	Kanagawa University, Japan
Masayuki Terada	NTT DOCOMO, Inc., Japan

Advisory Committee

Hideki Imai	University of Tokyo, Japan
Kwangjo Kim	Korea Advanced Institute of Science and Technology, South Korea
Christopher Kruegel	University of California, Santa Barbara, USA
Günter Müeller	University of Freiburg, Germany
Yuko Murayama	Tsuda College, Japan
Koji Nakao	National Institute of Information and Communications Technology, Japan
Eiji Okamoto	University of Tsukuba, Japan
C. Pandu Rangan	Indian National Academy of Engineering, India
Kai Rannenberg	Goethe University Frankfurt, Germany
Ryoichi Sasaki	Tokyo Denki University, Japan

Program Co-chairs

Atsuo Inomata	Tokyo Denki University, Japan
Kan Yasuda	Nippon Telegraph and Telephone Corporation, Japan

Organizing Committee

Hiroaki Anada	University of Nagasaki, Japan
Nuttapong Attrapadung	National Institute of Advanced Industrial Science and Technology, Japan
Keita Emura	National Institute of Information and Communications Technology, Japan
Takuya Hayashi	National Institute of Information and Communications Technology, Japan
Makoto Iguchi	Kii Corporation, Japan
Shuji Isobe	Tohoku University, Japan
Satoru Izumi	Tohoku University, Japan
Ryo Kikuchi	Nippon Telegraph and Telephone Corporation, Japan
Takaaki Mizuki	Tohoku University, Japan
Shiho Moriai	National Institute of Information and Communications Technology, Japan
Ken Naganuma	Hitachi, Ltd., Japan
Yoshitaka Nakamura	Future University Hakodate, Japan
Tetsushi Ohki	Shizuoka University, Japan
Yuji Suga	Internet Initiative Japan Inc., Japan
Nobuyuki Sugio	NTT DOCOMO, Inc., Japan
Keisuke Tanaka	Tokyo Institute of Technology, Japan
Yohei Watanabe	The University of Electro-Communications, Japan
Sven Wohlgemuth	Hitachi, Ltd., Japan
Takeshi Yagi	NTT Security (Japan) KK, Japan
Dai Yamamoto	Fujitsu Limited, Japan
Toshihiro Yamauchi	Okayama University, Japan

Program Committee

Mohamed Abid	University of Gabes, Tunisia
Mitsuaki Akiyama	Nippon Telegraph and Telephone Corporation, Japan
Nuttapong Attrapadung	National Institute of Advanced Industrial Science and Technology, Japan
Josep Balasch	KU Leuven, Belgium
Gregory Blanc	Telecom SudParis, France
Olivier Blazy	Université de Limoges, France
Yue Chen	Palo Alto Networks, USA
Celine Chevalier	Université Pantheon-Assas, France
Sabrina De Capitani di Vimercati	DI - Università degli Studi di Milano, Italy
Herve Debar	Telecom SudParis, France
Itai Dinur	Ben-Gurion University, Israel
Josep Domingo-Ferrer	Universitat Rovira i Virgili, Catalonia
Dawu Gu	Shanghai Jiao Tong University, China
Florian Hahn	SAP, Germany

Chung-Huang Yang	National Kaohsiung Normal University, Taiwan
Atsuo Inomata	Tokyo Denki University, Japan
Akira Kanaoka	Toho University, Japan
Yuichi Komano	Toshiba Corporation, Japan
Noboru Kunihiro	The University of Tokyo, Japan
Maryline Laurent	Telecom SudParis, France
Zhou Li	RSA Labs, USA
Atul Luykx	Visa Inc., USA
Frederic Majorczyk	DGA-MI/CentraleSupelec, France
Florian Mendel	Graz University of Technology, Austria
Bart Mennink	Radboud University, The Netherlands
Kirill Morozov	University of North Texas, USA
Ivica Nikolic	Nanyang Technological University, Singapore
Yin Minn Pa Pa	PwC Japan, Japan
Reza Reyhanitabar	KU Leuven, Belgium
Yusuke Sakai	National Institute of Advanced Industrial Science and Technology, Japan
Yu Sasaki	Nippon Telegraph and Telephone Corporation, Japan
Dominique Schroeder	Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
Yannick Seurin	Agence Nationale de la Securite des Systemes d'Information, France
Willy Susilo	University of Wollongong, Australia
Katsuyuki Takashima	Mitsubishi Electric Corporation, Japan
Mehdi Tibouchi	Nippon Telegraph and Telephone Corporation, Japan
Giorgos Vasiliadis	Qatar Computing Research Institute HBKU, Qatar
Sven Wohlgemuth	Hitachi, Ltd., Japan
Takeshi Yagi	NTT Security (Japan) KK, Japan
Kan Yasuda	Nippon Telegraph and Telephone Corporation, Japan
Rui Zhang	Chinese Academy of Sciences, China

Additional Reviewers

Anglès-Tafalla, Carles
Azaiez, Ikbel
Ben Amor, Arij
Benletaief, Nedra
Blanco-Justicia, Alberto
Cheng, Chen-Mou
de Saint Guilhem, Cyprien
Del Vasto, Luis
Deneuville, Jean-Christophe
Ding, Ning
Dobraunig, Christoph
Dramé-Maigné, Sophie
Eichlseder, Maria
Fech, Katharina
Gaborit, Philippe
Grujic, Milos
Hassan, Fadi
Hiromasa, Ryo
Jebri, Sarra
Kaaniche, Nesrine
Kakvi, Saqib A.
Kim, Jon-Lark

Lai, Russell W. F.
Lequesne, Matthieu
Long, Yu
Malavolta, Giulio
Matsuda, Takahiro
Matsuo, Kazuto
Morita, Hiraku
Ogata, Wakaha
Ricci, Sara
Ronge, Viktoria
Schläffer, Martin
Schuldt, Jacob
Shen, Yaobin
Shu, Jiang
Viguier, Benoît
Wouters, Lennert
Xu, Rui
Yamada, Shota
Yasuda, Takanori
Zhang, Chi

Contents

Cryptanalysis

Chosen Message Attack on Multivariate Signature ELSA at Asiacrypt 2017

Yasufumi Hashimoto[1], Yasuhiko Ikematsu[2(✉)], and Tsuyoshi Takagi[2]

[1] Department of Mathematical Science, University of the Ryukyus, Okinawa, Japan
hashimoto@math.u-ryukyu.ac.jp
[2] Department of Mathematical Informatics, University of Tokyo, Tokyo, Japan
{ikematsu,takagi}@mist.i.u-tokyo.ac.jp

Abstract. One of the most efficient post-quantum signature schemes is Rainbow whose hardness is based on the multivariate quadratic polynomial (MQ) problem. ELSA, a new multivariate signature scheme proposed at Asiacrypt 2017, has a similar construction to Rainbow. Its advantages, compared to Rainbow, are its smaller secret key and faster signature generation. In addition, its existential unforgeability against an adaptive chosen-message attack has been proven under the hardness of the MQ-problem induced by a public key of ELSA with a specific parameter set in the random oracle model. The high efficiency of ELSA is derived from a set of hidden quadratic equations used in the process of signature generation. However, the hidden quadratic equations yield a vulnerability. In fact, a piece of information of these equations can be recovered by using valid signatures and an equivalent secret key can be partially recovered from it. In this paper, we describe how to recover an equivalent secret key of ELSA by a chosen message attack. Our experiments show that we can recover an equivalent secret key for the claimed 128-bit security parameter of ELSA on a standard PC in 177 s with 1326 valid signatures.

Keywords: Post-quantum cryptography
Multivariate public-key cryptography · Chosen message attack
Rainbow · ELSA

1 Introduction

Shor [14] proposed quantum algorithms to factor large integers and to find discrete logarithms over a finite field of large order in polynomial time. This means that if large-scale quantum computers will be realized in the future, most currently used public key cryptosystems, such as RSA, DSA and ECC, will be insecure. The aim of Post-Quantum Cryptography (PQC) is to develop cryptosystems that are secure against attacks by future quantum computers [1]. At PQCrypto 2016, the National Institute of Standards and Technology (NIST) started the standardization process of post-quantum cryptography, and there are currently 69 proposals of post-quantum cryptography [10].

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 3–18, 2018.
https://doi.org/10.1007/978-3-319-97916-8_1

Multivariate public key cryptosystems (MPKCs) [5] are considered to be promising candidates for PQC. The early MPKCs are the Matsumoto-Imai scheme [9] and the Moon Letter scheme [15], and many MPKCs have been proposed until now. Among them, some schemes UOV [7] and HFEv$^-$ [11,12] have survived in this two decades, and seem efficient enough. Actually, several MPKCs are submitted to the NIST PQC standardization. Especially, Rainbow [6], a multi-layered version of the UOV scheme, has been remarked well because of its efficiency, modest computational cost, high security and simplicity.

The ELSA [13] signature scheme, studied in this paper, is a variant of Rainbow proposed at Asiacrypt 2017 by Shim et al. and is more efficient than Rainbow; that is, its secret key is smaller and its signature generation is faster. Shim et al. actually succeeded in reducing the complexity of signature generation from $O(n^3)$ for Rainbow to $O(n^2)$, where n is the number of variables used in a public key, without weakening the security against known attacks. The trick to reducing the complexity is choosing the secret keys sparsely and attaching several hidden quadratic equations in the process of signature generation (see Sect. 2.3). Furthermore, ELSA has an existential unforgeability against an adaptive chosen-message attack (EUF-CMA), which was proven under the hardness of the MQ problem induced by the public key of ELSA with a specific parameter set in the random oracle model. Note that EUF-CMA security for Rainbow was proven recently [4].

In this paper, we propose a chosen message attack on ELSA, under the condition that we can obtain valid signatures by repeatedly accessing a signing oracle. Recall that ELSA possesses hidden quadratic equations for accelerating the signature generation, that are not used in Rainbow. Once the hidden quadratic equations are recovered, an attacker can obtain an equivalent secret key of ELSA and forge a signature for an arbitrary message by its equivalent secret key. We show that a piece of information in the hidden quadratic equations can be recovered from at most n^2 valid signatures given by the chosen message attack. Our attack is very efficient, more precisely, its complexity is $O(n^{2\omega})$, where n is the number of variables and $2 \leq \omega < 3$ is the linear algebra constant. We implemented our attack on Magma [3], and succeeded in recovering an equivalent secret key with 1326 valid signatures in 177 s for the parameters selected in [13] as 128-bit security.

Our paper is organized as follows: in Sect. 2, we briefly summarize the ELSA scheme and its previous security analysis given in [13]. In Sect. 3, we discuss our new attack and give a detailed algorithm to obtain an equivalent secret key of ELSA. In Sect. 4, we preform the complexity analysis of our new attack and present a Magma implementation of our algorithm. We conclude our paper in Sect. 5.

2 The ELSA Signature Scheme

We briefly explain the basic concept of multivariate signature schemes and summarize the construction of the ELSA scheme and its previous security analysis following [13].

2.1 Multivariate Signature Scheme

Let $n, m \geq 1$ be integers, q a power of prime, and $\mathbb{F}_q$ a finite field of order q. In a multivariate signature scheme, the public key $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ is a quadratic map, namely $\mathcal{P}(x_1, \ldots, x_n) = {}^t(\mathcal{P}_1(x_1, \ldots, x_n), \ldots, \mathcal{P}_m(x_1, \ldots, x_n))$ given by

$$\mathcal{P}_l(x_1, \ldots, x_n) = \sum_{1 \leq i \leq j \leq n} \alpha_{ij}^{(l)} x_i x_j + \sum_{1 \leq i \leq n} \beta_i^{(l)} x_i + \gamma^{(l)}$$

for $1 \leq l \leq m$, where $\alpha_{ij}^{(l)}, \beta_i^{(l)}, \gamma^{(l)} \in \mathbb{F}_q$. For such a signature scheme, the public key $\mathcal{P}$ is generated by $\mathcal{P} = \mathcal{T} \circ \mathcal{F} \circ \mathcal{S}$ with invertible affine maps $\mathcal{T} : \mathbb{F}_q^m \to \mathbb{F}_q^m$, $\mathcal{S} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ and a quadratic map $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ that can be feasibly inverted. Thus the secret key consists of $\mathcal{T}, \mathcal{F}$ and $\mathcal{S}$.

To generate a signature of a message $\mathbf{m} \in \mathbb{F}_q^m$, one computes $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{m})$, finds $\mathbf{y}$ with $\mathcal{F}(\mathbf{y}) = \mathbf{z}$, and then computes $\mathbf{w} = \mathcal{S}^{-1}(\mathbf{y})$. A signature for $\mathbf{m}$ is given by $\mathbf{w}$. The verification involves checking whether $\mathcal{P}(\mathbf{w}) = \mathbf{m}$.

2.2 Key Generation of ELSA

We now describe the construction of ELSA [13].

Let l, k, u, r be positive integers and set $n = l + k + u + r$ and $m = k + u$. Denote the sets of l, k, u, r and n variables by

$$\begin{aligned}
&\mathbf{x}_L := (x_{L,1}, \ldots, x_{L,l}), \quad \mathbf{x}_K := (x_{K,1}, \ldots, x_{K,k}),\\
&\mathbf{x}_U := (x_{U,1}, \ldots, x_{U,u}), \quad \mathbf{x}_R := (x_{R,1}, \ldots, x_{R,r}),\\
&\mathbf{x} := {}^t(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, \mathbf{x}_R) = {}^t(x_{L,1}, \ldots, x_{R,r}).
\end{aligned}$$

First, we explain the construction of the central map of ELSA consisting of two layers. Let $L_i(\mathbf{x}) = L_i(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_R), R_{ij}(\mathbf{x}) = R_{ij}(\mathbf{x}_L, \mathbf{x}_K)$ $(1 \leq i \leq r, 1 \leq j \leq k)$ be linear polynomials and $\Phi_j(\mathbf{x}) = \Phi_j(\mathbf{x}_L)$ $(1 \leq j \leq k)$ quadratic polynomials. The first layer $(\mathcal{F}_1, \ldots, \mathcal{F}_k)$ of the central map of ELSA is

$$\mathcal{F}_j(\mathbf{x}) := \sum_{1 \leq i \leq r} L_i(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_R) R_{ij}(\mathbf{x}_L, \mathbf{x}_K) + \Phi_j(\mathbf{x}_L), \quad (1 \leq j \leq k).$$

To construct the second layer, let $R_{i,k+j}(\mathbf{x})$ $(1 \leq i \leq r, 1 \leq j \leq u)$, $L'_j(\mathbf{x}) = L'_j(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_R)$ $(1 \leq j \leq u)$ be linear polynomials and $\Phi_{k+j}(\mathbf{x}) = \Phi_{k+j}(\mathbf{x}_L, \mathbf{x}_K)$ $(1 \leq j \leq u)$ quadratic polynomials. The second layer $(\mathcal{F}_{k+1}, \ldots, \mathcal{F}_m)$ is

$$\mathcal{F}_{k+j}(\mathbf{x}) := \sum_{1 \leq i \leq r} L_i(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_R) R_{i,k+j}(\mathbf{x}) + \Phi_{k+j}(\mathbf{x}_L, \mathbf{x}_K) + L'_j(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_R),$$

for $1 \leq j \leq u$. The central map of ELSA is given by

$$\mathcal{F} := {}^t(\mathcal{F}_1, \ldots, \mathcal{F}_k, \mathcal{F}_{k+1}, \ldots, \mathcal{F}_m) : \mathbb{F}_q^n \to \mathbb{F}_q^n.$$

Next let us explain the secret and public keys of ELSA. Randomly choose two invertible affine maps $\mathcal{T} : \mathbb{F}_q^m \to \mathbb{F}_q^m$ and $\mathcal{S} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ and fix a linear polynomial

$L(\mathbf{x}) = L(\mathbf{x}_L)$ and r elements $\xi_1, \ldots, \xi_r \in \mathbb{F}_q^\times$ to generate a signature in ELSA efficiently (see Sect. 2.3).

Secret Key. The invertible affine maps $\mathcal{T}, \mathcal{S}$, the quadratic map $\mathcal{F}$, the linear polynomial L, and the constants $\xi_1, \ldots, \xi_r \in \mathbb{F}_q^\times$.

Public Key. The quadratic map $\mathcal{P} = {}^t(\mathcal{P}_1, \ldots, \mathcal{P}_m) := \mathcal{T} \circ \mathcal{F} \circ \mathcal{S} : \mathbb{F}_q^n \to \mathbb{F}_q^m$.

2.3 Signature Generation and Verification of ELSA

Here, we describe signature generation and signature verification of ELSA.

Signature Generation. For a message $\mathbf{m} \in \mathbb{F}_q^m$, compute $\mathbf{z} = {}^t(z_1, \ldots, z_m) = \mathcal{T}^{-1}(\mathbf{m})$. Next, find $\mathbf{y} \in \mathbb{F}_q^n$ with $\mathcal{F}(\mathbf{y}) = \mathbf{z}$ and $L(\mathbf{y})L_i(\mathbf{y}) = \xi_i$ $(1 \le i \le r)$. Finally, compute $\mathbf{w} = \mathcal{S}^{-1}(\mathbf{y}) \in \mathbb{F}_q^n$. The signature for $\mathbf{m}$ is $\mathbf{w}$.

Signature Verification. Check whether $\mathcal{P}(\mathbf{w}) = \mathbf{m}$ or not.

In the process of the signature generation, $\mathbf{y} \in \mathbb{F}_q^n$ is found as follows.

How to Find $\mathbf{y} \in \mathbb{F}_q^n$

First, randomly choose $\mathbf{y}_L \in \mathbb{F}_q^l$ with $L(\mathbf{y}_L) \neq 0$ and find a solution $\mathbf{y}_K$ to the system of k linear equations in $\mathbf{x}_K$:

$$\sum_{1 \le i \le r} \xi_i R_{ij}(\mathbf{y}_L, \mathbf{x}_K) = L(\mathbf{y}_L)(z_j - \Phi_j(\mathbf{y}_L)), \quad (1 \le j \le k). \tag{1}$$

Next, find a solution $\mathbf{y}_R$ to the system of r linear equations in $\mathbf{x}_R$:

$$L_i(\mathbf{y}_L, \mathbf{y}_K, \mathbf{x}_R) = L(\mathbf{y}_L)^{-1}\xi_i, \quad (1 \le i \le r). \tag{2}$$

Finally, find a solution $\mathbf{y}_U \in \mathbb{F}_q^u$ to the system of u linear equations in $\mathbf{x}_U$:

$$\sum_{1 \le i \le r} \xi_i R_{i,k+j}(\mathbf{y}_L, \mathbf{y}_K, \mathbf{x}_U, \mathbf{y}_R) = L(\mathbf{y}_L)(z_j - \Phi_{k+j}(\mathbf{y}_L, \mathbf{y}_K) - L'_j(\mathbf{y}_L, \mathbf{y}_K, \mathbf{y}_R)) \tag{3}$$

for $1 \le j \le u$. In this way, we find $\mathbf{y} = {}^t(\mathbf{y}_L, \mathbf{y}_K, \mathbf{y}_U, \mathbf{y}_R) \in \mathbb{F}_q^n$ such that $\mathcal{F}(\mathbf{y}) = \mathbf{z}$.

Note that Eqs. (1)–(3) are derived from

$$L(\mathbf{x}_L)\mathcal{F}_j(\mathbf{x}) = L(\mathbf{x}_L)z_j, \quad L(\mathbf{x}_L)L_i(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_R) = \xi_i. \tag{4}$$

Thus we see that $\mathbf{y}$ computed above satisfies $\mathcal{F}(\mathbf{y}) = \mathbf{z}$ and $L(\mathbf{y}_L)L_i(\mathbf{y}_L, \mathbf{y}_K, \mathbf{y}_R) = \xi_i$ for $i = 1, \ldots, r$. Since $\mathbf{w} = \mathcal{S}^{-1}(\mathbf{y}) \in \mathbb{F}_q^n$, the signature $\mathbf{w}$ obtained above satisfies

$$L(\mathcal{S}(\mathbf{w})) \cdot L_i(\mathcal{S}(\mathbf{w})) = \xi_i$$

for $i = 1, \ldots, r$. We also have $L(\mathcal{S}(\mathbf{w})) \neq 0$.

Equation (1) for $1 \le j \le k$ can be written as

$$\mathbf{x}_K A + \mathbf{c} = L(\mathbf{y}_L)\left[\mathbf{z}_K - \mathbf{b}(\mathbf{y}_L)\right], \tag{5}$$

where A is a $k \times k$ matrix over $\mathbb{F}_q$, $\mathbf{c}$ is an element of $\mathbb{F}_q^k$, $\mathbf{z}_K := (z_1, \ldots, z_k)$ and $\mathbf{b}(\mathbf{y}_L) = (\Phi_1(\mathbf{y}_L), \ldots, \Phi_k(\mathbf{y}_L)) \in \mathbb{F}_q^k$. Since the entries of A do not depend on $\mathbf{y}_K$, the process of finding $\mathbf{y}_K$ of (5) can be implemented as

$$\mathbf{y}_K = L(\mathbf{y}_L)\left[\mathbf{z}_K - \mathbf{b}(\mathbf{y}_L)\right] A_1 - \mathbf{c}A_1,$$

where $A_1 := A^{-1}$. This means that, if we have A_1 as a part of the secret key and l is small enough, $\mathbf{y}_K$ can be computed in $O(k^2) = O(n^2)$ time. We can easily check that Eqs. (2) and (3) are similar. Then, by choosing Φ_{k+j} sparsely as in [13], one can find $\mathbf{y}_R, \mathbf{y}_U$ in $O(n^2)$ time. As a result, the complexity $O(n^2)$ of the signature generation of ELSA is smaller than that the $O(n^3)$ complexity of Rainbow (see [13, Sect. 5]).

2.4 Previous Security Analysis and Parameter Selection

In this subsection, we give a short survey of the security analysis of ELSA discussed in [13] and recall the 128-bit security parameter based on that security analysis.

Direct Attack. The direct attack generates a dummy signature of a given message by directly solving a system of quadratic equations $\mathcal{P}(\mathbf{x}) = \mathbf{m}$. It is known that, if the polynomial system $\mathcal{P}(\mathbf{x}) - \mathbf{m}$ is semi-regular, the complexity of the hybrid method [2] between the Gröbner basis attack and the exhaustive attack is

$$\ll \min_{k \geq 0} q^k \cdot \left(m \binom{n-k+d_{reg}-1}{d_{reg}} \right)^w, \tag{6}$$

where d_{reg} is the degree of regularity given as the first non-positive coefficient of $(1-t)^m/(1-t)^{m-k}$, and $2 \leq w < 3$ is the linear algebra constant. In [13], the authors chose (6) with $w = 2$ as a lower bound of security against the direct attack.

Rainbow Band Separation (RBS). Let $\varphi : \mathbb{F}_q^n \to \mathbb{F}_q^n$ be the affine map such that $\varphi(\mathbf{x}) = (\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, L_1(\mathbf{x}), \ldots, L_r(\mathbf{x}))$ and put $\mathcal{F}' := \mathcal{F} \circ \varphi^{-1}$. A similar argument to the one in [13, Sect. 3.2] shows that the coefficient matrices $F_1', \ldots, F_m'$ of $\mathcal{F}_1'(\mathbf{x}), \ldots, \mathcal{F}_m'(\mathbf{x})$, i.e. $\mathcal{F}_j'(\mathbf{x}) = {}^t\mathbf{x}F_j'\mathbf{x} + (\text{linear polynomial})$, can be written as follows:

$$F_j' = \begin{pmatrix} *_l & 0 & 0 & * \\ 0 & 0_k & 0 & * \\ 0 & 0 & 0_u & 0 \\ * & * & 0 & 0_r \end{pmatrix} \ (1 \leq j \leq k), \quad \text{and} \quad F_j' = \begin{pmatrix} *_l & * & 0 & * \\ * & *_k & 0 & * \\ 0 & 0 & 0_u & * \\ * & * & * & *_r \end{pmatrix} \ (k+1 \leq j \leq m). \tag{7}$$

Due to these, we see that there exist vectors $\mathbf{t} = {}^t(t_1, \ldots, t_m) \in \mathbb{F}_q^{m-1}$ and $\mathbf{s} = {}^t(s_1, \ldots, s_n) \in \mathbb{F}_q^n$ such that

$$\sum_{1 \leq i \leq m} s_i \mathcal{P}_i \left(\begin{pmatrix} I_{n-1} & \mathbf{t} \\ 0 & 1 \end{pmatrix} \mathbf{x} \right) = {}^t\mathbf{x} \begin{pmatrix} *_{l+k+u-2} & 0 & * \\ 0 & 0_1 & 0 \\ * & 0 & *_{r+1} \end{pmatrix} \mathbf{x} + (\text{linear polyn.}).$$

Such $(\mathbf{t}, \mathbf{s})$ is part of an equivalent secret key. To recover $(\mathbf{t}, \mathbf{s})$, the attacker has to solve a system of cubic polynomial equations of $\mathbf{t}, \mathbf{s}$. Though it is not easy to estimate its complexity in general, the authors of [13] concluded that ELSA is secure enough against RBS attack under a suitable parameter selection.

Rank Attacks. Let $P_1, \ldots, P_m$ be the coefficient matrices of $\mathcal{P}_1(\mathbf{x}), \ldots, \mathcal{P}_m(\mathbf{x})$; that is, each P_i is the symmetric matrix of size n such that $\mathcal{P}_i(\mathbf{x}) = {}^t\mathbf{x}P_i\mathbf{x} +$ (linear polynomial in $\mathbf{x}$). The rank attack recovers an equivalent secret key by finding $\alpha_1, \ldots, \alpha_m \in \mathbb{F}_q$ such that the rank of $\alpha_1 P_1 + \cdots + \alpha_m P_m$ is small. By checking the coefficient matrices $F'_1, \ldots, F'_m$ of $\mathcal{F}'_1(\mathbf{x}), \ldots, \mathcal{F}'_m(\mathbf{x})$ given in (7) carefully, the authors of [13] estimated the complexities of the rank attacks as follows:

Min-Rank attack: $O(q^{\min\{l+k+1, l+2r-k+1, l+2r+1, 2l+k+1\}} \cdot (\text{polyn.}))$.

High-Rank attack: $O(q^u \cdot \frac{n^3}{6})$.

Kipnis-Shamir's (UOV) Attack. Kipnis and Shamir [8] proposed a polynomial time attack to recover an equivalent secret key of the oil and vinegar signature scheme, and Kipnis, Patarin and Goubin [7] generalized it to the unbalanced oil and vinegar signature scheme (UOV). It is known that this attack is also possible when the coefficient matrices are in the form $\begin{pmatrix} 0_o & * \\ * & *_v \end{pmatrix}$ and its complexity is $O(q^{\max\{v-o,0\}} \cdot (\text{polyn.}))$. The authors of [13] concluded that the complexity of Kipnis-Shamir's attack on ELSA is

$$O(q^{\min\{r-u, k+u, l+r, n-2u-1\}} \cdot (\text{polyn.}))$$

by studying the structure of the coefficient matrices $F'_1, \ldots, F'_m$ of $\mathcal{F}'_1(\mathbf{x})$, $\ldots, \mathcal{F}'_m(\mathbf{x})$ given in (7) and the process of this attack carefully.

The 128-bit Security Parameter Recommendation. Based on the security analyses above, the authors of [13] proposed the following 128-bit security parameter

$$\text{ELSA-128}: \quad (q, l, k, u, r, n, m) = (2^8, 6, 28, 15, 30, 79, 43).$$

See [13, Table 4] for a performance comparison with other signature schemes.

3 Our Attack on ELSA

In this section, we describe a chosen message attack on ELSA. Indeed, we show how to recover an equivalent secret key from the information associated with Eq. (4) by launching a chosen message attack. We also explain the construction of the equivalent secret key and a method for forging a signature from it.

3.1 Chosen Massage Attack

A chosen message attack is a standard security notion in signature schemes. Let $\mathcal{O}$ be a signing oracle which computes the signature $\mathbf{w} \in \mathbb{F}_q^n$ from a message $\mathbf{m} \in \mathbb{F}_q^m$ using the secret key of ELSA. The chosen message attack tries to generate a valid pair of a message $\mathbf{m}'$ and signature $\mathbf{w}'$ by repeatedly accessing the signing oracle $\mathcal{O}$, where $\mathcal{P}(\mathbf{w}') = \mathbf{m}'$ for the public key $\mathcal{P}$. The authors of ELSA [13] proved that ELSA is existentially unforgeable against the chosen message attack. However, we show that there is a way to recover an equivalent secret key by launching a chosen message attack. Recall that the signature generation of ELSA uses Eq. (4) in order to accelerate the signature generation. The reduced problem used in ELSA is different from that used in Rainbow, namely, ELSA has a special structure of using Eq. (4), which leaks the information related to the secret key. We propose an attack that recovers the information associated with Eq. (4) from the signatures $\mathbf{w}$ given in the chosen message attack.

In a weaker setting, the attacker is not allowed to choose the message $\mathbf{m}$ before asking the signing oracle, which is sometimes called the known message attack. We show that our attack is also feasible in this setting.

3.2 How to Recover the Information Associated with Eq. (4)

As shown in Sect. 2.3, we use the hidden quadratic equations $L(\mathbf{x})L_i(\mathbf{x}) = \xi_i$ in (4) to generate a signature in ELSA. In our attack, we recover the space

$$\mathcal{L}_{\mathcal{S}} := \mathrm{Span}_{\mathbb{F}_q} \{L_1(\mathcal{S}(\mathbf{x})), \ldots, L_r(\mathcal{S}(\mathbf{x}))\} \subset \mathbb{F}_q[\mathbf{x}] \tag{8}$$

from N valid signatures, where $N := \max\{n+1, \frac{1}{2}(n-r+2)(n-r+3)\}$.

Let $W \subset \mathbb{F}_q^n$ be the set of signatures generated by the ELSA scheme. From Sect. 2.3, it is clear that $\mathbf{w} \in W$ satisfies $L(\mathcal{S}(\mathbf{w})) \cdot L_i(\mathcal{S}(\mathbf{w})) = \xi_i$ for $1 \le i \le r$. Since $L(\mathcal{S}(\mathbf{w})) \neq 0$, we have

$$\xi_i L_j(\mathcal{S}(\mathbf{w})) - \xi_j L_i(\mathcal{S}(\mathbf{w})) = 0, \text{ for } 1 \le i, j \le r. \tag{9}$$

Let $L_{ij}(\mathbf{x}) := \xi_i L_j(\mathcal{S}(\mathbf{x})) - \xi_j L_i(\mathcal{S}(\mathbf{x}))$ and $\mathcal{L}_{\mathcal{S}}^0 := \mathrm{Span}_{\mathbb{F}_q}\{L_{ij}(\mathbf{x})\}_{i,j}$. It is easy to see that $\mathcal{L}_{\mathcal{S}}^0 \subset \mathcal{L}_{\mathcal{S}}$ and the set of linear forms $\{L_{12}(\mathbf{x}), \ldots, L_{1r}(\mathbf{x})\}$ is a basis of $\mathcal{L}_{\mathcal{S}}^0$, namely, $\dim_{\mathbb{F}_q} \mathcal{L}_{\mathcal{S}}^0 = r - 1$.

We first recover $\mathcal{L}_{\mathcal{S}}^0$. Due to (9), we can expect that

$$\mathcal{L}_{\mathcal{S}}^0 = \{f(\mathbf{x}) \in \mathbb{F}_q[\mathbf{x}] \mid \deg f \le 1, f(\mathbf{w}) = 0 \text{ for all } \mathbf{w} \in W\}. \tag{10}$$

Since the number of coefficients of a linear polynomial f in n variables is $n+1$, we can consider that $n+1$ valid signatures are equivalent to recovering $\mathcal{L}_{\mathcal{S}}^0$. In fact, we succeeded recovering $\mathcal{L}_{\mathcal{S}}^0$ from $n+1$ valid signatures in all of 100 experiments under the following parameters: Example-1 and Example-2 defined below, and ELSA-128 in Sect. 2.4.

$$\begin{aligned}
&\text{Example-1:} \quad (q, l, k, u, r, n, m) = (2^8, 4, 15, 5, 20, 44, 20),\\
&\text{Example-2:} \quad (q, l, k, u, r, n, m) = (2^8, 5, 20, 10, 25, 60, 30),\\
&\text{ELSA-128:} \quad (q, l, k, u, r, n, m) = (2^8, 6, 28, 15, 30, 79, 43).
\end{aligned}$$

Once the subspace $\mathcal{L}_{\mathcal{S}}^0$ is recovered, choose a basis $\{\mathcal{L}_1, \ldots, \mathcal{L}_{r-1}\}$ of $\mathcal{L}_{\mathcal{S}}^0$ and find an invertible affine map $\mathcal{S}_1 : \mathbb{F}_q^n \to \mathbb{F}_q^n$ such that

$$(\mathcal{L}_i \circ \mathcal{S}_1)(\mathbf{x}) = x_{R,i}, \quad (1 \leq i \leq r-1). \tag{11}$$

Since $\mathcal{L}_1(\mathbf{x}), \ldots, \mathcal{L}_{r-1}(\mathbf{x})$ are linear sum of $L_1(\mathcal{S}(\mathbf{x})), \ldots, L_r(\mathcal{S}(\mathbf{x}))$, we can easily check that for such $\mathcal{S}_1$, there exists an $r \times (l+k+u)$ matrix M of rank 1 such that

$$(\varphi \circ \mathcal{S} \circ \mathcal{S}_1)(\mathbf{x}) = \left(\begin{array}{c:c} *_{l+k+u} & * \\ \hdashline M & *_r \end{array}\right) \mathbf{x}.$$

Let $\mathbf{w} \in W$ and set $\mathbf{w}' := \mathcal{S}_1^{-1}(\mathbf{w}) \in \mathbb{F}_q^n$. Since $(\mathcal{L}_i \circ \mathcal{S}_1)(\mathbf{w}') = \mathcal{L}_i(\mathbf{w}) = 0$ by (9), the $x_{R,i}$-component of $\mathbf{w}'$ is zero for $1 \leq i \leq r-1$. Namely, we can write $\mathbf{w}' = (\mathbf{w}'_L, \mathbf{w}'_K, \mathbf{w}'_U, 0, \ldots, 0, w'_{R,r})$. Define the quadratic polynomial in variables $\mathbf{x}$:

$$\begin{aligned} Q(\mathbf{x}) := (L \circ \mathcal{S} \circ \mathcal{S}_1)(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, 0, \ldots, 0, x_{R,r}) \cdot \\ (L_1 \circ \mathcal{S} \circ \mathcal{S}_1)(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, 0, \ldots, 0, x_{R,r}) - \xi_1. \end{aligned}$$

More precisely, $Q(\mathbf{x})$ is a quadratic polynomial in $(n-r+1)$-variables $\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U$, $x_{R,r}$. Then we have

$$\begin{aligned} Q(\mathbf{w}') = \; & (L \circ \mathcal{S} \circ \mathcal{S}_1)(\mathbf{w}'_L, \mathbf{w}'_K, \mathbf{w}'_U, 0, \ldots, 0, w'_{R,r}) \cdot \\ & (L_1 \circ \mathcal{S} \circ \mathcal{S}_1)(\mathbf{w}'_L, \mathbf{w}'_K, \mathbf{w}'_U, 0, \ldots, 0, w'_{R,r}) - \xi_1 \\ = \; & (L \circ \mathcal{S} \circ \mathcal{S}_1)(\mathbf{w}') \cdot (L_1 \circ \mathcal{S} \circ \mathcal{S}_1)(\mathbf{w}') - \xi_1 \\ = \; & (L \circ \mathcal{S})(\mathbf{w}) \cdot (L_1 \circ \mathcal{S})(\mathbf{w}) - \xi_1 \\ = \; & 0. \end{aligned} \tag{12}$$

Therefore, $Q(\mathbf{x})$ vanishes at $\mathcal{S}_1^{-1}(\mathbf{w})$ for any $\mathbf{w} \in W$ and is a quadratic polynomial in $(n-r+1)$-variables $\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, x_{R,r}$. Note that $Q(\mathbf{x})$ is invariant up to a constant factor, even if we exchange L_1, ξ_1 in the definition of $Q(\mathbf{x})$ with other L_i, ξ_i. Thus, we can expect that

$$\mathbb{F}_q Q(\mathbf{x}) = \left\{ f \in \mathbb{F}_q[\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, x_{R,r}] \middle| \deg f \leq 2, f(\mathbf{w}') = 0 \text{ for all } \mathbf{w}' \in \mathcal{S}_1^{-1}(W) \right\}.$$

Here, $\mathbb{F}_q Q(\mathbf{x})$ is the vector space generated by $Q(\mathbf{x})$. Since the number of coefficients of a quadratic polynomial f in $(n-r+1)$-variables $\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, x_{R,r}$ is $N' := \frac{1}{2}(n-r+2)(n-r+3)$, we can consider that N' valid signatures are equivalent to recovering $\mathbb{F}_q Q(\mathbf{x})$. In fact, we succeeded recovering $\mathbb{F}_q Q(\mathbf{x})$ from N' valid signatures in all of 100 experiments under the same three parameters above.

Finally, we have the following:

Proposition 1. *Set* $N := \max\{n+1, \frac{1}{2}(n-r+2)(n-r+3)\}$. *The following subspaces (a) and (b) of* $\mathbb{F}_q[\mathbf{x}]$ *can be recovered from* N *valid signatures:*

$$(a)\ \mathcal{L}_{\mathcal{S}}, \qquad (b)\ \mathcal{L}_{\mathcal{S}}^0 + \mathbb{F}_q(L \circ \mathcal{S})(\mathbf{x}).$$

Proof. From the above arguments, $\mathcal{L}_S^0$ and $Q(\mathbf{x})$ (up to a constant factor) can be recovered from N valid signatures. By the definition of $Q(\mathbf{x})$, we can decompose

$$Q(\mathbf{x}) = D_1(\mathbf{x})D_2(\mathbf{x}) + c,$$

where $D_1(\mathbf{x}), D_2(\mathbf{x})$ are linear polynomials in $\mathbf{x}$ and $c \in \mathbb{F}_q$, and we have

$$\{D_1(\mathbf{x}), D_2(\mathbf{x})\} = \{(L\circ\mathcal{S}\circ\mathcal{S}_1)(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, x_{R,r}), (L_1\circ\mathcal{S}\circ\mathcal{S}_1)(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, x_{R,r})\}.$$

Since $\mathcal{L}_{\mathcal{S}} = \mathcal{L}_{\mathcal{S}}^0 + \mathbb{F}_q(L_1 \circ \mathcal{S})(\mathbf{x})$, either

$$\mathcal{L}_{\mathcal{S}}^0 + \mathbb{F}_q(D_1 \circ \mathcal{S}_1^{-1})(\mathbf{x}) \quad \text{or} \quad \mathcal{L}_S^0 + \mathbb{F}_q(D_2 \circ \mathcal{S}_1^{-1})(\mathbf{x})$$

is equal to $\mathcal{L}_{\mathcal{S}}$. Moreover, the other is equal to $\mathcal{L}_{\mathcal{S}}^0 + \mathbb{F}_q(L \circ \mathcal{S})(\mathbf{x})$. Therefore, we have

$$\left\{\mathcal{L}_{\mathcal{S}}^0 + \mathbb{F}_q(D_1 \circ \mathcal{S}_1^{-1})(\mathbf{x}), \mathcal{L}_S^0 + \mathbb{F}_q(D_2 \circ \mathcal{S}_1^{-1})(\mathbf{x})\right\} = \left\{\mathcal{L}_{\mathcal{S}}, \mathcal{L}_{\mathcal{S}}^0 + \mathbb{F}_q(L \circ \mathcal{S})(\mathbf{x})\right\}.$$

Thus, we can recover two subspaces (a) and (b) from N valid signatures. □

From Proposition 1, we have two subspaces, i.e., $\mathcal{L}_{\mathcal{S}}$ and $\mathcal{L}_{\mathcal{S}}^0 + \mathbb{F}_q(L \circ \mathcal{S})(\mathbf{x})$. At this stage, we cannot determine which one is $\mathcal{L}_{\mathcal{S}}$. However, it is not hard to construct an attack on ELSA.

3.3 Equivalent Secret Key of ELSA and Forging a Signature

We construct an equivalent secret key of ELSA by deforming the central map $\mathcal{F}$ as follows. Let $\varphi : \mathbb{F}_q^n \to \mathbb{F}_q^n$ be the invertible affine map such that

$$\varphi(\mathbf{x}) = {}^t\left(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, L_1(\mathbf{x}), \ldots, L_r(\mathbf{x})\right). \tag{13}$$

Put $\mathcal{F}' := \mathcal{F} \circ \varphi^{-1}$. Thus we have

$$\mathcal{P} = \mathcal{T} \circ \mathcal{F} \circ \mathcal{S} = \mathcal{T} \circ \mathcal{F}' \circ (\varphi \circ \mathcal{S}).$$

By a similar argument as [13, Sect. 3.2], it is easy to see that $\mathcal{F}'(\mathbf{x}) = {}^t(\mathcal{F}'_1(\mathbf{x}), \ldots, \mathcal{F}'_m(\mathbf{x}))$ can be written as

$$\begin{aligned}
\mathcal{F}'_j(\mathbf{x}) &= \sum_{1\le i\le r} x_{R,i} R'_{ij}(\mathbf{x}_L, \mathbf{x}_K) + \Phi'_j(\mathbf{x}_L) \\
&= {}^t\mathbf{x} \begin{pmatrix} *_l & 0 & 0 & * \\ 0 & 0_k & 0 & * \\ 0 & 0 & 0_u & 0 \\ * & * & 0 & 0_r \end{pmatrix} \mathbf{x} + (\text{linear polyn.}), \qquad (1 \le j \le k), \qquad (14) \\
\mathcal{F}'_j(\mathbf{x}) &= \sum_{1\le i\le r} x_{R,i} R'_{ij}(\mathbf{x}) + \Phi'_j(\mathbf{x}_L, \mathbf{x}_K) \\
&= {}^t\mathbf{x} \begin{pmatrix} *_l & * & 0 & * \\ * & *_k & 0 & * \\ 0 & 0 & 0_u & * \\ * & * & * & *_r \end{pmatrix} \mathbf{x} + (\text{linear polyn.}), \qquad (k+1 \le j \le m), \qquad (15)
\end{aligned}$$

for linear polynomials R'_{ij} and quadratic polynomials Φ'_j.

We define an equivalent secret key of ELSA:

Definition 1. *If two invertible affine maps* $\bar{\mathcal{T}} : \mathbb{F}_q^m \to \mathbb{F}_q^m$ *and* $\bar{\mathcal{S}} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ *satisfy the following conditions, then the pair* $(\bar{\mathcal{T}}, \bar{\mathcal{S}})$ *is called an equivalent secret key of the ELSA scheme.*

1. $\mathcal{P}' = {}^t(\mathcal{P}'_1, \ldots, \mathcal{P}'_m) := \bar{\mathcal{T}} \circ \mathcal{P} \circ \bar{\mathcal{S}}$.
2. *For* $1 \le j \le k$,
$$\mathcal{P}'_j(\mathbf{x}) = \mathcal{P}'_j(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_R) = \sum_{1 \le i \le r} x_{R,i} \cdot (\textit{linear polyn .in } \mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_R) + (\textit{quadratic polyn. in } \mathbf{x}_L, \mathbf{x}_R).$$
3. *For* $k+1 \le j \le m$,
$$\mathcal{P}'_j(\mathbf{x}) = \sum_{1 \le i \le r} x_{R,i} \cdot (\textit{linear polyn. in } \mathbf{x}) + (\textit{quadratic polyn. in } \mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_R).$$

From (14) and (15), it is enough to find a pair $(\bar{\mathcal{T}}, \bar{\mathcal{S}})$ such that

$$(\varphi \circ \mathcal{S} \circ \bar{\mathcal{S}})(\mathbf{x}) = \begin{pmatrix} *_l & 0 & 0 & * \\ * & *_k & 0 & * \\ * & * & *_u & * \\ 0 & 0 & 0 & *_r \end{pmatrix} \mathbf{x}, \quad (\bar{\mathcal{T}} \circ \mathcal{T})(\mathbf{y}) = \begin{pmatrix} *_k & 0 \\ * & *_u \end{pmatrix} \mathbf{y},$$

where $\mathbf{y} = {}^t(y_1, \ldots, y_m)$.

Once an equivalent secret key $(\bar{\mathcal{T}}, \bar{\mathcal{S}})$ is found, we can forge a signature $\mathbf{w}$ for each message $\mathbf{m} \in \mathbb{F}_q^m$ in the complexity $O(n^3)$. First compute

$$\bar{\mathbf{m}} = {}^t(\bar{m}_1, \ldots, \bar{m}_m) := \bar{\mathcal{T}}(\mathbf{m}).$$

Then, randomly choose $\mathbf{y}_L \in \mathbb{F}_q^l$ and $\mathbf{y}_R \in \mathbb{F}_q^r$. After that, find a solution $\mathbf{y}_K \in \mathbb{F}_q^k$ of the system of k linear equations in $\mathbf{x}_K$:

$$\bar{m}_j = \mathcal{P}'_j(\mathbf{y}_L, \mathbf{x}_K, \mathbf{y}_R), \quad (1 \le i \le k).$$

Next, find a solution $\mathbf{y}_U \in \mathbb{F}_q^u$ of the system of u linear equations in $\mathbf{x}_U$:

$$\bar{m}_j = \mathcal{P}'_j(\mathbf{y}_L, \mathbf{y}_K, \mathbf{x}_U, \mathbf{x}_R), \quad (k+1 \le j \le m)$$

and compute $\mathbf{w} = \bar{\mathcal{S}}(\mathbf{y}_L, \mathbf{y}_K, \mathbf{y}_U, \mathbf{y}_R)$, which is a signature of the message $\mathbf{m}$.

3.4 How to Recover an Equivalent Secret Key

In Sect. 3.2, we showed how to recover the space $\mathcal{L}_{\mathcal{S}}$ in (8) from N valid signatures. In this subsection, we explain how to recover an equivalent secret key of the ELSA scheme from the space $\mathcal{L}_{\mathcal{S}}$.

We assume the following:

(i) Choose either (a) or (b) in Proposition 1, and assume it is $\mathcal{L}_S$ in (8),
(ii) all linear and quadratic polynomials L_i, R_{ij}, Φ_j in Sect. 2.2 are homogeneous,
(iii) the linear polynomials $L'_j(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_R)$ $(1 \le j \le u)$ in Sect. 2.2 are zero, and
(iv) the secret key $\mathcal{T}, \mathcal{S}$ are linear maps.

Note that our attack described below can be easily modified without the assumptions (i), (ii), (iii) and (iv).

Choose a basis $\{\mathcal{L}_1, \ldots, \mathcal{L}_r\}$ of the space $\mathcal{L}_{\mathcal{S}}$ and an invertible linear map $\mathcal{S}' : \mathbb{F}_q^n \to \mathbb{F}_q^n$ such that

$$(\mathcal{L}_i \circ \mathcal{S}')(\mathbf{x}) = x_{R,i}, \quad (1 \le i \le r). \tag{16}$$

Since $\mathcal{L}_i(\mathbf{x})$ is a linear combination of $L_1(\mathcal{S}(\mathbf{x})), \ldots, L_r(\mathcal{S}(\mathbf{x}))$, we have

$$(\varphi \circ \mathcal{S} \circ \mathcal{S}')(\mathbf{x}) = \left(\begin{array}{cc:c} \begin{pmatrix} *_l & 0 \\ 0 & 0_k \end{pmatrix} & \begin{matrix} 0 & * \\ 0 & * \end{matrix} \\ \hdashline \begin{matrix} 0 & 0 \end{matrix} & \begin{matrix} 0_u & 0 \end{matrix} \\ \hdashline \begin{matrix} * & * \end{matrix} & \begin{matrix} 0 & *_r \end{matrix} \end{array}\right).$$

We now denote the matrix above in the right-hand-side by $\begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$ with matrices A, B, C of the sizes $(l+k+u) \times (l+k+u), (l+k+u) \times r, r \times r$, respectively. Due to (14) and (15), we can easily check that

$$\mathcal{P}' = {}^t(\mathcal{P}'_1, \ldots, \mathcal{P}'_m) := \mathcal{P} \circ \mathcal{S}' = \mathcal{T} \circ \mathcal{F}' \circ (\varphi \circ \mathcal{S} \circ \mathcal{S}')$$

is given by

$$\mathcal{P}'_j(\mathbf{x}) = {}^t\mathbf{x} \left(\begin{array}{c:c} {}^tA \begin{pmatrix} *_l & * & 0 \\ * & *_k & 0 \\ 0 & 0 & 0_u \end{pmatrix} A & \begin{matrix} * \\ * \\ * \end{matrix} \\ \hdashline \begin{matrix} * & * & * \end{matrix} & *_r \end{array}\right) \mathbf{x}, \quad (1 \le j \le m).$$

Thus, there exists an invertible $(l+k+u) \times (l+k+u)$ matrix S_2 such that

$$\mathcal{P}'_j(\begin{pmatrix} S_2 & 0 \\ 0 & I_r \end{pmatrix} \mathbf{x}) = {}^t\mathbf{x} \begin{pmatrix} *_l & * & 0 & * \\ * & *_k & 0 & * \\ 0 & 0 & 0_u & * \\ * & * & * & *_r \end{pmatrix} \mathbf{x}, \quad (1 \le j \le m) \tag{17}$$

and it holds $AS_2 = \begin{pmatrix} *_l & * & 0 \\ * & *_k & 0 \\ * & * & *_u \end{pmatrix}$. This means that the linear map $\mathcal{S}'' : \mathbb{F}_q^n \to \mathbb{F}_q^n$ defined by $\mathcal{S}''(\mathbf{x}) = \begin{pmatrix} S_2 & 0 \\ 0 & I_r \end{pmatrix} \mathbf{x}$ satisfies

$$(\varphi \circ \mathcal{S} \circ \mathcal{S}' \circ \mathcal{S}'')(\mathbf{x}) = \begin{pmatrix} AS_2 & B \\ 0 & C \end{pmatrix} \mathbf{x} = \begin{pmatrix} *_l & * & 0 & * \\ * & *_k & 0 & * \\ * & * & *_u & * \\ 0 & 0 & 0 & *_r \end{pmatrix} \mathbf{x}. \tag{18}$$

The following lemma follows immediately from (14), (15) and (18).

Lemma 1. *Set* $\tilde{\mathcal{S}} := \mathcal{S} \circ \mathcal{S}' \circ \mathcal{S}''$. *We obtain*

$$\mathcal{F}_j(\tilde{\mathcal{S}}(\mathbf{x})) = \mathcal{F}'_j(\varphi \circ \tilde{\mathcal{S}}(\mathbf{x})) = \begin{cases} {}^t\mathbf{x} \begin{pmatrix} *_l & * & 0 & * \\ * & *_k & 0 & * \\ 0 & 0 & 0_u & 0 \\ * & * & 0 & *_r \end{pmatrix} \mathbf{x}, & (1 \le j \le k) \\ {}^t\mathbf{x} \begin{pmatrix} *_l & * & 0 & * \\ * & *_k & 0 & * \\ 0 & 0 & 0_u & * \\ * & * & * & *_r \end{pmatrix} \mathbf{x}, & (k+1 \le j \le m). \end{cases}$$

From this lemma, it is clear that if $1 \le j \le k$, then the variables $\mathbf{x}_U$ do not appear in $\mathcal{F}'_j(\varphi \circ \tilde{\mathcal{S}}(\mathbf{x}))$. Also if $k+1 \le j \le m$, then $\mathbf{x}_U$ appear in $\mathcal{F}'_j(\varphi \circ \tilde{\mathcal{S}}(\mathbf{x}))$. From this fact, we have the following corollary:

Corollary 1. *Let* $\bar{\mathcal{T}} : \mathbb{F}_q^m \to \mathbb{F}_q^m$ *be an invertible linear map and define* $\mathcal{P}'' = (\mathcal{P}''_1, \ldots, \mathcal{P}''_m) := \bar{\mathcal{T}} \circ \mathcal{P} \circ (\mathcal{S}' \circ \mathcal{S}'') = (\bar{\mathcal{T}} \circ \mathcal{T}) \circ \mathcal{F}' \circ (\varphi \circ \tilde{\mathcal{S}})$. *If the variables* $\mathbf{x}_U$ *do not appear in* $\mathcal{P}''_j(\mathbf{x})$ *for any* $1 \le j \le k$, *then each* $\mathcal{P}''_j(\mathbf{x})$ *is a linear combination of* $\mathcal{F}'_1(\varphi \circ \tilde{\mathcal{S}}(\mathbf{x})), \ldots, \mathcal{F}'_k(\varphi \circ \tilde{\mathcal{S}}(\mathbf{x}))$. *Thus we have*

$$\bar{\mathcal{T}} \circ \mathcal{T}(\mathbf{y}) = \begin{pmatrix} *_k & 0 \\ * & *_u \end{pmatrix} \mathbf{y}.$$

In (18), we now write $\varphi \circ \tilde{\mathcal{S}}(\mathbf{x}) = (\varphi \circ \mathcal{S} \circ \mathcal{S}' \circ \mathcal{S}'')(\mathbf{x}) = \left(\begin{array}{c|c} A' & \begin{matrix} 0 & * \\ 0 & * \end{matrix} \\ \hline \begin{matrix} * & * \\ 0 & 0 \end{matrix} & \begin{matrix} *_u & * \\ 0 & *_r \end{matrix} \end{array}\right) \mathbf{x}$ with the matrix A' of the sizes $(l+k) \times (l+k)$. Since $\mathcal{P}''_j(\mathbf{x})$ $(1 \le j \le k)$ is a linear combination of $\mathcal{F}'_1(\varphi \circ \tilde{\mathcal{S}}(\mathbf{x})), \ldots, \mathcal{F}'_k(\varphi \circ \tilde{\mathcal{S}}(\mathbf{x}))$, from (14), we can easily check that

$$\mathcal{P}''_j(\mathbf{x}) = {}^t\mathbf{x} \left(\begin{array}{c|c} {}^tA' \begin{pmatrix} *_l & 0 \\ 0 & 0_k \end{pmatrix} A' & \begin{matrix} 0 & * \\ 0 & * \end{matrix} \\ \hline \begin{matrix} 0 & 0 \end{matrix} & \begin{matrix} 0_u & 0 \\ 0 & *_r \end{matrix} \\ \begin{matrix} * & * \end{matrix} & \end{array}\right) \mathbf{x}, \quad (1 \le j \le k).$$

Thus, there exists an invertible $(l+k) \times (l+k)$ matrix S_3 such that

$${}^tS_3 {}^tA' \begin{pmatrix} *_l & 0 \\ 0 & 0_k \end{pmatrix} A' S_3 = \begin{pmatrix} *_l & 0 \\ 0 & 0_k \end{pmatrix}$$

and it holds $A'S_3 = \begin{pmatrix} *_l & 0 \\ * & *_k \end{pmatrix}$. If we define the linear map $\mathcal{S}''' : \mathbb{F}_q^n \to \mathbb{F}_q^n$ by $\mathcal{S}'''(\mathbf{x}) = \begin{pmatrix} S_3 & 0 \\ 0 & I_{u+r} \end{pmatrix} \mathbf{x}$ and $\bar{\mathcal{S}} := \mathcal{S}' \circ \mathcal{S}'' \circ \mathcal{S}'''$, then we have

$$(\varphi \circ \mathcal{S} \circ \bar{\mathcal{S}})(\mathbf{x}) = \left(\begin{array}{cc:cc} & & 0 & * \\ & A' & 0 & * \\ \hdashline * & * & *_u & * \\ 0 & 0 & 0 & *_r \end{array}\right) \begin{pmatrix} S_3 & 0 \\ 0 & I_{u+r} \end{pmatrix} \mathbf{x}$$

$$= \left(\begin{array}{cc:cc} & & 0 & * \\ & A'S_3 & 0 & * \\ \hdashline * & * & *_u & * \\ 0 & 0 & 0 & *_r \end{array}\right) \mathbf{x} = \begin{pmatrix} *_l & 0 & 0 & * \\ * & *_k & 0 & * \\ * & * & *_u & * \\ 0 & 0 & 0 & *_r \end{pmatrix} \mathbf{x}.$$

From this and Corollary 1, the pair $(\bar{\mathcal{T}}, \bar{\mathcal{S}})$ satisfy Definition 1 in Sect. 3.3. Thus we recovered an equivalent secret key from the space $\mathcal{L}_{\mathcal{S}}$ in (8).

In Algorithm 1, we describe the detailed algorithm of our proposed attack in this section. Note that our attack needs $N = \max\{n, \frac{1}{2}(n-r+2)(n-r+3)\}$ valid signatures for ELSA with parameter (q, l, k, u, r, n, m).

4 Complexity Analysis and Experimental Results

This section analyzes the complexity of our attack on ELSA and describes an experiment on it.

4.1 Complexity Analysis for Our Proposed Attack

We will use Algorithm 1 to analyze the complexity of our attack described in Sect. 3,

Proposition 2. *The complexity of our proposed attack (Algorithm 1) is $O(n^{2\omega})$, where $2 \leq \omega < 3$ is the linear algebra constant.*

Proof. In Step 1, we solve a linear system of size $n+1$ to compute

$$\{f(\mathbf{x}) \in \mathbb{F}_q[\mathbf{x}] \mid \deg f \leq 1, f(\mathbf{w}_i) = 0,\ i = 1, \ldots, n+1\}.$$

This complexity is $O(n^\omega)$. Similarly, in Step 2, we solve a linear system of size N. Thus the complexity is $O(N^\omega) = O(n^{2\omega})$. In Step 3, we must compute the intersection of the kernel of $\tilde{P}_i$ $(1 \leq i \leq m)$ of size $l+k+u$. This complexity is $O((l+k+u)^{\omega+1}) = O(n^{\omega+1})$. In Step 4, we solve a linear system of size m $(< n)$. In Step 5, we compute the intersection of the kernel of $\tilde{P}'_i$ $(1 \leq i \leq m)$ of size $l+k$. This complexity is $O((l+k)^{\omega+1}) = O(n^{\omega+1})$. Therefore, the complexity of our attack is $O(n^{2\omega})$. □

Proposition 2 shows that our attack is efficient enough to break ELSA. As shown in the next subsection, we implemented our attack on Magma and succeeded in recovering an equivalent secret key efficiently. If we reset 128-bit security parameter for the ELSA scheme based on Proposition 2, then the number n of variables should be larger than 2^{20}, which is no longer practical.

Algorithm 1. The detailed algorithm of our proposed attack in §3

Input: The public key $\mathcal{P}(\mathbf{x}) = {}^t(\mathcal{P}_1(\mathbf{x}), \ldots, \mathcal{P}_m(\mathbf{x})) \in \mathbb{F}_q[\mathbf{x}]^m$ of ELSA with parameter (q, l, k, u, r, n, m) and N valid signatures $\mathbf{w}_1, \ldots, \mathbf{w}_N \in \mathbb{F}_q^n$, where $N := \max\left\{n+1, \frac{1}{2}(n-r+2)(n-r+3)\right\}$.

Output: An equivalent secret key $(\bar{\mathcal{T}}, \bar{\mathcal{S}})$ of Definition 1 in §3.3.

1: Compute a basis $\{\mathcal{L}_1(\mathbf{x}), \ldots, \mathcal{L}_{r-1}(\mathbf{x})\}$ of the $r-1$ dimensional vector space over $\mathbb{F}_q$:

$$\{f(\mathbf{x}) \in \mathbb{F}_q[\mathbf{x}] \mid \deg f \leq 1, f(\mathbf{w}_i) = 0,\ i = 1, \ldots, n+1\}.$$

Choose an invertible affine map $\mathcal{S}_1 : \mathbb{F}_q^n \to \mathbb{F}_q^n$ such that

$$(\mathcal{L}_i \circ \mathcal{S}_1)(\mathbf{x}) = x_{R,i},\ \ (1 \leq i \leq r-1).$$

2: Choose a non-zero polynomial $Q(\mathbf{x})$ of the one-dimensional vector space:

$$\left\{f \in \mathbb{F}_q[\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, x_{R,r}] \;\middle|\; \deg f \leq 2, f(\mathcal{S}_1^{-1}(\mathbf{w}_i)) = 0, 1 \leq i \leq N\right\}.$$

Decompose $Q(\mathbf{x})$ as follows:

$$Q(\mathbf{x}) = D_1(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, x_{R,r}) D_2(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, x_{R,r}) + c,$$

where D_1 and D_2 are linear polynomials in $\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U$, and $x_{R,r}$ and $c \in \mathbb{F}_q$. Set

$$D(\mathbf{x}) := D_1(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, x_{R,r}). \tag{19}$$

Choose an invertible affine map $\mathcal{S}' : \mathbb{F}_q^n \to \mathbb{F}_q^n$ such that

$$(\mathcal{L}_i \circ \mathcal{S}')(\mathbf{x}) = x_{R,i},\ \ (1 \leq i \leq r-1),\quad (D \circ \mathcal{S}_1^{-1} \circ \mathcal{S}')(\mathbf{x}) = x_{R,r}.$$

3: Compute the coefficient matrix $\tilde{P}'_j$ of size $l+k+u$ associated with the quadratic polynomial $(\mathcal{P}_j \circ S')(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U)$ for $1 \leq j \leq m$. Find an invertible matrix S_2 of size $l+k+u$ such that ${}^t S_2 \tilde{P}'_j S_2 = \begin{pmatrix} *_{l+k} & 0 \\ 0 & 0_u \end{pmatrix}$ for $1 \leq j \leq m$. If there is no such matrix, then return to Step 2 and reset (19)

$$D(\mathbf{x}) := D_2(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_U, x_{R,r}).$$

Let $\mathcal{S}'' : \mathbb{F}_q^n \to \mathbb{F}_q^n$ be the invertible linear map such that $\mathcal{S}''(\mathbf{x}) = \begin{pmatrix} S_2 & 0 \\ 0 & I_r \end{pmatrix} \mathbf{x}$.

4: Compute an invertible linear map $\bar{\mathcal{T}} : \mathbb{F}_q^m \to \mathbb{F}_q^m$ such that the variables $\mathbf{x}_U$ do not appear in $\mathcal{P}''_j(\mathbf{x})$ for any $1 \leq j \leq k$, where $\mathcal{P}'' = (\mathcal{P}''_1, \ldots, \mathcal{P}''_m) := \bar{\mathcal{T}} \circ \mathcal{P} \circ (\mathcal{S}' \circ \mathcal{S}'')$. Namely, $\mathcal{P}''_j(\mathbf{x}) = \mathcal{P}''_j(\mathbf{x}_L, \mathbf{x}_K, \mathbf{x}_R)$ for $1 \leq j \leq k$.

5: Compute the coefficient matrix $\tilde{P}''_j$ of size $l+k$ associated with $\mathcal{P}''_j(\mathbf{x}_L, \mathbf{x}_K)$ for $1 \leq j \leq k$. Find an invertible matrix S_3 of size $l+k$ such that for $1 \leq j \leq m$, ${}^t S_3 \tilde{P}''_j S_3 = \begin{pmatrix} *_l & 0 \\ 0 & 0_k \end{pmatrix}$. Let $\mathcal{S}''' : \mathbb{F}_q^n \to \mathbb{F}_q^n$ be the invertible linear map such that $\mathcal{S}'''(\mathbf{x}) = \begin{pmatrix} S_3 & 0 \\ 0 & I_{u+r} \end{pmatrix} \mathbf{x}$. Finally compute $\bar{\mathcal{S}} := \mathcal{S}' \circ \mathcal{S}'' \circ \mathcal{S}'''$.

4.2 Experimental Results of Our Proposed Attack

The experimental results of Algorithm 1 in Sect. 3 are presented in Table 1. The experiments were performed using Magma V2.21-6 [3] running on a 1.6 GHz Intel® Core™ i5 processor with 8 GB of memory.

We experimented with three different parameters: Example-1, Example-2, and ELSA-128. ELSA-128 is the 128-bit security parameter in Sect. 2.4. For each parameter, we measured the time taken to generate an equivalent secret key with our algorithm, and to forge a signature using the equivalent secret key. Table 1 presents the average timing of 100 experiments for each parameter. Here, $N := \max\{n+1, \frac{1}{2}(n-r+2)(n-r+3)\}$ is the number of valid signatures needed to recover an equivalent secret key of ELSA with parameter (q, l, k, u, r, n, m).

Table 1. Experimental results (second) of our attack against ELSA with parameter (q, l, k, u, r, n, m) and $N = \max\{n+1, \frac{1}{2}(n-r+2)(n-r+3)\}$ is the number of valid signatures.

Parameters	(q, l, k, u, r, n, m)	N	Algorithm 1	Forging a signature
Example-1	$(2^8, 4, 15, 5, 20, 44, 20)$	351	7.928	0.021
Example-2	$(2^8, 5, 20, 10, 25, 60, 30)$	703	40.19	0.069
ELSA-128	$(2^8, 6, 28, 15, 30, 79, 43)$	1326	176.68	0.101

For example, the number of valid signatures in ELSA-128 required in our attack is 1326. We succeeded in recovering an equivalent secret key in 176.68 s and forging a signature in 0.101 s.

5 Conclusion

We studied the security of ELSA [13], an efficient variant of Rainbow. ELSA uses special hidden quadratic equations to accelerate signature generation. However, such hidden quadratic equations weaken the security. In fact, we proved that such hidden quadratic equations can be recovered from sufficiently many valid signatures, and an equivalent secret key of ELSA can be obtained from the hidden quadratic equations. Our attack implemented on Magma with a standard personal computer succeeded in recovering an equivalent secret key in about 177 s with 1326 valid signatures for the claimed 128-bit security parameter of ELSA.

Finally, we stress that the original Rainbow has no hidden quadratic equations discussed in this paper. Our attack is thus unusable on Rainbow.

Acknowledgements. This work was supported by JST CREST (Grant Number JPMJCR14D6). The first author was also supported by JSPS Grant-in-Aid for Scientific Research (C) no. 17K05181.

References

1. Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.): Post-Quantum Cryptography. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88702-7
2. Bettale, L., Faugère, J.C., Perret, L.: Solving polynomial systems over finite fields: improved analysis of the hybrid approach. In: ISSAC 2012, pp. 67–74 (2012)
3. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symb. Comput. **24**, 235–265 (1997)
4. Ding, J., Chen, M.C., Petzoldt, A., Schmidt, D., Yang, B.Y.: Rainbow, NIST, Post-Quantum Cryptography Standardization, Round 1 Submissions. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions
5. Ding, J., Gower, J.E., Schmidt, D.S.: Multivariate Public Key Cryptosystems. Springer, Boston (2006). https://doi.org/10.1007/978-0-387-36946-4
6. Ding, J., Schmidt, D.: Rainbow, a new multivariable polynomial signature scheme. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 164–175. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_12
7. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 206–222. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_15
8. Kipnis, A., Shamir, A.: Cryptanalysis of the oil and vinegar signature scheme. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 257–266. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055733
9. Matsumoto, T., Imai, H.: Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In: Barstow, D., et al. (eds.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-45961-8_39
10. NIST, Post-Quantum Cryptography Standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/
11. Patarin, J., Courtois, N., Goubin, L.: QUARTZ, 128-bit long digital signatures. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 282–297. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_21
12. Petzoldt, A., Chen, M.-S., Yang, B.-Y., Tao, C., Ding, J.: Design principles for HFEv-based multivariate signature schemes. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 311–334. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_14
13. Shim, K.-A., Park, C.-M., Koo, N.: An existential unforgeable signature scheme based on multivariate quadratic equations. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 37–64. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_2
14. Shor, P.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)
15. Tsujii, S., Itoh, T., Fujioka, A., Kurosawa, K., Matsumoto, T.: A public-key cryptosystem based on the difficulty of solving a system of non-linear equations. Syst. Comput. Jpn. **19**(2), 10–18 (1988)

Key Recovery Attack on McNie Based on Low Rank Parity Check Codes and Its Reparation

Terry Shue Chien Lau(✉) and Chik How Tan

Temasek Laboratories, National University of Singapore,
5A Engineering Drive 1, #09-02, Singapore 117411, Singapore
{tsltlsc,tsltch}@nus.edu.sg

Abstract. Recently, Galvez et al. submitted McNie, a new public key encryption scheme to NIST as a candidate for the standard of post-quantum cryptography. They claimed that their parameters achieve 128-bit security with small key size by using Quasi-Cyclic Low Rank Parity Check codes (QC-LRPC) and block circulant matrices as the public key and secret key for the encryption. However, McNie based on QC-LRPC has several limitations in its design. In addition, Gaborit suggested an attack against the McNie encryption, which reduces the security levels of Galvez et al.'s proposals. Moreover, McNie based on LRPC codes has decryption failure. We propose a key recovery attack which recovers the secret key of their encryption of the claimed security level for all the proposed parameters. Even Galvez et al. revised their parameters against Gaborit's attack, we are still able to recover the secret key for the revised parameters by our key recovery attack. We propose a new McNie encryption based on Gabidulin codes with appropriate choices of secret key. Our McNie based on Gabidulin codes has error free decryption.

Keywords: Code-based cryptography · McEliece · Niederreiter
Key recovery attack · Public-key encryption

1 Introduction

In December of 2017, NIST published the Round 1 submissions for the Post-Quantum Cryptography. A new compact McEliece-Niederreiter cryptosystem called McNie was proposed by Galvez et al. [10]. Galvez et al. claimed that their proposal of McNie based on Quasi-Cyclic Low Rank Parity Check codes (QC-LRPC) provides smaller key sizes by employing Quasi-Cyclicity of matrices for 128-bit, 192-bit and 256-bit security level compared of those of RSA. For instance, they claimed that their proposal of McNie based on a 4-Quasi-Cyclic $[60, 30]$ LRPC code over $\mathbb{F}_{2^{37}}$ achieve 128-bit security with key size of 347 bytes.

There are two parts in McNie cryptosystem, the McEliece part which produces ciphertext $\boldsymbol{c}_1$, and the Niederreiter part which produces ciphertext $\boldsymbol{c}_2$. The sender first encrypts the message by multiplying the plaintext $\boldsymbol{m} \in \mathbb{F}_{q^m}^l$ with a

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 19–34, 2018.
https://doi.org/10.1007/978-3-319-97916-8_2

given $l \times n$ matrix G' and add a random error vector $\boldsymbol{e} \in \mathbb{F}_{q^m}^n$ of rank weight at most r, producing $\boldsymbol{c}_1 = \boldsymbol{m}G' + \boldsymbol{e}$. For the Niederreiter part, the sender computes a syndrome $\boldsymbol{c}_2 = \boldsymbol{m}F$, where F is a given $l \times (n-k)$ matrix and $l > n-k$. F and G' are related by $F = G'P^{-1}H^TS$ where P is a random $n \times n$ permutation matrix, H is an $(n-k) \times n$ parity check matrix for an r-error correcting code $\mathcal{C}$ with a known efficient decoding algorithm, and S is a random invertible $(n-k) \times (n-k)$ matrix over $\mathbb{F}_{q^m}$. Note that P, H and S are secret key used to compute $\boldsymbol{e}P^{-1}H^T = \boldsymbol{c}_1P^{-1}H^T - \boldsymbol{c}_2S^{-1}$. Since $(\boldsymbol{e}P^{-1})H^T$ is a syndrome for the code $\mathcal{C}$, $\boldsymbol{e}P^{-1}$ could be obtained by decoding $(\boldsymbol{e}P^{-1})H^T$, then we can recover $\boldsymbol{e}$ from $\boldsymbol{e}P^{-1}P$. Finally, we solve the linear system $\boldsymbol{m}G' = \boldsymbol{c}_1 - \boldsymbol{e}$ to recover $\boldsymbol{m}$.

To reduce the public key size, Galvez et al. suggested two proposals for the code which McNie is based on. Their first proposal, namely the 3-Quasi-Cyclic LRPC codes considered n to be a multiple of 3, P to be the identity matrix, G' in systematic 3-Quasi-Cyclic form (refer to Definition 4 for definition), H to be a parity check matrix for an LRPC code of weight d and F^T in systematic 2-Quasi-Cyclic form. Such setting requires 3 vectors of length $\frac{n}{3}$ each to generate G' and H, which gives a public key size of $nm\log_2(q)$ bits. Similarly, their second proposal, namely the 4-Quasi-Cyclic LRPC codes considered n to be a multiple of 4, P to be the identity matrix, G' in systematic 4-Quasi-Cyclic form, H to be a parity check matrix for an LRPC code of weight d and F^T in systematic 3-Quasi-Cyclic form. Such setting requires 5 vectors of length $\frac{n}{4}$ each to generate G' and H, which gives a public key size of $\frac{5}{4}nm\log_2(q)$ bits. In both proposals above, the matrices S and H are required to be block-circulant matrices (refer to Definition 4 for definition) for G' and F^T to be in systematic Quasi-Cyclic form. In addition, since LRPC decoding is probabilistic, therefore McNie based on LRPC has decryption failure.

Our Contribution. In this paper, we show that Galvez et al.'s proposal of McNie using QC-LRPC codes with secret key S and H being block-circulant matrices has several weaknesses in security. More precisely, we first include an attack by Gaborit [5] that reduces l, the number of unknowns in $\boldsymbol{m}$ into $l-(n-k)$, the number of unknowns in $\boldsymbol{m'}$. This attack reduces the general complexity of both combinatorial and algebraic attack on $\boldsymbol{c}_1$. More importantly, we propose a key recovery attack that helps us to recover the secret key S and H. We show that we are able to recover the secret key of the claimed security level for all the parameters suggested by Galvez et al. Even Galvez et al. revised their parameters against Gaborit's attack, we are still able to recover the secret key for the revised parameters by our key recovery attack. As a result, the parameters for McNie using QC-LRPC codes has to be modified to meet the required security. Finally, we propose a new McNie encryption based on Gabidulin codes with appropriate secret key P and S. Our proposal overcomes the limitations of McNie based on LRPC codes, and has error free decryption.

Organization of the Paper. The rest of the paper is organized as follows: we first review in Sect. 2 some preliminaries for rank metric, circulant matrix and block-circulant matrices. We also introduce the hard problems which our reparation of McNie is based on, and give the known generic attacks on the problem.

In Sect. 3 we describe briefly the McNie public-key encryption scheme based on QC-LRPC codes proposed by Galvez et al., and discuss some of its limitations. In Sect. 4, we include the Gaborit's attack [5]. In Sect. 5, we propose our key recovery attack on the McNie based on QC-LRPC codes by recovering the secret key, and show that all the parameters suggested do not achieve the claimed security. In Sect. 6, we propose a new McNie encryption based on Gabidulin codes with an appropriate choice of P and S. We also show that our proposal is `IND-CPA` secure under Decisional Rank Syndrome Decoding assumption. We provide some parameters for the proposal based on the Gabidulin codes. Finally, we give our final considerations of this paper in Sect. 7.

2 Preliminaries

In this section we recall the definition of rank metric for rank code. We also introduce the hard problems in coding theory - Decisional Rank Syndrome Decoding problem which McNie encryption is based on and give the existing generic attacks on the Rank Syndrome Decoding problem.

2.1 Background

Let $\mathbb{F}_{q^m}$ be a finite field with q^m elements and let $\{\beta_1, \ldots, \beta_m\}$ be a basis of $\mathbb{F}_{q^m}$ over the base field $\mathbb{F}_q$.

Definition 1. A *linear code* of length n and dimension k is a linear subspace $\mathcal{C}$ of the vector space $\mathbb{F}_{q^m}^n$.

Given a matrix M over a field $\mathbf{F}$, the rank of M, $\mathrm{rk}(M)$ is the dimension of the row span of M as a vector space over $\mathbf{F}$. The row span of a matrix M over $\mathbf{F}$ is denoted as $\langle M \rangle_{\mathbf{F}}$. We define the rank metric of a vector on $\mathbb{F}_{q^m}^n$:

Definition 2. Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$ and $M \in \mathbb{F}_{q^m}^{k \times n}$. The *rank* of $\boldsymbol{x}$ in $\mathbb{F}_q$, denoted by $\mathrm{rk}_q(\boldsymbol{x})$ is the rank of the matrix $X = (x_{ij}) \in \mathbb{F}_q^{m \times n}$ where $x_j = \sum_{i=1}^{m} x_{ij}\beta_i$. The *column rank* of M over $\mathbb{F}_q$, denoted by $\mathrm{colrk}_q(M)$ is the maximum number of columns that are linearly independent over $\mathbb{F}_q$.

We now define circulant matrix and k-partial circulant matrix induced by $\boldsymbol{x}$:

Definition 3. Let $\boldsymbol{x} = (x_0, \ldots, x_{n-1}) \in \mathbb{F}_{q^m}^n$. The *circulant matrix*, $\mathrm{Cir}_n(\boldsymbol{x})$ induced by $\boldsymbol{x}$ is defined as $\mathrm{Cir}_n(\boldsymbol{x}) = \left(x_{(i-j) \bmod n}\right) \in \mathbb{F}_{q^m}^{n \times n}$. The *k-partial circulant matrix*, $\mathrm{Cir}_k(\boldsymbol{x})$ induced by $\boldsymbol{x}$ is the first k rows of $\mathrm{Cir}_n(\boldsymbol{x})$.

Definition 4. An $[m'n, s'n]$ *block-circulant* matrix $M \in \mathbb{F}_{q^m}^{m'n \times s'n}$ is a matrix of the form $M = \begin{pmatrix} M_{11} & \ldots & M_{1s'} \\ \vdots & \ddots & \vdots \\ M_{m'1} & \ldots & M_{m's'} \end{pmatrix}$ where each M_{ij} is an $n \times n$ circulant matrix for $1 \leq i \leq m'$, $1 \leq j \leq s'$. A *systematic s-Quasi-Cyclic matrix* $M_{sys} \in$

$\mathbb{F}_{q^m}^{(s-1)n \times sn}$ is a matrix of the form $M_{sys} = \begin{pmatrix} I_n & & \mathbf{0} & M_1 \\ & \ddots & & \vdots \\ \mathbf{0} & & I_n & M_{s-1} \end{pmatrix}$ where each M_i is an $n \times n$ circulant matrix for $1 \leq i \leq s-1$.

2.2 Hard Problems in Coding Theory

We describe the hard problems which McNie is based on.

Definition 5. Rank Syndrome Decoding Problem (RSD). Let H be a full rank $(n-k) \times n$ matrix over $\mathbb{F}_{q^m}$, $\boldsymbol{s} \in \mathbb{F}_{q^m}^{n-k}$ and w an integer. The *Rank Syndrome Decoding Problem* $\mathsf{RSD}(q, m, n, k, w)$ needs to determine $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ such that $\mathrm{rk}_q(\boldsymbol{x}) = w$ and $H\boldsymbol{x}^T = \boldsymbol{s}^T$.

Recently, the RSD problem has been proven to be hard with a probabilistic reduction to the Hamming setting [9].

Given $G \in \mathbb{F}_{q^m}^{k \times n}$ a full rank parity check matrix of H in an RSD problem and $\boldsymbol{y} \in \mathbb{F}_{q^m}^n$. Then the dual version of $\mathsf{RSD}(q, m, n, k, w)$ is to determine $\boldsymbol{m} \in \mathbb{F}_{q^m}^k$ and $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ such that $\mathrm{rk}_q(\boldsymbol{x}) = w$ and $\boldsymbol{y} = \boldsymbol{m}G + \boldsymbol{x}$.

Notation. If X is a finite set, we write $\boldsymbol{x} \xleftarrow{\$} X$ to denote assignment to $\boldsymbol{x}$ of an element sampled from the uniform distribution on X.

We now give the definition of Decisional version of RSD problem in its dual form.

Definition 6. Decisional RSD Problem (DRSD). Let G be a full rank $k \times n$ matrix over F_{q^m}, $\boldsymbol{m} \in \mathbb{F}_{q^m}^k$ and $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ of rank r. The *Decisional RSD Problem* $\mathsf{DRSD}(q, m, n, k, w)$ needs to distinguish the pair $(\boldsymbol{m}G + \boldsymbol{x}, G)$ from $(\boldsymbol{y}, G)$ where $\boldsymbol{y} \xleftarrow{\$} \mathbb{F}_{q^m}^n$.

It is shown that DRSD is hard in the worst case [6]. Therefore, DRSD is eligible to be a candidate of hard problems in coding theory.

2.3 Generic Attacks on RSD

Combinatorial Attack. The combinatorial approach depends on counting the number of possible supports of size r for a rank code of length n over $\mathbb{F}_{q^m}$, which corresponds to the number of subspaces of dimension r in $\mathbb{F}_{q^m}$. The complexities of the best combinatorial attacks proposed in [1,7,15] is lower bounded by

$$\left\{(n-k)^3 m^3 q^{\min\left\{r\frac{(k+1)m}{n} - m, (r-1)k\right\}}, r^3 m^3 q^{(r-1)(k+1)}, (k+r)^3 r^3 q^{(m-r)(r-1)}\right\}.$$

Algebraic Attack. The nature of the rank metric favors algebraic attacks using Gröbner bases and became efficient when q increases. There are mainly three approaches to translate the notion of rank into algebraic setting: considering directly the RSD problem [13]; reducing RSD problem into MinRank [2]; using linearized q-polynomials [7]. The complexities of the best algebraic attacks proposed in [7,12] is lower bounded by $\left\{k^3 m^3 q^{r\left\lceil \frac{km}{n} \right\rceil}, r^3 k^3 q^{r\left\lceil \frac{(r+1)(k+1)-(n+1)}{r} \right\rceil}\right\}$.

3 McNie Public-Key Encryption Scheme

In this section, we recall the general algorithm specification of the McNie public-key encryption proposed by Galvez et al. [10]. Furthermore, we include their proposed encryption using 3-Quasi-Cyclic and 4-Quasi-Cyclic LRPC codes.

3.1 McNie Encryption

Rank Metric Public Key Encryption. A public-key encryption scheme $\mathtt{PE} = (\mathcal{S}_{\mathtt{PE}}, \mathcal{K}_{\mathtt{PE}}, \mathcal{E}_{\mathtt{PE}}, \mathcal{D}_{\mathtt{PE}})$ consists of four algorithms, where $\mathcal{S}_{\mathtt{PE}}$ is a set up algorithm, $\mathcal{K}_{\mathtt{PE}}$ is a key generation algorithm, $\mathcal{E}_{\mathtt{PE}}$ is an encryption algorithm and $\mathcal{D}_{\mathtt{PE}}$ is a decryption algorithm.

We now present the original McNie public-key encryption scheme [10].

Setup, $\mathcal{S}_{\mathtt{PE}}$ Generates global parameters m, n, l, k, r such that $l > n - k$. The plaintext space is $\mathbb{F}_{q^m}^l$. Outputs parameters $= (m, n, l, k, r)$.

Key Generation, $\mathcal{K}_{\mathtt{PE}}$ Generates $S \xleftarrow{\$} \mathrm{GL}_k(\mathbb{F}_{q^m})$, an $(n-k) \times n$ parity check matrix H for an r-error correcting code $\mathcal{C}$ over $\mathbb{F}_{q^m}$ with a known efficient decoding algorithm $\mathcal{C}.\mathfrak{Dec}(\cdot)$, a random $n \times n$ permutation matrix P, a random matrix $G' \xleftarrow{\$} \mathbb{F}_{q^m}^{l \times n}$ with dimension l. Outputs public key $\kappa_{pub} = \left(F = G'P^{-1}H^T S, G'\right)$ and secret key $\kappa_{sec} = (S, H, P)$.

Encryption, $\mathcal{E}_{\mathtt{PE}}(\kappa_{pub} = (F, G'), \boldsymbol{m})$ Let $\boldsymbol{m} \in \mathbb{F}_{q^m}^l$ be the message to be encrypted. Generates $\boldsymbol{e} \xleftarrow{\$} \mathbb{F}_{q^m}^n$ such that $\mathrm{rk}_q(\boldsymbol{e}) \leq r$. Computes $\boldsymbol{c}_1 = \boldsymbol{m}G' + \boldsymbol{e}$ and $\boldsymbol{c}_2 = \boldsymbol{m}F$. Outputs $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$ as the ciphertext.

Decryption, $\mathcal{D}_{\mathtt{PE}}(\kappa_{sec} = (S, H, P), \boldsymbol{c})$ Computes $\boldsymbol{c}_1 P^{-1} H^T - \boldsymbol{c}_2 S^{-1} = (\boldsymbol{m}G' + \boldsymbol{e})P^{-1}H^T - (\boldsymbol{m}G'P^{-1}H^T S)S^{-1} = \boldsymbol{e}P^{-1}H^T$. Since $\mathrm{rk}_q\left(\boldsymbol{e}P^{-1}\right) = \mathrm{rk}_q\left(\boldsymbol{e}\right) \leq r$, then the algorithm $\mathcal{C}.\mathfrak{Dec}(\cdot)$ can decode correctly and retrieve $\boldsymbol{e}P^{-1}$. Multiply $\left(\boldsymbol{e}P^{-1}\right)P$ to get $\boldsymbol{e}$. Finally, obtain $\boldsymbol{m}$ by solving $\boldsymbol{m}G' = \boldsymbol{c}_1 - \boldsymbol{e}$.

3.2 McNie Based on LRPC Codes

We now give the definitions of LRPC codes, Quasi-Cyclic codes and Quasi-Cyclic LRPC codes.

Definition 7. LRPC Codes. An $[n, k, d]$-Low Rank Parity Check (LRPC) code of rank d, length n and dimension k over $\mathbb{F}_{q^m}$ is a code such that the code has for parity check matrix, an $(n-k) \times n$ matrix $H = (h_{ij})$ such that the coefficients h_{ij} generate a vector subspace, V of $\mathbb{F}_{q^m}$ with dimension at most d. We call this dimension the weight of H. We denote one of V's bases by $\{F_1, \ldots, F_d\}$.

Probabilistic Decoding. Let the error vector $\boldsymbol{e}$ belongs to a vector space E of dimension r, and the syndrome $\boldsymbol{s} = (s_1, \ldots, s_{n-k})$ with syndrome space $S = \langle s_1, \ldots, s_{n-k} \rangle_{\mathbb{F}_q}$. The decoding of LRPC codes has a failure probability $q^{-(n-k+1-rd)}$. Moreover, there is a nonzero probability that $\dim(\cap S_i) \neq r$ where $S_i = F_i^{-1} S = \langle F_i^{-1} s_1, \ldots, F_i^{-1} s_{n-k} \rangle_{\mathbb{F}_q}$. Therefore, the decoding algorithm of LRPC codes is probabilistic.

Definition 8. Quasi-Cyclic Codes. An $[n, k]$ linear code is an $[n, k]$-*Quasi-Cyclic* code if there is some integer n_0 such that every cyclic shift of a codeword by n_0 places is again a codeword.

When $n = n_0 p$ for some integer p, it is possible to have both the generator and parity check matrices composed by $p \times p$ circulant blocks.

Definition 9. Quasi-Cyclic LRPC. An $[n, k, d]$-*Quasi-Cyclic Low Rank Parity Check* (QC LRPC) code of rank d, is an $[n, k]$-Quasi-Cyclic code which has for parity check matrix, an $(n-k) \times n$ matrix $H = (h_{ij})$ such that the coefficients h_{ij} generate a vector subspace, V of $\mathbb{F}_{q^m}$ with dimension at most d.

McNie Based on 3-Quasi-Cyclic LRPC. n has to be a multiple of 3 and $l = k = \frac{2n}{3}$. The secret key H is a parity check matrix for an $[n, \frac{2n}{3}]$-QC LRPC code of weight d, where H is an $\left[\frac{n}{3}, 3\left(\frac{n}{3}\right)\right]$ block-circulant matrix, and S is an $\frac{n}{3} \times \frac{n}{3}$ circulant matrix. The public key G' is a systematic $[2n, 3n]$ block-circulant matrix and F^T is a systematic $[n, 2n]$ block-circulant matrix.

McNie Based on 4-Quasi-Cyclic LRPC. n has to be a multiple of 4, $l = \frac{3n}{4}$ and $k = \frac{2n}{4}$. The secret key H is a parity check matrix for an $[n, \frac{2n}{4}]$-QC LRPC code of weight d, where H is a $\left[2\left(\frac{n}{4}\right), 4\left(\frac{n}{4}\right)\right]$ block-circulant matrix, and S is a $\left[2\left(\frac{n}{4}\right), 2\left(\frac{n}{4}\right)\right]$ block-circulant matrix. The public key G' is a systematic $[3n, 4n]$ block-circulant matrix and F^T is a systematic $[2n, 3n]$ block-circulant matrix.

Some Limitations of McNie Based on LRPC. Since LRPC codes has probabilistic decoding, McNie based on LRPC is a non-deterministic encryption scheme, i.e., with decryption failure. Furthermore, if the plaintext $\boldsymbol{m}$ for encryption has rank t, then solving $\boldsymbol{c}_2 = \boldsymbol{m}F$ is equivalent to solving an $\mathsf{RSD}(q, m, l, l-(n-k), t)$. In other words, if t is small, then the adversary is able to recover $\boldsymbol{m}$ easily by solving $\mathsf{RSD}(q, m, l, l-(n-k), t)$.

4 Gaborit's Attack on General McNie

We describe briefly Gaborit's attack on the general McNie encryption, which reduces $\boldsymbol{m} = (m_1, \ldots, m_l) \in \mathbb{F}_{q^m}^l$ to $\boldsymbol{m'} = (m_{n-k+1}, \ldots, m_l) \in \mathbb{F}_{q^m}^{l-(n-k)}$. We refer readers to [5] for complete idea of the attack.

Gaborit's Attack. Denote the matrix $F = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix}$ where $F_1 \in \mathbb{F}_{q^m}^{(n-k)\times(n-k)}$ and $F_2 \in \mathbb{F}_{q^m}^{(l-(n-k))\times(n-k)}$, then $\boldsymbol{c}_2 = \boldsymbol{m}F = (m_1, \ldots, m_{n-k})F_1 + \boldsymbol{m'}F_2$, giving

us $(m_1, \ldots, m_{n-k}) = \boldsymbol{c}_2 F_1^{-1} - \boldsymbol{m'} F_2 F_1^{-1}$. Substituting this into $\boldsymbol{c}_1$, we have $\boldsymbol{c}_1 = \boldsymbol{m}G' + \boldsymbol{e} = (\boldsymbol{c}_2 F_1^{-1} - \boldsymbol{m}' F_2 F_1^{-1}, \boldsymbol{m'})G' + \boldsymbol{e}$, resulting in a linear system of n equations with $l-(n-k)$ unknowns in $\boldsymbol{m'}$ and n unknowns in $\boldsymbol{e}$ to be solved.

As a result, the actual complexity is reduced for the suggested parameters of McNie based on QC-LRPC codes. Taking account of Gaborit's attack, Galvez et al. revised their parameters for McNie using 3-Quasi-Cyclic and 4-Quasi-Cyclic LRPC codes [11] in Table 1.

Table 1. Revised parameters for 3-Quasi-Cyclic and 4-Quasi-Cyclic LRPC

3-Quasi-Cyclic LRPC										
n	k	l	d	r	m	q	Failure 1	Failure 2	Security achieved	Key size
120	80	80	3	8	53	2	2^{-17}	2^{-42}	128	0.795 KB
138	92	92	3	10	67	2	2^{-17}	2^{-54}	192	1.156 KB
156	104	104	3	12	71	2	2^{-17}	2^{-46}	256	1.385 KB
4-Quasi-Cyclic LRPC										
92	46	69	3	10	59	2	2^{-17}	2^{-38}	128	0.848 KB
112	56	84	3	13	67	2	2^{-18}	2^{-30}	192	1.173 KB
128	64	96	3	16	73	2	2^{-17}	2^{-18}	256	1.460 KB

The "Failure 1" [8] refers to the probability of failure in decryption that the $n-k$ syndromes does not generate the product space $P = \langle E.F \rangle$ for recovering the error vector $\boldsymbol{e}$. "Failure 2" [8] refers to the probability that $\dim(\cap S_i) \neq r$ where $S_i = F_i^{-1} S = \langle F_i^{-1} s_1, \ldots, F_i^{-1} s_{n-k} \rangle_{\mathbb{F}_q}$.

5 Our Key Recovery Attack on McNie Based on QC-LRPC Codes

In this section, we will first describe our idea to transform the quadratic system $F = G' P^{-1} H^T S$ into a liner system for a general McNie system. Then, we will apply our idea for both the 3-Quasi-Cyclic and 4-Quasi-Cyclic cases and reduce the problem to solve for H and S into an RSD problem. Our key recovery attack shows that both the original parameters in [10] and revised parameters in [11] do not achieve the claimed security.

5.1 Our Idea - Transforming Quadratic System into Linear System

Recall that G' and $F = G' P^{-1} H^T S$ are both known to the adversaries. Note that P is an identity matrix. Now consider

$$F = G' H^T S. \tag{1}$$

We have a quadratic system consisting $l \times (n-k)$ equations and $(n-k) \times (2n-k)$ unknowns to be solved ($(n-k) \times n$ unknowns from H and $(n-k) \times (n-k)$ unknowns from S). The complexity of solving (1) is large, as $l < n < 2n-k$.

However, since S is an invertible matrix, we can rewrite the system (1) into the following new system:

$$FS^{-1} = G'H^T. \tag{2}$$

Now, instead of having a quadratic system as in (1), we have a linear system (2) consisting $l \times (n-k)$ equations and $(n-k) \times (2n-k)$ unknowns to be solved ($(n-k) \times n$ unknowns from H and $(n-k) \times (n-k)$ unknowns from S). This reduces the difficulty of solving S and H from a quadratic system to a linear system. In general, linear system (2) is not easy to be solved when the number of unknowns is greater than the number of equations in the linear system, i.e., $(n-k) \times (2n-k) > l \times (n-k)$.

5.2 Attack Against 3-Quasi-Cyclic LRPC Codes

Recall the settings in 3-Quasi-Cyclic LRPC codes with $l = k = \frac{2n}{3}$, $P = I_n$, $G' = \begin{pmatrix} I_{\frac{n}{3}} & \mathbf{0}_{\frac{n}{3}} & G_1 \\ \mathbf{0}_{\frac{n}{3}} & I_{\frac{n}{3}} & G_2 \end{pmatrix}$, $H = (H_1\, H_2\, H_3)$ and $F = \begin{pmatrix} I_{\frac{n}{3}} \\ F' \end{pmatrix}$ where H_1, H_2, H_3, G_1, G_2 are $\frac{n}{3} \times \frac{n}{3}$ circulant matrices and $F' = (H_2^T + G_2 H_3^T)(H_1 + H_3 G_1^T)^{-1}$ is an $\frac{n}{3} \times \frac{n}{3}$ circulant matrix. Consider the system of the form (2), we have

$$FS^{-1} = \begin{pmatrix} I_{\frac{n}{3}} \\ F' \end{pmatrix} S^{-1} = \begin{pmatrix} S^{-1} \\ F'S^{-1} \end{pmatrix},$$

$$G'P^{-1}H^T = \begin{pmatrix} I_{\frac{n}{3}} & \mathbf{0}_{\frac{n}{3}} & G_1 \\ \mathbf{0}_{\frac{n}{3}} & I_{\frac{n}{3}} & G_2 \end{pmatrix} \begin{pmatrix} H_1^T \\ H_2^T \\ H_3^T \end{pmatrix} = \begin{pmatrix} H_1^T + G_1 H_3^T \\ H_2^T + G_2 H_3^T \end{pmatrix}. \tag{3}$$

Let $\boldsymbol{h_i}$ be the vector that induces the circulant matrix H_i for $i = 1, 2, 3$. Since

$$FS^{-1} = G'P^{-1}H^T \quad \Leftrightarrow \quad \begin{pmatrix} S^{-1} \\ F'S^{-1} \end{pmatrix} = \begin{pmatrix} H_1^T + G_1 H_3^T \\ H_2^T + G_2 H_3^T \end{pmatrix},$$

by substituting $S^{-1} = H_1^T + G_1 H_3^T$, we have

$$F'S^{-1} = F'(H_1^T + G_1 H_3^T) = H_2^T + G_2 H_3^T$$

$$\Rightarrow\ \mathbf{0}_{\frac{n}{3} \times \frac{n}{3}} = F'H_1^T - H_2^T + (F'G_1 - G_2)H_3^T = (F', -I_{\frac{n}{3}}, F'G_1 - G_2) \begin{pmatrix} H_1^T \\ H_2^T \\ H_3^T \end{pmatrix}.$$

Let $\mathcal{H} = (F', -I_{\frac{n}{3}}, F'G_1 - G_2)$. By viewing the matrices $H_1, \ldots, H_3$ as vector, we can rewrite the equations $\mathbf{0}_{\frac{n}{3} \times \frac{n}{3}}$ as

$$\mathbf{0}_{\frac{n}{3}} = (\boldsymbol{h_1}, \boldsymbol{h_2}, \boldsymbol{h_3})\mathcal{H}^T. \tag{4}$$

Since the coefficients of H generate a vector subspace with dimension at most d, the vector $\mathfrak{h} = (\boldsymbol{h_1}, \boldsymbol{h_2}, \boldsymbol{h_3})$ is then a vector with $\mathrm{rk}_q(\mathfrak{h}) = d$. Solving (4) to recover $\mathfrak{h}$ is now equivalent to an $\mathsf{RSD}(q, m, n, 2\left(\frac{n}{3}\right), d)$ problem.

5.3 Attack Against 4-Quasi-Cyclic LRPC Codes

Recall the settings in 4-Quasi-Cyclic LRPC codes, $l = \frac{3n}{4}$ and $k = \frac{n}{2}$. Taking $P = I_n$, $G' = \begin{pmatrix} I_{\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}} & G_1 \\ \mathbf{0}_{\frac{n}{4}} & I_{\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}} & G_2 \\ \mathbf{0}_{\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}} & I_{\frac{n}{4}} & G_3 \end{pmatrix}$, $H = \begin{pmatrix} H_1 & H_2 & H_3 & H_4 \\ H_5 & H_6 & H_7 & H_8 \end{pmatrix}$ and $\bar{S} = \begin{pmatrix} S_1 & S_2 \\ S_3 & S_4 \end{pmatrix}$ where $G_1, \ldots, G_3, S_1, \ldots, S_4, H_1, \ldots, H_8$ are $\frac{n}{4} \times \frac{n}{4}$ circulant matrices. Then compute $\bar{F} = G'P^{-1}H^T\bar{S}$ and reduce $\bar{F}$ into column echelon form, $F = \bar{F}E = \begin{pmatrix} I_{\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}} \\ \mathbf{0}_{\frac{n}{4}} & I_{\frac{n}{4}} \\ F' & F'' \end{pmatrix}$ where F' and F'' are $\frac{n}{4} \times \frac{n}{4}$ circulant matrices. Then we have $F = \bar{F}E = G'P^{-1}H^T\bar{S}E$. Let $S = \bar{S}E$ and $S^{-1} := \begin{pmatrix} \hat{S}_1 & \hat{S}_2 \\ \hat{S}_3 & \hat{S}_4 \end{pmatrix}$. Consider the system of the form (2), we have

$$FS^{-1} = \begin{pmatrix} I_{\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}} \\ \mathbf{0}_{\frac{n}{4}} & I_{\frac{n}{4}} \\ F' & F'' \end{pmatrix} \begin{pmatrix} \hat{S}_1 & \hat{S}_2 \\ \hat{S}_3 & \hat{S}_4 \end{pmatrix} = \begin{pmatrix} \hat{S}_1 & \hat{S}_2 \\ \hat{S}_3 & \hat{S}_4 \\ F'\hat{S}_1 + F''\hat{S}_3 & F'\hat{S}_2 + F''\hat{S}_4 \end{pmatrix}, \tag{5}$$

$$G'P^{-1}H^T = \begin{pmatrix} I_{\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}} & G_1 \\ \mathbf{0}_{\frac{n}{4}} & I_{\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}} & G_2 \\ \mathbf{0}_{\frac{n}{4}} & \mathbf{0}_{\frac{n}{4}} & I_{\frac{n}{4}} & G_3 \end{pmatrix} \begin{pmatrix} H_1^T & H_5^T \\ H_2^T & H_6^T \\ H_3^T & H_7^T \\ H_4^T & H_8^T \end{pmatrix} = \begin{pmatrix} H_1^T + G_1H_4^T & H_5^T + G_1H_8^T \\ H_2^T + G_2H_4^T & H_6^T + G_2H_8^T \\ H_3^T + G_3H_4^T & H_7^T + G_3H_8^T \end{pmatrix}.$$

Let $\boldsymbol{h_i}$ be the vector that induces the circulant matrix H_i for $i = 1, \ldots, 8$. Since $FS^{-1} = G'P^{-1}H^T$, we have

$$\begin{pmatrix} \hat{S}_1 & \hat{S}_2 \\ \hat{S}_3 & \hat{S}_4 \\ F'\hat{S}_1 + F''\hat{S}_3 & F'\hat{S}_2 + F''\hat{S}_4 \end{pmatrix} = \begin{pmatrix} H_1^T + G_1H_4^T & H_5^T + G_1H_8^T \\ H_2^T + G_2H_4^T & H_6^T + G_2H_8^T \\ H_3^T + G_3H_4^T & H_7^T + G_3H_8^T \end{pmatrix},$$

by substituting $\hat{S}_1 = H_1^T + G_1H_4^T$ and $\hat{S}_3 = H_2^T + G_2H_4^T$, we have

$$\begin{aligned} F'\hat{S}_1 + F''\hat{S}_3 &= F'(H_1^T + G_1H_4^T) + F''(H_2^T + G_2H_4^T) = H_3^T + G_3H_4^T \\ \Rightarrow \quad \mathbf{0}_{\frac{n}{4} \times \frac{n}{4}} &= F'H_1^T + F''H_2^T - H_3^T + (F'G_1 + F''G_2 - G_3)H_4^T \\ &= (F', F'', -I_{\frac{n}{4}}, F'G_1 + F''G_2 - G_3) \begin{pmatrix} H_1^T \\ H_2^T \\ H_3^T \\ H_4^T \end{pmatrix}. \end{aligned} \tag{6}$$

Let $\mathcal{H} = (F', F'', -I_{\frac{n}{4}}, F'G_1 + F''G_2 - G_3)$. By viewing the matrices $H_1, \ldots, H_4$ as vector, we can rewrite (6) as

$$\mathbf{0}_{\frac{n}{4}} = (\boldsymbol{h_1}, \boldsymbol{h_2}, \boldsymbol{h_3}, \boldsymbol{h_4})\mathcal{H}^T. \tag{7}$$

Since the coefficients of H generate a vector subspace with dimension at most d, the vector $\mathfrak{h}_1 = (\boldsymbol{h_1}, \ldots, \boldsymbol{h_4})$ is then a vector with $\mathrm{rk}_q(\mathfrak{h}_1) = d$. Solving (7) to recover $\mathfrak{h}_1$ is now equivalent to an $\mathsf{RSD}(q, m, n, 3\left(\frac{n}{4}\right), d)$ problem.

Similarly, we can repeat the procedure above for $\hat{S}_2$ and $\hat{S}_4$ to obtain

$$\mathbf{0}_{\frac{n}{4}} = (\boldsymbol{h_5}, \boldsymbol{h_6}, \boldsymbol{h_7}, \boldsymbol{h_8})\mathcal{H}^T. \tag{8}$$

Let $\mathfrak{h}_\mathbf{2} = (\boldsymbol{h_5}, \ldots, \boldsymbol{h_8})$. Then, solving (8) to recover $\mathfrak{h}_\mathbf{2}$ is now equivalent to an $\mathsf{RSD}(q, m, n, 3\left(\frac{n}{4}\right), d)$ problem.

5.4 Recovering Plaintext from Our Recovered $\bar{H}$ and $\bar{S}$

Suppose that vectors $\bar{\boldsymbol{h}}_i$ are recovered and $\bar{H}_i$ are generated by $\bar{\boldsymbol{h}}_i$, then we can determine a matrix $\bar{S}^{-1}$, satisfying $F = G'\bar{H}^T\bar{S}$. We can use $\bar{H}$ and $\bar{S}$ to compute $\boldsymbol{c}_1\bar{H}^T - \boldsymbol{c}_2\bar{S}^{-1} = (\boldsymbol{m}G' + \boldsymbol{e})\bar{H}^T - (\boldsymbol{m}G'\bar{H}^T\bar{S})\bar{S}^{-1} = \boldsymbol{e}\bar{H}^T$. Since $\bar{H}$ is a parity check matrix of LRPC code and $\bar{H}$ is known, we can decode $\boldsymbol{e}\bar{H}^T$ and recover $\boldsymbol{e}$, thus retrieving $\boldsymbol{m}$ by solving $\boldsymbol{m}G' = \boldsymbol{c}_1 - \boldsymbol{e}$.

5.5 Actual Security Level of Proposed Parameters

Table 2 shows the key recovery attack complexity (KRA) in solving the problem $\mathsf{RSD}(q, m, n, 2\left(\frac{n}{3}\right), d)$ and $\mathsf{RSD}(q, m, n, 3\left(\frac{n}{4}\right), d)$ using formulas in Sect. 2.3. For all the original and revised parameters of 3-Quasi-Cyclic and 4-Quasi-Cyclic LRPC codes suggested by Galvez et al., our key recovery attack is able to recover the secret key H and S of the proposed security level (Claimed Security).

Table 2. Complexity to recover the secret key of 3-Quasi-Cyclic and 4-Quasi-Cyclic LRPC using key recovery attack (KRA)

Original parameters in [10]																	
3-Quasi-Cyclic LRPC									4-Quasi-Cyclic LRPC								
n	k	l	d	r	m	q	Claimed security	KRA	n	k	l	d	r	m	q	Claimed security	KRA
84	56	56	3	4	29	2	80	59	48	24	36	2	4	29	2	80	42
96	64	64	3	4	29	2	80	59	56	28	42	2	4	37	2	80	47
93	62	62	3	5	37	2	128	68	60	30	45	3	5	37	2	128	76
105	70	70	3	5	37	2	128	69	72	36	54	3	5	37	2	128	76
111	74	74	3	7	41	2	192	73	76	38	57	3	7	41	2	192	82
123	82	82	3	7	41	2	192	74	84	42	63	3	7	41	2	192	82
111	74	74	3	7	59	2	256	93	76	38	57	3	7	53	2	256	99
141	94	94	3	9	47	2	256	81	88	44	66	3	8	47	2	256	91
Revised parameters in [11]																	
120	80	80	3	8	53	2	128	87	92	46	69	3	10	59	2	128	107
138	92	92	3	10	67	2	192	103	112	56	84	3	13	67	2	192	119
156	104	104	3	12	71	2	256	107	128	64	96	3	16	73	2	256	127

For instance, our key recovery attack can recover all the secret key for the revised parameters of 3-Quasi-Cyclic LRPC of claimed security level of 256-bit in 2^{107} complexity. This implies that to achieve the claimed security, it is required to adjust the parameters, resulting in increase in the key size of the encryption scheme.

6 A New Proposal - McNie Based on Gabidulin Codes

In this section we use Gabidulin code as the r-error correcting code over a finite field $\mathbb{F}_{q^m}$ for the McNie encryption. We consider an l-partical circulant induced by $\boldsymbol{u}$ of $\mathrm{rk}_q(\boldsymbol{u}) = n$ as the G' in the encryption. Now, we give the definition for Moore matrix and Gabidulin codes.

Definition 10. A matrix $G \in \mathbb{F}_{q^m}^{k\times n}$ is called a *Moore matrix* if there exists $\boldsymbol{g} = (g_1, \ldots, g_n) \in \mathbb{F}_{q^m}^n$ such that $G = \left(g_j^{[i-1]}\right)_{i,j}$ for $1 \le i \le k$ and $1 \le j \le n$, where $[i] := q^i$ is the ith Frobenius power. If $\boldsymbol{g} = (g_1, \ldots, g_n) \in \mathbb{F}_{q^m}^n$ with $\mathrm{rk}_q(\boldsymbol{g}) = n$, then the $[n,k]$-*Gabidulin code* $\mathrm{Gab}_{n,k}(\boldsymbol{g})$ over $\mathbb{F}_{q^m}$ of dimension k with generator vector $\boldsymbol{g}$ is the code generated by the matrix G.

The error-correcting capability of $\mathrm{Gab}_{n,k}(\boldsymbol{g})$ is $r = \lfloor \frac{n-k}{2} \rfloor$. There exist efficient deterministic decoding algorithms for Gabidulin codes up to the rank error correcting capability (for example [4]). Therefore, our McNie based on Gabidulin codes has error free decryption.

6.1 McNie Based on Gabidulin Codes

Setup, $\mathcal{S}_{\mathtt{PE}}$ Generates global parameters m, n, l, k, k', r, t_1 and t_2 such that $m \ge n > l > n-k$, $l = k' + r$, and $t_1 + t_2 \le r = \lfloor \frac{n-k}{2} \rfloor$. The plaintext space is $\mathbb{F}_{q^m}^{k'}$. Outputs $\mathrm{param} = (m, n, l, k, k', r, t_1, t_2)$.

Key Generation, $\mathcal{K}_{\mathtt{PE}}$ Generates a vector $\boldsymbol{g} \xleftarrow{\$} \mathbb{F}_{q^m}^n$ such that $\mathrm{rk}_q(\boldsymbol{g}) = n$. Computes a parity check matrix $H \in \mathbb{F}_{q^m}^{(n-k)\times n}$ of $\mathrm{Gab}_{n,k}(\boldsymbol{g})$ with an efficient syndrome decoding algorithm $\mathrm{Gab}_{n,k}(\boldsymbol{g}).\mathfrak{SDec}(\cdot)$. Generates a vector $\boldsymbol{u} \xleftarrow{\$} \mathbb{F}_{q^m}^n$ such that $\mathrm{rk}_q(\boldsymbol{u}) = n$. Constructs a l-partial circulant matrix, $G' = \mathrm{Cir}_l(\boldsymbol{u})$. Generates $P_1 \xleftarrow{\$} \mathbb{F}_{q^m}^{n\times t_2}$ and $P_2 \xleftarrow{\$} \mathbb{F}_q^{n\times(n-t_2)}$ such that $P = (P_1 \mid P_2)$ is invertible. Generates $S \xleftarrow{\$} \mathrm{GL}_{n-k}(\mathbb{F}_{q^m})$ such that $\bar{F} = G'PH^TS$ could be reduced into column echelon form, i.e., there exists $E \in \mathrm{GL}_{n-k}(\mathbb{F}_{q^m})$ such that $F = \bar{F}E = \begin{pmatrix} I_{n-k} \\ \hat{F} \end{pmatrix}$, where $\hat{F} \in \mathbb{F}_{q^m}^{(l-(n-k))\times(n-k)}$. Outputs public key $\kappa_{pub} = (F, G')$ and secret key $\kappa_{sec} = (E_S := E^{-1}S^{-1}, \boldsymbol{g}, P)$.

Encryption, $\mathcal{E}_{\mathtt{PE}}(\kappa_{pub} = (F, G'), \boldsymbol{m})$ Let $\boldsymbol{m} \in \mathbb{F}_{q^m}^{k'}$ be the message to be encrypted. Generates $\boldsymbol{s} \xleftarrow{\$} \mathbb{F}_{q^m}^r$ such that $\mathrm{rk}_q(\boldsymbol{s}) = r$. Generates $\boldsymbol{e} \xleftarrow{\$} \mathbb{F}_{q^m}^n$ such that $\mathrm{rk}_q(\boldsymbol{e}) \le t_1$. Outputs $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2) = ((\boldsymbol{m}\|\boldsymbol{s})G' + \boldsymbol{e}, (\boldsymbol{m}\|\boldsymbol{s})F)$ as the ciphertext.

Decryption, $\mathcal{D}_{\mathtt{PE}}(\kappa_{sec} = (E_S, \boldsymbol{g}, P), \boldsymbol{c})$ Using P, H and $E_S = E^{-1}S^{-1}$, computes $\boldsymbol{c}_1PH^T - \boldsymbol{c}_2E_S = ((\boldsymbol{m}\|\boldsymbol{s})G' + \boldsymbol{e})PH^T - (\boldsymbol{m}\|\boldsymbol{s})(G'PH^TSE)E^{-1}S^{-1} = \boldsymbol{e}PH^T$. Since $\mathrm{rk}_q(\boldsymbol{e}P) \le \mathrm{rk}_q(\boldsymbol{e}P_1) + \mathrm{rk}_q(\boldsymbol{e}P_2) \le t_2 + t_1 \le r$ and $(\boldsymbol{e}P)H^T$ is a syndrome of $\mathrm{Gab}_{n,k}(\boldsymbol{g})$, computes $\boldsymbol{e} = \mathrm{Gab}_{n,k}(\boldsymbol{g}).\mathfrak{SDec}(\boldsymbol{e}PH^T)P^{-1} = \boldsymbol{e}PP^{-1}$. Retrieves $\boldsymbol{m}$ by solving the system $(\boldsymbol{m}\|\boldsymbol{s})G' = \boldsymbol{c}_1 - \boldsymbol{e}$.

Remark 1. Let $G' = \begin{pmatrix} G_1 \\ G_2 \end{pmatrix}$, since $F = \begin{pmatrix} I_{n-k} \\ \hat{F} \end{pmatrix} = G'PH^TSE$, the secret key E_S could be calculated from G_1, H and P, i.e., $E_S = G_1PH^T$. Therefore we do not need to store E_S as a part of the secret key.

6.2 Practical Security

We consider different attacks on our McNie reparation and discuss the complexity of the attacks:

1. Gaborit's Attack: The complexity of combinatorial and algebraic attacks on $\boldsymbol{c}_1$ are calculated based on the complexities in Sect. 2.3, by replacing the term "k" in the complexity with "$l-(n-k)$".
2. Key Recovery Attack: Our $F = G'PH^TS$ is a cubic multivariate system of equations. Although we can transform $F = G'PH^TS$ into $FS^{-1} = GPH^T$, it is a quadratic multivariate system of equations, which has large solving complexity. Moreover, our H is not block-circulant matrix and the entries in H is not of low rank, therefore this attack is not applicable.
3. Recovering Plaintext of Low Rank: In our proposal, we have $\mathrm{rk}_q(\boldsymbol{m}\|\boldsymbol{s}) \geq r$ and $\boldsymbol{c}_2 = (\boldsymbol{m}\|\boldsymbol{s})F$. Since $\mathrm{rk}_q(\boldsymbol{m}\|\boldsymbol{s}) \geq r$, we set our parameters in a way that the adversary will take more complexities than the security level to recover the plaintext.
4. Overbeck's Attack: In Overbeck's attack [16] on $G_{pub} = S(X \mid G)P$ where $S \in \mathrm{GL}_k(\mathbb{F}_{q^m})$, $X \in \mathbb{F}_{q^m}^{k\times t_1}$, $P \in \mathrm{GL}_{n+t_1}(\mathbb{F}_q)$, and G is a $k \times n$ Moore matrix. Consider the code $\bar{\mathcal{C}}$ generated by $\bar{G}^T = \left(G_{pub}^T, \ldots, (G_{pub}^{[n-k-1]})^T\right)$, since $G_{pub}^{[i]} = S^{[i]}(X^{[i]} \mid G^{[i]})P$, then $\ker(\bar{\mathcal{C}}) = n+t_1-1$. An alternative column scrambler matrix $\bar{P}$ over $\mathbb{F}_q$ could be computed, giving $G_{pub}\bar{P}^{-1} = S(Z \mid G^*)$ where G^* is a Moore matrix. In our construction, $F^T = S^THP^T(G')^T$ and $(F^T)^{[i]} = (S^T)^{[i]}H^{[i]}(P^T)^{[i]}((G')^{[i]})^T$, the matrix $(P^T)^{[i]}((G')^{[i]})^T$ is not over $\mathbb{F}_q$, thus we are not able to determine an alternative column scrambler matrix $\bar{P}$ over $\mathbb{F}_q$ so that $F^T\bar{P}^{-1} = S^TH^*$, then Overbeck's attacks fails.
5. Otmani et al.'s Attack: In Otmani et al.'s attack [14], they need to transform the public key matrix G_{pub} into the form above, so that it could be attacked by Overbeck's attack. Let $A = S^THP^T$ and $F^T = A(G')^T$ in our construction. Assume A is known, the adversary wants to transform P^T into the form $Q = \begin{pmatrix} Q_1 & \mathbf{0} \\ Q_2 & Q_4 \end{pmatrix}$ where Q_4 is a matrix over $\mathbb{F}_q$. Let $\bar{S} = S^T$ and $\bar{P} = P^T = \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix}$, where P_{11} and P_{12} are over $\mathbb{F}_{q^m}$, P_{21} and P_{22} are over $\mathbb{F}_q$. Rewrite $A = \bar{S}H \begin{pmatrix} I_{t_2} & P' \\ \mathbf{0} & I_{n-t_2} \end{pmatrix} \begin{pmatrix} I_{t_2} & -P' \\ \mathbf{0} & I_{n-t_2} \end{pmatrix} \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} = \bar{S}H \begin{pmatrix} I_{t_2} & P' \\ \mathbf{0} & I_{n-t_2} \end{pmatrix} \begin{pmatrix} P'_{11} & \mathbf{0} \\ P_{21} & P_{22} \end{pmatrix} = \bar{S}(H_1 \mid H_1P' + H_2) \begin{pmatrix} P'_{11} & \mathbf{0} \\ P_{21} & P_{22} \end{pmatrix}$ where $P' = P_{12}P_{22}^{-1}$ and $P'_{11} = P_{11} - P'P_{21}$. Then $A = \bar{S}(\bar{H}_{Moore} + X)$, where $\bar{H}_{Moore} =$

$(H_2P_{21} \mid H_2P_{22})$ and $X = (H_1P'_{11} + H_1P'P_{21} \mid H_1P'P_{22})$ of full column rank. Since X is of full column rank, therefore [14, Proposition 6] cannot be applied. Therefore Otmani et al.'s attack on A is not applicable.

6.3 IND-CPA Secure Encryption

The desired security properties of a public-key encryption scheme is indistinguishability under chosen plaintext attack (IND-CPA). This is normally defined by a security game which is interacting between a challenger and an adversary $\mathcal{A}$. The security game is described as follows:

Set up: Given a security parameter, the challenger first runs the key generation algorithm and send κ_{pub} to $\mathcal{A}$. **Challenge:** $\mathcal{A}$ chooses two equal length plaintexts $\boldsymbol{m}_0$ and $\boldsymbol{m}_1$; and sends these to the challenger. **Encrypt challenge messages:** The challenger chooses a random $b \in \{0,1\}$, computes a challenge ciphertext $\boldsymbol{c} = \mathcal{E}_{\mathtt{PE}}(\kappa_{pub}, \boldsymbol{m}_b)$ and returns $\boldsymbol{c}$ to $\mathcal{A}$. **Guess:** $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. $\mathcal{A}$ wins if $b' = b$.

The advantage of an adversary $\mathcal{A}$ is defined as $\mathsf{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PE},\mathcal{A}}(\lambda) = \left|\Pr[b' = b] - \frac{1}{2}\right|$. A secure public-key encryption scheme against chosen plaintext attack is formally defined as follows:

Definition 11. A public-key encryption scheme $\mathtt{PE} = (\mathcal{S}_{\mathtt{PE}}, \mathcal{K}_{\mathtt{PE}}, \mathcal{E}_{\mathtt{PE}}, \mathcal{D}_{\mathtt{PE}})$ is (t, ϵ)-IND-CPA secure if for any probabilistic t-polynomial time adversary $\mathcal{A}$ has the advantage less than ϵ, that is, $\mathsf{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PE},\mathcal{A}}(\lambda) < \epsilon$.

We will show that our encryption has indistinguishability under chosen plaintext attack (IND-CPA). Denote $\mathcal{E}_{n,w} := \{\boldsymbol{x} : \boldsymbol{x} \in \mathbb{F}_{q^m}^n, \mathrm{rk}_q(\boldsymbol{x}) = w\}$. We first describe:

The Decisional Rank Syndrome Decoding (DRSD) assumption. Let $\mathcal{D}$ be a distinguishing algorithm that takes as input a vector in $\mathbb{F}_{q^m}^{n-k}$ and a matrix $Q \in \mathbb{F}_{q^m}^{n\times(n-k)}$, and outputs a bit. The DRSD advantage of $\mathcal{D}$ is defined as

$$\mathsf{Adv}^{\mathtt{DRSD}}_{Q,n,k}(\mathcal{D}) = \Big|\Pr\left[\boldsymbol{e} \xleftarrow{\$} \mathcal{E}_{n,w}, \boldsymbol{u} = \boldsymbol{e}Q : \mathcal{D}(Q,\boldsymbol{u}) = 1\right] - \Pr\left[\boldsymbol{z} \xleftarrow{\$} \mathbb{F}_{q^m}^{n-k} : \mathcal{D}(Q,\boldsymbol{z}) = 1\right]\Big|.$$

The DRSD assumption is the assumption that the advantage $\mathsf{Adv}^{\mathtt{DRSD}}_{Q,n,k}(\mathcal{D})$ is negligible for any $\mathcal{D}$, i.e., $\mathsf{Adv}^{\mathtt{DRSD}}_{Q,n,k}(\mathcal{D}) < \varepsilon_Q$.

Consider the dual problem of RSD problem, we define:

The Decisional Rank Syndrome Decoding Dual (DRSDD) Assumption. Let $\mathcal{D}$ be a distinguishing algorithm that takes as input a vector in $\mathbb{F}_{q^m}^n$ and a matrix $M \in \mathbb{F}_{q^m}^{k\times n}$, and outputs a bit. The DRSDD advantage of $\mathcal{D}$ is defined as

$$\mathsf{Adv}^{\mathtt{DRSDD}}_{M,n,k}(\mathcal{D}) = \Big|\Pr\left[\boldsymbol{v} \xleftarrow{\$} \mathbb{F}_{q^m}^k, \boldsymbol{e} \xleftarrow{\$} \mathcal{E}_{n,w}, \boldsymbol{x} = \boldsymbol{v}M + \boldsymbol{e} : \mathcal{D}(M,\boldsymbol{x}) = 1\right] - \Pr\left[\boldsymbol{y} \xleftarrow{\$} \mathbb{F}_{q^m}^n : \mathcal{D}(M,\boldsymbol{y}) = 1\right]\Big|.$$

The DRSDD assumption is the assumption that the advantage $\mathsf{Adv}^{\mathsf{DRSDD}}_{M,n,k}(\mathcal{D})$ is negligible for any $\mathcal{D}$, i.e., $\mathsf{Adv}^{\mathsf{DRSDD}}_{M,n,k}(\mathcal{D}) < \varepsilon_M$.

We state the assumptions for which our encryption is based on:

Assumption 1. Let $(F, \mathrm{Cir}_l(\boldsymbol{u}))$ be the public key in our encryption, where $F \in \mathbb{F}_{q^m}^{l\times(n-k)}$ and $\mathrm{Cir}_l(\boldsymbol{u}) \in \mathbb{F}_{q^m}^{l\times n}$. The DRSD_F assumption is the assumption that $\mathsf{Adv}^{\mathsf{DRSD}}_{F,l,l-(n-k)}(\mathcal{D})$ is negligible for any $\mathcal{D}$, i.e., $\mathsf{Adv}^{\mathsf{DRSD}}_{F,l,l-(n-k)}(\mathcal{D}) < \varepsilon_F$. The $\mathsf{DRSDD}_{\mathrm{Cir}_l(\boldsymbol{u})}$ assumption is the assumption that $\mathsf{Adv}^{\mathsf{DRSDD}}_{\mathrm{Cir}_l(\boldsymbol{u}),n,l}(\mathcal{D})$ is negligible for any $\mathcal{D}$, i.e., $\mathsf{Adv}^{\mathsf{DRSDD}}_{\mathrm{Cir}_l(\boldsymbol{u}),n,l}(\mathcal{D}) < \varepsilon_{\mathrm{Cir}_l(\boldsymbol{u})}$.

We will now prove that our encryption is IND-CPA secure under DRSD_F and $\mathsf{DRSDD}_{\mathrm{Cir}_l(\boldsymbol{u})}$ assumptions.

Theorem 1. Under the $\mathsf{DRSDD}_{\mathrm{Cir}_l(\boldsymbol{u})}$ and DRSD_F assumptions, our proposed public-key encryption scheme McNie PE is IND-CPA secure.

Proof. To prove the security of the scheme, we are using a sequence of games. Let CEF be an algorithm that inputs a matrix A and output (B, C) where B is the column echelon form of A and C is an invertible matrix such that $B = AC$.

Game $\mathcal{G}_0$: This is the real IND-CPA attack game against an adversary $\mathcal{A}$ in the definition of semantic security. We run the following attack game algorithm:

$P \xleftarrow{\$} \Big\{(P_1|P_2) : P_1 \in \mathbb{F}_q^{n\times(n-t_2)}, P_2 \in \mathbb{F}_{q^m}^{n\times t_2}, (P_1|P_2) \in \mathrm{GL}_n(\mathbb{F}_{q^m})\Big\}$,
$S \xleftarrow{\$} \mathrm{GL}_{n-k}(\mathbb{F}_{q^m})$, $\boldsymbol{u} \xleftarrow{\$} \{\boldsymbol{u} : \boldsymbol{u} \in \mathbb{F}_{q^m}^n, \mathrm{rk}_q(\boldsymbol{u}) = n\}$, $G' \leftarrow \mathrm{Cir}_l(\boldsymbol{u})$,
$(F, E) \leftarrow \mathtt{CEF}(G'PH^TS)$, $E_S \leftarrow E^{-1}S^{-1}$,
$\kappa_{pub} \leftarrow (F, G')$, $\kappa_{sec} \leftarrow (E_S, \boldsymbol{g}, P)$
$(m_0, m_1) \xleftarrow{\$} \mathcal{A}(\kappa_{pub})$
$b \xleftarrow{\$} \{0, 1\}$, $\boldsymbol{s} \xleftarrow{\$} \Big\{\boldsymbol{s} : \boldsymbol{s} \in \mathbb{F}_{q^m}^{l-k'}, \mathrm{rk}_q(\boldsymbol{s}) = r\Big\}$,
$\boldsymbol{e} \xleftarrow{\$} \big\{\boldsymbol{x} : \boldsymbol{x} \in \mathbb{F}_{q^m}^n, \mathrm{rk}_q(\boldsymbol{x}) \le r\big\}$,
$\boldsymbol{c}_1 \leftarrow (\boldsymbol{m}_b\|\boldsymbol{s})G' + \boldsymbol{e}$, $\boldsymbol{c}_2 \leftarrow (\boldsymbol{m}_b\|\boldsymbol{s})F$
$\hat{b} \leftarrow \mathcal{A}(\kappa_{pub}, \boldsymbol{c}_1, \boldsymbol{c}_2)$
if $\hat{b} = b$ **then** return 1 **else** return 0

Denote $\mathcal{S}_0$ the event that $\mathcal{A}$ wins in Game $\mathcal{G}_0$. Then $\mathsf{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PE},\mathcal{A}}(\lambda) = \big|\Pr[\mathcal{S}_0] - \frac{1}{2}\big|$.

Game $\mathcal{G}_1$: We now make one small change to $\mathcal{G}_0$. In this game, we pick a random vector $\boldsymbol{y} \xleftarrow{\$} \mathbb{F}_{q^m}^n$ and replace $\boldsymbol{c}_1$ in $\mathcal{G}_0$ for $\mathcal{E}_{\mathtt{PE}}(\kappa_{pub}, (\boldsymbol{m}_b\|\boldsymbol{s}))$ by $\boldsymbol{c}_1 \leftarrow \boldsymbol{y}$. We denote $\mathcal{S}_1$ the event that $\mathcal{A}$ wins in Game $\mathcal{G}_1$. Under the $\mathsf{DRSDD}_{\mathrm{Cir}_l(\boldsymbol{u})}$ assumption, the two games $\mathcal{G}_1$ and $\mathcal{G}_0$ are indistinguishable with $|\Pr[\mathcal{S}_1] - \Pr[\mathcal{S}_0]| < \varepsilon_{\mathrm{Cir}_l(\boldsymbol{u})}$.

Game $\mathcal{G}_2$: We now make one small change to $\mathcal{G}_1$. In this game, we pick a random vector $\boldsymbol{z} \xleftarrow{\$} \mathbb{F}_{q^m}^{n-k}$ and replace $\boldsymbol{c}_2$ in $\mathcal{G}_1$ for $\mathcal{E}_{\mathtt{PE}}(\kappa_{pub}, (\boldsymbol{m}_b\|\boldsymbol{s}))$ by $\boldsymbol{c}_2 \leftarrow \boldsymbol{z}$. We denote $\mathcal{S}_2$ the event that $\mathcal{A}$ wins in Game $\mathcal{G}_2$. Under the DRSD_F assumption, the two games $\mathcal{G}_2$ and $\mathcal{G}_1$ are indistinguishable with $|\Pr[\mathcal{S}_2] - \Pr[\mathcal{S}_1]| < \varepsilon_F$.

As the ciphertext challenge $\boldsymbol{c} = (\boldsymbol{c}_1, \boldsymbol{c}_2)$ is perfectly random, b is hidden to any adversary $\mathcal{A}$ without any advantage, we have $\Pr[\mathcal{S}_2] = \frac{1}{2}$. Therefore, we have $\mathsf{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PE},\mathcal{A}}(\lambda) = |\Pr[\mathcal{S}_0] - 1/2| = |\Pr[\mathcal{S}_0] - \Pr[\mathcal{S}_2]| \leq |\Pr[\mathcal{S}_0] - \Pr[\mathcal{S}_1]| + |\Pr[\mathcal{S}_1] - \Pr[\mathcal{S}_2]| < \varepsilon_{\mathrm{Cir}_l(\boldsymbol{u})} + \varepsilon_F$. Under the $\mathsf{DRSDD}_{\mathrm{Cir}_l(\boldsymbol{u})}$ and DRSD_F assumption, our proposed public-key encryption scheme McNie $\mathtt{PE}$ is $\mathtt{IND\text{-}CPA}$ secure. □

6.4 Proposed Parameters

We propose the parameters of our McNie based on Gabidulin codes. Denote "Sec." as the achieved security, by considering all the complexities for attacks in Sect. 2.3. The public key size (denoted as "PK") is $\frac{(l-(n-k))(n-k)m+nm}{8}\log_2(q)$ bytes. While the secret key size (denoted as "SK") is $\frac{nm+n(n-t_2)+nt_2m}{8}\log_2(q)$ bytes. The ciphertext size (denoted as "CT") is $\frac{(2n-k)m}{8}\log_2(q)$ bytes (Table 3).

Table 3. Proposed parameters of McNie on Gabidulin codes

m	n	k	l	t_1	t_2	q	Sec.	PK	SK	CT
43	38	14	37	9	3	2	128	1.88KB	0.98KB	0.33KB
44	40	14	38	10	3	2	129	1.94KB	1.07KB	0.36KB
50	45	19	44	10	3	2	192	3.21KB	1.36KB	0.44KB
52	47	19	45	11	3	2	198	3.40KB	1.48KB	0.49KB
57	52	20	51	13	3	2	257	4.70KB	1.80KB	0.60KB
59	54	22	51	13	3	2	257	4.88KB	1.94KB	0.63KB

7 Conclusion

McNie based on QC-LRPC has limitations such as decryption failure and possible recovery of low rank plaintext of their claimed security level. We show that McNie based on QC-LRPC codes with the proposed parameters is not secure of the claimed security. Our key recovery attack can successfully recover the secret key for all the given original and revised parameters. As such, their parameters must be adjusted to achieve the required security level, resulting in higher key size. We overcome the weaknesses of McNie based on QC-LRPC, by proposing McNie based on Gabidulin codes, which has error free decryption. Our proposal has $\mathtt{IND\text{-}CPA}$ security under $\mathsf{DRSDD}_{\mathrm{Cir}_l(\boldsymbol{u})}$ and DRSD_F assumptions. We can explore the possibility to apply Fujisaki-Okamoto transformation [3] similar with the approach in [10] to convert our encryption into $\mathtt{IND\text{-}CCA2}$ in the random oracle model.

References

1. Aragon, N., Gaborit, P., Hauteville, A., Tillich, J.P.: Improvement of Generic Attacks on the Rank Syndrome Decoding Problem (2017). <hal-01618464>
2. Faugère, J.-C., Levy-dit-Vehel, F., Perret, L.: Cryptanalysis of MinRank. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 280–296. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_16
3. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. J. Cryptology **26**(1), 80–101 (2013)
4. Gabidulin, E.M.: Theory of codes with maximum rank distance. Problemy Peredachi Informatsii **21**(1), 3–16 (1985)
5. Gaborit, P.: Attack on McNie. In: Post-Quantum Cryptography, Round 1 Submmisions, McNie, Official Comments. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/official-comments/McNie-official-comment.pdf
6. Gaborit, P., Hauteville, A., Phan, D.H., Tillich, J.-P.: Identity-based encryption from codes with rank metric. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 194–224. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_7
7. Gaborit, P., Ruatta, O., Schrek, J.: On the complexity of the rank syndrome decoding problem. IEEE Trans. Inf. Theory **62**(2), 1006–1019 (2016)
8. Gaborit, P., Murat, G., Ruatta, O., Zémor, G.: Low Rank Parity Check codes and their application to cryptography. In: The Proceedings of Workshop on Coding and Cryptography (WCC) 2013, Borgen, Norway, pp. 168–180 (2013)
9. Gaborit, P., Zémor, G.: On the hardness of the decoding and the minimum distance problems for rank codes. IEEE Trans. Inf. Theory **62**(12), 7245–7252 (2016)
10. Galvez, L., Kim, J., Kim, M.J., Kim, Y., Lee, N.: McNie: Compact McEliece-Niederreiter Cryptosystem. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/McNie.zip
11. Galvez, L., Kim, J., Kim, M.J., Kim, Y., Lee, N.: New McNie parameters. In: Post-Quantum Cryptography, Round 1 Submmisions, McNie, Official Comments. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/official-comments/McNie-official-comment.pdf
12. Goubin, L., Courtois, N.T.: Cryptanalysis of the TTM cryptosystem. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 44–57. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_4
13. Levy-dit-Vehel, F., Perret, L.: Algebraic decoding of rank metric codes. In: Proceedings of YACC 2006, pp. 142–152 (2006)
14. Otmani, A., Kalachi, H.T., Ndjeya, S.: Improved Cryptanalysis of Rank Metric Schemes Based on Gabidulin Codes. CoRR abs/1602.08549 (2016)
15. Ourivski, A.V., Johansson, T.: New technique for decoding codes in the rank metric and its cryptography applications. Probl. Inf. Trans. **38**(3), 237–246 (2002)
16. Overbeck, R.: Structural attacks for public key cryptosystems based on Gabidulin codes. J. Cryptology **21**(2), 280–301 (2008)

Inference Attacks on Encrypted Databases Based on Order Preserving Assignment Problem

Sota Onozawa[1], Noboru Kunihiro[1], Masayuki Yoshino[2(✉)], and Ken Naganuma[1,2]

[1] The University of Tokyo, Tokyo, Japan
[2] Hitachi, Ltd., Tokyo, Japan
masayuki.yoshino.aa@hitachi.com

Abstract. In ACM CCS 2015, Naveed et al. proposed attacks using plaintext auxiliary data for databases encrypted by ordered preserving encryption or more general property preserving encryptions. Their attacks are based on the Hungarian algorithm for solving the linear sum assignment problem (LSAP). In this work, we define a new assignment optimization problem with an additional condition of order structure and propose a search algorithm for finding its exact solution. We apply the new algorithm to attack an encrypted database in the same situation as Naveed et al. and found that our proposed method improves the success probability of the attacks compared with the attacks of Naveed et al.

Keywords: Linear sum assignment problem
Order-preserving encryption · Inference attacks · Encrypted databases

1 Introduction

The migration of the computing environment to the cloud is a critical issue, particularly for online applications [15]. Curious administrators on the cloud can snoop on all data, even if the data are encrypted by a standard encryption scheme such as AES. The administrators possess the secret keys to decrypt the encrypted data. One approach to reduce the damage caused by cloud compromises is to encrypt sensitive data using something like CryptDB [8], which is an encrypted database (EDB) equipped with functions of encrypted SQL operations such as join, sort, and addition.

EDBs are created by incorporating property-preserving encryption (PPE) into a relational database (RDB). PPE has the characteristic that certain plaintext properties are maintained when converting to ciphertext.The PPE includes a deterministic encryption (DTE) and an order-preservation encryption (OPE) [1–3] that preserves the order relation of a plaintext in a ciphertext as well. The searchable encryption (SE) technique is also utilized in the creation of EDBs.

CryptDB is the first EDB using PPE based on open source RDB such as MySQL or Postgres [8]. In CryptDB, DTE, OPE, and SE are used: the DTE for

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 35–47, 2018.
https://doi.org/10.1007/978-3-319-97916-8_3

encrypted equality match, the OPE typically for encrypted range search, and the SE for encrypted word search. CryptDB has been used by several organizations, and a few companies developed encrypted database inspired by CryptDB [19–21].

To evaluate its security, Naveed et al. [9] proposed a known plaintext attack method against the property preserved cryptosystem used in cryptDB. In their attack scenario, the attacker has an encrypted data set of a target DB that was encrypted by PPE and a plaintext data set having a similar background distribution. The attacker guesses the plaintext of the encrypted DB on the basis of the frequency, the order structure, and both sets of encrypted data.

In the present work, we focus on this attack method based on the linear sum assignment problem (LSAP) and its solution searching algorithm, called the Hungarian algorithm, and define a new optimization problem, namely, an order preserving assignment problem (OPAP), that considers the order structure. We show an exact solving algorithm for OPAP and use it to improve the method proposed by Naveed et al. We should point out that our new algorithm for OPAP is faster than the Hungarian algorithm, therefore not only improvement of the attack but also our new algorithm for OPAP are contribution of our work.

1.1 Related Works

Many works for PPE and security analysis have been performed on encryption schemes, including OPE and order-revealing encryption (ORE) [5–7]. They include a range search function that discloses the corresponding order relation for each query. The cloud ordinarily receives most queries, so the function and security level of OPE and ORE are essentially the same. Lewi et al. [11] demonstrated attacks against two-dimensional data using order information leakage from ORE, and Horst et al. showed an attack against the newly proposed OPE [4] by focusing on its configuration method [10].

CryptDB [8] has been proposed as an encrypted DB using PPE. Naveed et al. [9] proposed a known plaintext attack method against a property preserved cryptosystem used in cryptDB. They assumed an attacker who had a plaintext with a background distribution similar to the attack target and attacked using frequency information, order information, and cumulative density function information. Their attack method is mainly based on the LSAP and its solving algorithm (the Hungarian algorithm). This algorithm outputs an exact solution at $O(n^3)$ step, but because order is not considered, there are cases where plaintext guess results with collapsed order structure are output.

Grubbs et al. [17] formulated an optimization problem with the ordered structure, which is added to LSAP, and improved the attack proposed by Naveed et al. by using a classical algorithm that gives an approximate solution to it. In this paper, we formulate a similar optimization problem and propose an algorithm giving its exact solution.

1.2 Our Contribution

As stated above, the LSAP does not take the order structure into account, so we formulate a new optimization problem with it and give its exact solution algorithm, which is asymptotically optimized. The proposed method not only speeds up the inference attack but also improves the success probability of the attack.

We assume that database is encrypted by deterministic order preserving encryptions, which is the same scenario such as [9,17]. Database encrypted by probabilistic order preserving encryptions, which is proposed by Kerschbaum et al. [18], is out of scope in this paper.

Our contribution is summarized as follows.

1. To deal with attacks against ordered preservation encryption, we modify the LSAP and formulate a new optimization problem, OPAP, that considers order structure.
2. We describe an exact solution search algorithm using dynamic programming for OPAP. The number of steps in this algorithm is asymptotically optimal.
3. We experimentally demonstrate that using our search algorithm improves the success probability compared to Naveed's attacks for some data sets.
4. We provide a new evaluation method using Levenshtein distance for attacks against order preserving encryption and demonstrate that our proposed method is an improvement on the conventional method.

The reason for using an evaluation based on Levenshtein distance is as follows. For an order preserving encryption's ciphertext Enc(28), an attacker A guesses the plain text as 29 and an attacker B guesses it as 35. Both attackers A and B fail to make a correct guess, but attacker A makes a "better" guess than B. Our stance is that the difference from the true value of plaintext should be evaluated even if the estimated value deviates, so to this end, we provide an evaluation method for the speculation result of the attacker using Levenshtein distance.

2 Previous Research

2.1 Preliminary

We denote message space and ciphertext space by $\mathbb{M}$ and $\mathbb{C}$, respectively. We also denote the ciphertext of message sequence $\boldsymbol{m} = (m_1, \ldots, m_k)$ by $\boldsymbol{c} = (c_1, \ldots, c_k)$, where $m_i \in \mathbb{M}$ and $c_i \in \mathbb{C}$. The attacker applies an inference attack using both frequency information and order information derived from the ciphertext of OPE, which leaks order relation and equality: $\mathrm{Enc}(m_1) \odot \mathrm{Enc}(m_2)$ if $m_1 \odot m_2$ with an order notation $\odot$ such as $\odot \in \{<, =, >\}$.

Let $\mathbb{D}$ be a totally ordered set. We write data $\boldsymbol{d}$ as $\boldsymbol{d} = (d_1, \ldots, d_k)$ ($d_i \in \mathbb{D}$, $1 \leq i \leq k$). The histogram $\mathbf{Hist}(\boldsymbol{d}) = (h_1, \ldots, h_n)$ is on a $|\mathbb{D}|$-dimensional vector, where h_i is the frequency of the i-th element of the vector. For example, if we calculate the histogram of $\boldsymbol{d} = (1, 1, 2, 3, 5, 1, 3)$ in $\mathbb{D} = \{1, 2, 3, 4, 5\}$, $\mathbf{Hist}(\boldsymbol{d})$ is given by $(3, 1, 2, 0, 1)$.

Next, we define the cumulative distribution function (CDF) of $\boldsymbol{d}$. By defining $f_i := \sum_{j=1}^{i} h_j$, we have $\mathbf{CDF}(\boldsymbol{d}) = (f_1, \ldots, f_n)$. For example, if we calculate the CDF of $\boldsymbol{d} = (1, 1, 2, 3, 5, 1, 3)$ in $\mathbb{D} = \{1, 2, 3, 4, 5\}$, $\mathbf{CDF}(\boldsymbol{d})$ is $(3, 4, 6, 6, 7)$.

2.2 Inference Attack

In addition to the encrypted database most inference attack needs an auxiliary data, therefore we assume that our adversary has access to auxiliary data. Auxiliary data is standard in any practical adversarial model since the adversary can always consult public information sources to carry out the attack. Their success depends on how well-correlated the auxiliary data is with the encrypted database. The choice of auxiliary data is therefore an important consideration when evaluating an inference attack.

One of the most fundamental elements of an inference attack is the frequency analysis, which assigns encrypted data to auxiliary data in order of frequency. Numerical ordered data such as the age can be stored in EDB, so a census data including the age is a good candidate for auxiliary data (as in the scenario of the cumulative attack proposed by Naveed [9]).

Inference attacks on OPE can utilize both frequency and order information. The simplest attack using order information leaked from ciphertext is an attack called the sort attack, which assigns a ciphertext to a plaintext by order of plaintext. This attack always succeeds if all ciphertext are obtained as encrypted data. In other words, the success rate of this attack is high, but only if the given ciphertext covers most of the ciphertext space.

The attack is detailed below and works as follows. Given an ciphertext sequence $\boldsymbol{c}$ generated by OPE over $\mathbb{C}$ and auxiliary data $\boldsymbol{z}$ over $\mathbb{M}$, the attacker computes the histogram $\mathbf{Hist}(\boldsymbol{c})$ and $\mathbf{Hist}(\boldsymbol{z})$, and the $\mathbf{CDFs}(\boldsymbol{c})$ and $\mathbf{CDFs}(\boldsymbol{z})$, respectively. It then finds the permutation $\boldsymbol{X}$ that simultaneously matches both the sample frequencies and the CDFs as closely as possible. In order words, the cumulative attack recovers the plaintext by minimizing the following equation, which we call cost function.

$$\sum_{i=1}^{|\mathbb{M}|} \left(\left| \mathbf{Hist}(\boldsymbol{c})_i - \langle \boldsymbol{X}_i, \mathbf{Hist}(\boldsymbol{z}) \rangle \right|^2 + \left| \mathbf{CDF}(\boldsymbol{c})_i - \langle \boldsymbol{X}_i, \mathbf{CDF}(\boldsymbol{z}) \rangle \right|^2 \right)$$

Here, $\boldsymbol{X}_i$ denotes the vector at the i-th row of matrix X, which is an $n \times n$ matrix. There is only one 1 per row and column; the other components are 0. $\mathbf{Hist}(\boldsymbol{c})_i$ denotes the i-th component of the histogram, and $\langle \boldsymbol{X}_i, \mathbf{Hist}(\boldsymbol{z}) \rangle$ denotes the inner product of $\boldsymbol{X}_i$ and $\mathbf{Hist}(\boldsymbol{z})$. The histogram quantifies frequency information and the CDF quantifies order information. If the j-th component of $\boldsymbol{X}_i$ is 1, the i-th term of the cost function is given as

$$\left| \mathbf{Hist}(\boldsymbol{c})_i - \mathbf{Hist}(\boldsymbol{z})_j \right|^2 + \left| \mathbf{CDF}(\boldsymbol{c})_i - \mathbf{CDF}(\boldsymbol{z})_j \right|^2 .$$

This is the sum of the squared errors of the histogram and CDF when assigning the i-th smallest ciphertext to the j-th smallest plaintext. If the attacker once

obtains the permutation matrix $\boldsymbol{X}$ minimizing the cost function, he can obtain the assignment between ciphertext and plaintext, which minimizes errors of the histogram and CDF.

The cumulative attack is a minimization problem with variable matrix $\boldsymbol{X}$ such that

$$\underset{\boldsymbol{X}\in\mathbb{P}}{\operatorname{argmin}} \sum_{i=1}^{|\mathbb{M}|} \left(\left|\mathbf{Hist}\left(\boldsymbol{c}\right)_i - \langle \boldsymbol{X}_i, \mathbf{Hist}\left(\boldsymbol{z}\right)\rangle\right|^2 + \left|\mathbf{CDF}\left(\boldsymbol{c}\right)_i - \langle \boldsymbol{X}_i, \mathbf{CDF}\left(\boldsymbol{z}\right)\rangle\right|^2 \right), \quad (1)$$

where $\mathbb{P}$ is the set of the entire $n \times n$ permutation with $n = |\mathbb{C}|$.

In [9], Naveed et al. reduce the minimization of the cost function to the linear sum assignment problem (LSAP) and then use the Hungarian algorithm [12,13] to solve the LSAP. This attack runs in $O(n^3)$. The LSAP is formulated for $n \times n$ cost matrix $C = (c_{ij}), (0 \leq c_{ij})$, as

$$\begin{aligned} &\text{minimize} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}X_{ij} \\ &\text{subject to} \sum_{i=1}^{n} X_{ij} = 1, 1 \leq j \leq n \\ &\quad \sum_{j=1}^{n} X_{ij} = 1, 1 \leq i \leq n \\ &\quad X_{ij} \in \{0,1\}, 1 \leq i,j \leq n. \end{aligned}$$

The cumulative attack is reduced to LSAP by setting the cost matrix $C = (c_{ij})$, as

$$c_{ij} = \left|\mathbf{Hist}\left(\boldsymbol{c}\right)_i - \mathbf{Hist}\left(\boldsymbol{z}\right)_j\right|^2 + \left|\mathbf{CDF}\left(\boldsymbol{c}\right)_i - \mathbf{CDF}\left(\boldsymbol{z}\right)_j\right|^2.$$

To set **Hist**, let $\mathbb{C}'$ be a set of ciphertext that appears in encrypted data $\boldsymbol{c}$. If $|\mathbb{C}'| = |\mathbb{C}|$, the attacker recovers all encrypted data by sequentially assigning ciphertext to plaintext using only the order information. We consider only the case that $|\mathbb{C}'| := l < |\mathbb{C}|$. Note that the cost matrix is uniquely determined when the histogram is determined. If the histogram can be determined appropriately, the cost matrix will also determined. On the other hand, the attacker cannot know all the positions of ciphertext in $\mathbb{C}$ in advance. If the attacker knows the correct positions of all ciphertext, he can recover all encrypted data by sequentially assigning ciphertext to plaintext.

We assume that Naveed et al. consider 0s to be padded at the position containing lower frequency counts in auxiliary data in the histogram, although the details are not written in [9]. Consequently, if the auxiliary data has a property in which a larger value appears less frequently, the following setting of **Hist**,

$$\mathbf{Hist}\left(\boldsymbol{c}\right) = \left(h_1, \ldots, h_l, 0, \ldots, 0\right),$$

will be obtained. The performance heavily relies on where the 0s are padded.

2.3 Drawback of the Cumulative Attack

The cumulative attack has the following drawback.

The output may conflict with the order of plaintext due to lack of order constraints; therefore, the success rate is low if the amount of given ciphertext is small.

In addition to the above drawback there is other drawback. Suppose the attacker obtains the plaintext such that $c_j \to m_j$ and $c_i \to m_j$ for ciphertext $c_i < c_j$ and plaintext $m_j < m_i$ by the cumulative attack. This output contradicts the order relation, and it is always incorrect. The nature of the cumulative attack does not prohibit this kind of inconsistent output, which leads to a low success rate, especially when the amount of ciphertext is small. Our research aim is to develop a method to overcome this problem.

3 New Optimization Problem

We resolve the problem in the cumulative attack by introducing a new optimization problem in the place of LSAP. This problem is obtained by appending order constraints to LSAP, and it overcomes the issue where the order of output contradicts the order of correct plaintext. In our new optimization problem, the input matrix does not need to be a square matrix, so it is unnecessary to pad the histogram of encrypted data with $0 \ldots 0$.

In the cumulative attack, the assignment is obtained by solving the LSAP, while in our method, we reduce the assignment to an optimization problem with order constraint and find the X that minimizes the cost function.

3.1 Order Preserving Assignment Problem

We introduce an order constraint by the following equations:

$$\text{If } X_{ij} = 1 \text{ then } X_{i'j'} \neq 1 \, (i < i', j' < j) \tag{2}$$

$$\text{If } X_{ij} = 1 \text{ then } X_{i'j'} \neq 1 \, (i' < i, j < j') \tag{3}$$

Equation (2) represents a constraint that ciphertext smaller than c_i are not assigned to plaintext larger than m_j when ciphertext c_i is assigned to plaintext m_j. Equation (3) represents a constraint that ciphertext larger than c_i are not assigned to plaintext smaller than m_j when ciphertext c_i is assigned to plaintext m_j. We introduce an optimization problem obtained by appending the constraints of Eqs. (2) and (3) to LSAP. We call this an *order preserving assignment problem* (OPAP).

The OPAP is defined for $m \times n$ matrix $C = (c_{ij})$ and $(0 \leq c_{ij}, 1 \leq i \leq m, 1 \leq j \leq n)$, and is formulated as follows.

$$\begin{aligned}
&\text{minimize} \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} X_{ij} \\
&\text{subject to} \sum_{j=1}^{n} X_{ij} = 1, 1 \leq i \leq m \\
&\quad X_{ij} \in \{0, 1\}, 1 \leq i \leq m, 1 \leq j \leq n \\
&\quad \text{If } X_{ij} = 1 \text{ then } X_{i'j'} \neq 1 \, (i < i', j' < j) \\
&\quad \text{If } X_{ij} = 1 \text{ then } X_{i'j'} \neq 1 \, (i' < i, j < j')
\end{aligned}$$

3.2 Solving Order Preserving Assignment Problem

In this subsection, we propose an algorithm to solve OPAP. In a nutshell, the OPAP given for the entire cost matrix is reduced to the OPAP for the submatrix of the cost matrix. First, we give the algorithm for solving the OPAP.

Algorithm 1. Algorithm for Solving OPAP

```
1: function Optimize(C):
2: Input: m × n cost matrix C
3: Output: Solution of the order preserving assignment problem
4: if C has one row then
5:    OUTPUT argmin C
6: else if C is n × n square matrix then
7:    OUTPUT E_n
8: else
9:    A ← 0_{m×n}
10:   A[1 : m − 1][1 : n − 1] ←Optimize(C[1 : m − 1][1 : n − 1])
11:   A[m][n] ←1
12:   B ← 0_{m×n}
13:   B[1 : m − 1][1 : n] ←Optimize(C[1 : m − 1][1 : n])
14:   OUTPUT argmin_{X∈{A,B}} Σ_{i=1}^{n} Σ_{j=1}^{n} C_ij X_ij
15: end if
```

We first explain lines 4 to 7. If there is one row in the cost matrix, the minimum value of the row is the minimum assignment. If the cost matrix is square, the assignment of diagonal entries is the only assignment, since the assignment of diagonal entries is the smallest assignment.

Next, we explain the lines after 8. Suppose the cost matrix C is $m \times n$ matrix, where $m < n$. We denote the $i \times j$ submatrix of C by $C_{[1:i][1:j]}$ and the solution of the order preserving assignment problem by $X_{[1:i][1:j]}$. The solution A or B of the order preserving problem for C is then given as

$$A = \begin{pmatrix} X_{[1:m-1][1:n-1]} & 0 \\ 0 & 1 \end{pmatrix}, \tag{4}$$

$$B = \begin{pmatrix} X_{[1:m-1][1:n]} 0 \end{pmatrix}. \tag{5}$$

Our algorithm calculates the cost functions of A and B and finds an assignment with small values. This property then becomes the solution for the cost matrix C.

3.3 Toy Example for Solving OPAP

We explain how to solve the OPAP given the following cost matrix.

$$C = \begin{pmatrix} 12 & 11 & 13 & 17 & 19 & 5 \\ 8 & 9 & 4 & 7 & 12 & 5 \\ 7 & 8 & 6 & 5 & 4 & 2 \end{pmatrix}$$

$$C_{[1:2,1:3]} = \begin{pmatrix} \boxed{12} & \boxed{11} & 13 \\ 8 & \boxed{9} & \boxed{4} \end{pmatrix}.$$

We compare the minimum sum of $C_{[1:2,1:2]}$ and the minimum sum of $C_{[1:1,1:2]} + C_{[2,3]}$ for solving $C_{[1:2,1:3]}$. Since $C_{[1:2,1:2]}$ is a square matrix, the order preservation assignment only has diagonal entries. The assignment is then given by the diagonal entries. Since $C_{[1:1,1:2]}$ has one row, the minimum value of the elements is a solution. Therefore, we compare the values for two assignments (1, 1), (2, 2) and (1, 2), (2, 3) and set the smaller one as the solution of $C_{[1:2,1:3]}$, where (k, l) is an assignment of k rows and l columns. The minimum assignment is (1, 2), (2, 3) and the minimum value is 15.

Next, we obtain the minimum assignment of $C_{[1:2,1:4]}$:

$$C_{[1:2,1:4]} = \begin{pmatrix} 12 & \boxed{\boxed{11}} & 13 & 17 \\ 8 & 9 & \boxed{4} & \boxed{7} \end{pmatrix}.$$

This problem is solved by comparing the minimum sum of $C_{[1:2,1:3]}$ and the minimum sum of $C_{[1:1,1:3]} + C_{[2,4]}$. Note that we have a minimum assignment of $C_{[1:2,1:3]}$ in the above procedure. Since $C_{[1:1,1:3]}$ has one row, the minimum value of the elements is a solution. We then compare the two assignments, (1, 2), (2, 3) and (1, 2), (2, 4), to solve $C_{[1:2,1:4]}$. Therefore, the minimum assignment of $C_{[1:2,1:4]}$ is given by (1, 2), (2, 3). Similarly, solutions of $C_{[1:2,1:5]}$ are obtained as (1, 2) and (2, 3).

Next, we find the solution of $C_{[1:3,1:4]}$:

$$C_{[1:3,1:4]} = \begin{pmatrix} \boxed{12} & \boxed{11} & 13 & 17 \\ 8 & \boxed{9} & \boxed{4} & 7 \\ 7 & 8 & \boxed{6} & \boxed{5} \end{pmatrix}.$$

We compare the minimum sum of $C_{[1:3,1:3]}$ and the minimum sum of $C_{[1:2,1:3]} + C_{[3,4]}$. Since $C_{[1:3,1:3]}$ is a square matrix, the assignment of diagonal entries is the smallest. Also, $C_{[1:2,1:3]}$ has the minimum assignment (1, 2), (2, 3) that we solved in the above procedure. When we compare (1, 1), (2, 2), (3, 3) and (1, 2), (2, 3), (3, 4), we obtain (1, 2), (2, 3), (3, 4) as the minimum assignment of $C_{[1:3,1:4]}$. Since we already obtained the minimum assignment of $C_{[1:3,1:4]}$ and $C_{[1:2,1:4]}$, we have the minimum assignment of $C_{[1:3,1:5]}$. Solutions of C are calculated from the solutions of $C_{[1:3,1:5]}$ and $C_{[1:2,1:5]}$. That is, the solution of C is given by $(1, 2), (2, 3), (3, 6)$.

4 Experiments Using Datasets

In [9], Naveed et al. show that the cumulative attack can recover most of the encrypted data when the encrypted data are large enough. Here, we experiment on situations where recovery with the cumulative attack is difficult: namely, when the amount of ciphertext is small.

4.1 Datasets

In this experiment, we use the UCI Machine Learning Repository Adult Data Set [14], which includes 48,842 units of data. This dataset consists of 32,561 training sets and 16,281 test sets. We sample the auxiliary data from the training sets and sample the target data to be attacked from the test sets. We design two scenarios:

Scenario 1: This scenario, which is optimistic, has a lot of auxiliary data and little encrypted data (no. of encrypted data: 1,000, no. of auxiliary data: 32,561).

Scenario 2: This scenario, which is pessimistic, has less auxiliary data and encrypted data (no. of encrypted data: 1,000, no. of auxiliary data: 1,000).

In the case that a similar data set is public, the attacker has lots of auxiliary data but less encrypted data. In addition to the situation of public data sets, we design a situation with less auxiliary data, which treats closed data sets such as sensitive data. In [9], Naveed et al. consider scenarios using past accidental data leakages in which auxiliary data may be less.

We select *age column* on the Adult Data Set as a target. The age column is ranged from $[17, 18, \ldots, 90$, more than 90],then $|\mathbb{D}| = 75$.

We prepare 500 databases in accordance with this database setting, attack them, and evaluate the output. The evaluation of the attack in scenario 1 randomly samples 1,000 encrypted data from test sets and generates 500 databases. Moreover, we attack 500 databases by using training sets as auxiliary data. For the evaluation of the attack in scenario 2, 1,000 units of encrypted data are randomly sampled from the test sets and 1,000 units of auxiliary data are randomly sampled from the training sets, and we generate 500 databases. Finally, we evaluate each of the 500 databases and graph the overall results.

4.2 Evaluation Metrics

We evaluate the performance of the algorithms using two metrics. The first one is the same as [9], which is the rate at which encrypted data can be correctly recovered. The second one is the Levenshtein distance [16] between the output and the correct solution, which is considered the similarity between them. The Levenshtein distance between two strings is the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into another.

The reason we newly consider the Levenshtein distance is as follows. The evaluation metrics of [9] are considered only when the data is completely matched, but there are many situations where attacks are considered successful if correct results are obtained to a certain extent. For example, an attack can be considered effective in many cases even if the values are recovered differently one by one. We use the Levenshtein distance for measuring the performance in such cases.

4.3 Experimental Results for Scenario 1

Figure 1 compares the proposed method with the previous method based on the recovery rate of encrypted data divided by 10%. Unlike the previous method the number of databases the proposed method can recover more than 80% of is 400 out of 500.

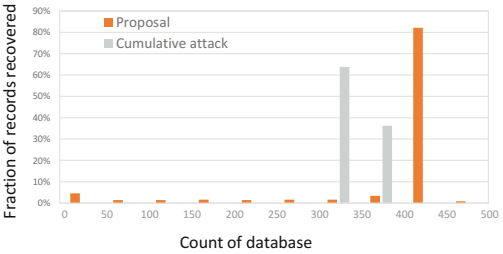


Fig. 1. Histogram of recovery rate.

Next, we discuss the evaluation based on Levenshtein distance. The average of the output and the correct result of the previous method is 14.65 and of the proposed method is 4.266. As shown in Fig. 2, the proposed method has an improved performance on the Levenshtein distance.

4.4 Experimental Results for Scenario 2

Next, we discuss the results of experiments where there are few encrypted data and auxiliary data. Figure 3 compares the recovery rate results for the proposed and previous methods. As shown here, the number of recoverable databases

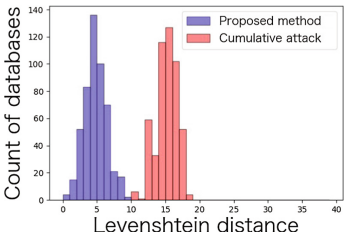


Fig. 2. Evaluation based on Levenshtein distance in Scenario 1.

of 70% or more and less than 90% in the proposed method is inferior to the previous method. This is because the previous method partially replaces the order, and the possibility that the output partially matches is larger than that of the proposed method. However, the proposed method shows better performance than the previous method for the recovery rate of 90% or more.

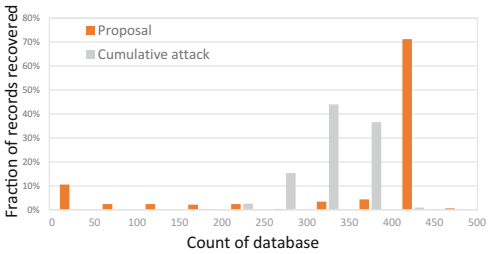


Fig. 3. Histogram of recovery rate.

Next, we compare the results using Levenshtein distance between the proposed and previous methods, as shown in Fig. 4. The average Levenshtein distance is 14.43 for the previous method and 5.14 for the proposed method. This demonstrates that the performance of the proposed method is significantly improved compared with the previous method.

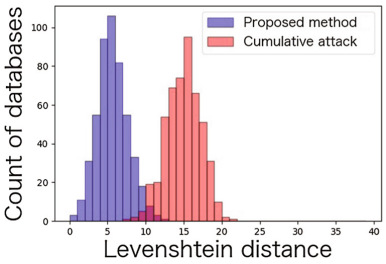


Fig. 4. Evaluation of Levenshtein distance in Scenario 2.

5 Conclusion

We proposed a new optimization problem and demonstrated an algorithm for solving it. This algorithm enabled us to improve the performance of the cumulative attack. Experimental results showed that the recovery rate of the proposed method exceeds that of the previous method. Additionally, the proposed method had a better performance than the previous method in many cases in the evaluation based on Levenshtein distance.

For future works, we plan to apply the proposed attack to database encrypted by probabilistic order preserving encryptions and evaluate the probability of recovering plaintext not only in theory but also in experiment.

Acknowledgments. This research was partially supported by JST CREST Grant Number JPMJCR1302, Japan.

References

1. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: Proceedings of the SIGMOD, pp. 563–574 (2004)
2. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Proceedings of the EUROCRYPT, pp. 224–241 (2009)
3. Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. In Proceedings of the CRYPTO, pp. 578–595 (2011)
4. Karras, P., Malhotra, S., Bhatt, R., Nikitin, A., Antyukhov, D., Idreos, S.: Adaptive indexing over encrypted numeric data. In Proceedings of the SIGMOD, pp. 171–183 (2016)
5. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. In: Proceedings of the EUROCRYPT, pp. 563–594 (2015)
6. Chenette, N., Lewi, K., Weis, S.A., Wu, D.J.: Practical Order-Revealing Encryption with Limited Leakage. In: Proceedings of the FSE, pp. 474–493 (2016)
7. Lewi, K., Wu, D.J.: Order-Revealing Encryption: New Constructions, Applications, and Lower Bounds. In: Proceedings of ACM CCS 2016, pp. 1167–1178 (2016)
8. Popa, R.A., Redeld, C., Zeldovich, N., Balakrishnan, H.: CryptDB: protecting confidentiality with encrypted query processing. In: Proceedings of the SOSP 2011, pp. 85–100 (2011)
9. Naveed, M., Kamara, S., Wright, C.V.: Inference attacks on property-preserving encrypted databases. In: Proceedings of the ACM CCS 2015, 644–655 (2015)
10. Horst, C., Kikuchi, R., Xagawa, K.: Cryptanalysis of comparable encryption in SIGMOD 2016. In: Proceedings of SIGMOD 2017, pp. 1069–1084 (2017)
11. Betül Durak, F., DuBuisson, T.M., Cash, D.: What else is revealed by order-revealing encryption? In: Proceedings of the ACM CCS, pp. 1155–1166 (2016)
12. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Res. Logistics Q. **2**, 83–87 (1955)
13. Munkres, J.: Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. **5**(1), 32–38 (1957)

14. UCI Machine Learning Repository: Adult Data Set. https://archive.ics.uci.edu/ml/datasets/adult
15. Privacy Rights Clearinghouse. Chronology of data breaches. http://www.privacyrights.org/data-breach
16. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1), 31–88 (2001)
17. Grubbs, P., Sekniqi, K., Bindschaedler, V., Naveed, M., Ristenpart, T.: Leakage-abuse attacks against order-revealing encryption. IEEE Symp. Secur. Priv. **2017**, 665–672 (2017)
18. Kerschbaum, F.: Frequency-hiding order-preserving encryption. ACM Conf. Comput. Commun. Secur. **2015**, 656–667 (2015)
19. Google, Encrypted BigQueqy. https://github.com/google/encrypted-bigquery-client
20. Sap AG, SEEED. https://www.sics.se/sites/default/files/pub/andreasschaad.pdf
21. Microsoft, Always Encrypted SQL Server. https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-database-engine?view=sql-server-2017

Implementation Security

Entropy Reduction for the Correlation-Enhanced Power Analysis Collision Attack

Andreas Wiemers and Dominik Klein(✉)

Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany
{andreas.wiemers,dominik.klein}@bsi.bund.de

Abstract. Side Channel Attacks are an important attack vector on secure AES implementations. The *Correlation-Enhanced Power Analysis Collision Attack* by Moradi et al. [MME10] is a powerful collision attack that exploits leakage caused by collisions in between S-Box computations of AES. The attack yields observations from which the AES key can be inferred. Due to noise, an insufficient number of collisions, or errors in the measurement setup, the attack does not find the correct AES key uniquely in practice, and it is unclear how to determine the key in such a scenario. Based on a theoretical analysis on how to quantify the remaining entropy, we derive a practical search algorithm. Both our theoretical analysis and practical experiments show that even in a setting with high noise or few available traces we can either successfully recover the full AES key or reduce its entropy significantly.

1 Introduction

Kocher's [Koc96] groundbreaking paper on side channel attacks has led both science and industry to focus on attacking and hardening their implementations [AARR03]. Due to its popularity and de-facto standard w.r.t. symmetric cryptographic algorithms, AES [BCO04,GMO01,KJJ99,Nat01,QS01] is of particular interest. Despite its theoretical cryptographic strength, a secure AES implementation that does not leak information about processed data remains to be a challenge. A popular counter-measure to minimize leakage about the AES key is *masking*. Different masking schemes exist, but the general idea of masking is that whenever secret data is about to enter critical stages of operation, some reversible operation that makes the data appear to be random is applied. Any cryptanalysis of intermediate data of the processing step is thus worthless. After leaving the critical stage of operation, the operation is reversed and the processed result can be used. Appropriate masking schemes can successfully prevent several attacks.

One particular class of attacks against AES are collision attacks. In collision attacks, one exploits the fact that sometimes leakage of the device can indicate that the same intermediate value has been processed during some critical stage of operation. By using this observation, one can gather information about

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 51–67, 2018.
https://doi.org/10.1007/978-3-319-97916-8_4

secret data and cryptanalyze the device. In particular attacks that detect internal collisions are of interest. This kind of attack method was originally applied to DES [LMV04,SWP03], but later applied to AES [Bog08,SLFP04] as well.

A very powerful kind of collision attack against AES was applied in [MME10], and later improved in [CFG+11], and reconsidered in [MS16]. The attack works by feeding data into a device in order to create collisions. A major important observation by [MME10] is that since the S-Box in AES is mathematically the same for every key byte (as opposed to i.e. DES), in most implementations the S-Box is also the same for every key byte. For example in a software implementation there is likely only one S-Box procedure that is called for every key byte, and in the case of a hardware implementation there is a single S-Box circuit that is used to process every key byte. This implies that for two same processed values, the resulting power consumption should be the same as well. Their idea is to create collisions such that this leakage *between* S-Box computations of different key byte positions is exploitable. This makes the attack very powerful—it is shown in [MME10] that a device with S-Boxes that are masked using the Canright S-Box implementation [Can05,CB08] can be broken with a reasonable amount of available traces. It is important to note here that in general the leakage of the device attacked in [MME10] was minimal, and in particular a typical state-of-the-art template attack [CRR03] was close to impossible to execute. In particular the amount of trace data needed to mount a successful attack was magnitudes lower for the correlation attack than for a template attack.

The attack gives some information about the correct AES key. However, the attack might not find the correct AES key uniquely in practice. There are several reasons for this: Noise, an insufficient number of collisions, errors in the measurement setup, or simply the device itself, i.e. the design and implementation of the cryptographic co-processor for a hardware implementation, or the processor design and execution flow in a software implementation. Moreover, it is not clear a priori how to find a set of key candidates that fit to the observations of the attack. A naive approach, i.e. enumerating all possible key candidates is computationally infeasible due to the large search space.

It is also unclear how to assess the leakage of the device; in particular it leaves open the question how many measurements (traces) are required to successfully mount the attack. Obviously, if the key uniquely identified for a certain amount of traces, this gives an upper bound. However what if less measurements are available?

In this paper we provide an algorithm to recover the AES key in the above scenario. The theoretical motivation of the algorithm is the basis for our analysis on how to quantify the remaining entropy, which can be used to assess the leakage of a device. Both our theoretical analysis and practical experiments show that even in a fuzzy setting with high noise or few available traces, we can either successfully recover the full AES key or reduce its entropy significantly.

This paper is structured as follows. In Sect. 2, we first briefly recall the attack by Moradi et al. as formulated in [MME10] and then introduce our algorithm in Sect. 3. For the algorithm we give a thorough theoretical justification in Sect. 4.

Then in Sect. 5 we analyze the success rate of the algorithm, i.e. its impact on the entropy of a vulnerable system w.r.t. its leakage, and give upper and lower bounds of the remaining entropy. Our theoretical findings are verified by providing experimental data in Sect. 6. The algorithm presented here can be seen as a particular form of a key-search algorithm. In Sect. 7 we show how our algorithm relates to known existing key search algorithms, and to the generalization of [MME10] described in [MS16]. Finally, we conclude our presentation in Sect. 8.

2 Correlation-Enhanced Power Analysis Collision Attack

Let $K_1, \ldots, K_{16}$ be the correct key which is used in the first round of an AES encryption. We denote by small letters $k_1, \ldots, k_{16}$ candidates for the key. We briefly recall the Correlation-Enhanced Power Analysis Collision Attack as described in [MME10].

During the measurement phase we record N power consumption traces of the first round of an AES-128 encryption[1]. These traces consists of 16 single S-Box computations. The measurement of each single S-Box computation is given as a vector of T numbers. We denote by $b_{i,w,t}$ this power consumption trace of a single S-Box computation i of a known plaintext $p_{w,i}$, $1 \le i \le 16$, $1 \le w \le N$, $1 \le t \le T$. As a first step we compute the average value $M_{i,\beta,t}$ over all w with $\beta = p_{w,i}$. Secondly, for any i, j and t we derive the empirical correlation coefficient $C_{i,j,\alpha,t}$ between $M_{i,\beta,t}$ and $M_{i,\beta\oplus\alpha,t}$ for any byte value α, where we treat β as a random variable uniformly distributed on all 256 byte values. At last, we set $c_{i,j}(\alpha)$ for the maximum of all $C_{i,j,\alpha,t}$, where t runs over all time points.

The idea of this approach is as follows: If the measurement $b_{i,w,t}$ is slightly dependent on the input byte $p_{w,i} \oplus K_i$ of the S-Box computation i, the average $M_{i,\beta,t}$ depends on $\beta \oplus K_i$ even more significantly. Now the input bytes $\beta \oplus K_i$ of S-Box i and $\beta \oplus K_j \oplus \alpha$ of S-Box j are the same for the choice $\alpha = K_i \oplus K_j$. Therefore, we can hope that the correlation $C_{i,j,\alpha,t}$ has—at least for some t—a significantly higher value for the correct choice $K_i \oplus K_j$ of α.

3 Recovering the AES Key

In this section we formulate our algorithm for computing candidates for the full AES key. We assume that we have given $120 \cdot 256$ values in the form

$$c_{i,j}(\alpha)$$

for $1 \le i < j \le 16$, where α runs over all byte values. If for each i, j the value $c_{i,j}(K_i \oplus K_j)$ is always the highest among all $c_{i,j}(\alpha)$, then it is easy to derive the full key. Here we are interested in the situation, where for each i, j, the value

[1] The attack can be extended to other key sizes in a straight-forward manner by targeting a second AES round [MME10].

$c_{i,j}(K_i \oplus K_j)$ has only a tendency of being large compared to other $c_{i,j}(\alpha)$ with $\alpha \neq K_i \oplus K_j$. The idea of our approach is to consider the ad-hoc evaluation function

$$B = \sum_{i<j} c_{i,j}(k_i \oplus k_j)$$

for any key candidate $(k_1, \cdots, k_{16})$ and choose the key candidate with the highest value in B. Since this is not feasible in a straightforward manner, we instead try to compute B via partial sums. To this end, we fix an integer W, resp. integers $g_2, \cdots, g_{16}$.

Algorithm 1. Recovering the AES Key

1: Set $k_1 = 0$, $S_1 = \{k_1\}$ and $B_1 = 0$.
2: **for** $s = 1, \ldots, 15$ **do**
3: **for** each key candidate $k_1, \ldots, k_s$ in S_s **do**
4: **for** each value of the next key bytes k_{s+1} **do**
5: compute the evaluation function

$$B_{s+1} = B_s((k_1, \ldots, k_s)) + \sum_{1 \leq i \leq s} c_{i,s+1}(k_i \oplus k_{s+1})$$

6: **end for**
7: **end for**
8: select subset of candidates $k_1, \ldots, k_s, k_{s+1}$ w.r.t. some criteria and store in S_{s+1}:
9: **Variant I**: Select W candidates $k_1, \ldots, k_s, k_{s+1}$ with largest B_{s+1}
10: **Variant II**: Select all $k_1, \ldots, k_s, k_{s+1}$ with $B_{s+1} \geq g_{s+1}$
11: **end for**
12: **return**

Remarks and Observations. We note some properties of Algorithm 1: Since B_s and B only depend on $\oplus$-sums of key bytes, we can choose one key byte as a fixed value. Here, we set $k_1 = 0$. The success probability of both variants of our algorithm for finding the correct key depends on the input parameters W, resp. $g_2, \cdots, g_{16}$. Obviously, if we choose $g_s = B_s((K_i \oplus K_j))$, Variant II of Algorithm 1 is guaranteed to output the correct key. However, in this case S_s might become too large to store in practice. The parameter W can be treated as a measure of the workload (i.e. the number of computational steps) of Variant I of Algorithm 1. Both variants of the algorithm assume a fixed order of key byte positions. The result of the algorithm depends on that assumed order of the key bytes. One can repeat the algorithm with different orders, though. As s grows, the order becomes less important. We investigate the effect of the order on the success of the algorithm in Sect. 6. The choice of the order could take into account the actual distribution of the values $c_{i,j}(\alpha)$. Those i, j with significantly high values in $c_{i,j}(\alpha)$ could be considered first. In a practical setting, visual inspection of $C_{i,j,\alpha,t}$ could give a hint, cf. for example Figs. 3a and b. In general

however, we are more interested in the situation where $c_{i,j}(K_i \oplus K_j)$ is not automatically the highest value among the $c_{i,j}(\alpha)$, but is only larger on average over all i, j.

4 Theoretical Justification

In this section, we give a justification of the evaluation function B. To this end, we treat $c_{i,j}(\alpha)$ for any i, j, α as a realization of a normally distributed random variable. Note that technically, this assumption is not correct. First, correlation values are bound in $[-1, 1]$ and thus cannot be normally distributed. Second, we do not consider correlation values directly, but instead $c_{i,j}(\alpha)$ is a maximum of $C_{i,j,\alpha,t}$ over several time points t. We can easily force $c_{i,j}(\alpha)$ to be normally distributed by fixing one single time-point, and apply Fisher's z transformation on the correlation values. In practice however, we can see that the correlations $C_{i,j,\alpha,t}$ for wrong α (cf. Figs. 3a, b) lie within $[-0.25, 0.25]$. Fisher's z transformation is almost identical for values in that interval, so that we do not consider this transformation in our case. In addition we observe (cf. Figs. 5a, b) that the distribution of $c_{i,j}(\alpha)$ is not perfectly symmetrical, but minimally skewed. This can theoretically be justified by the fact that $c_{i,j}(\alpha)$ is a maximum of $C_{i,j,\alpha,t}$ over several time points t. All in all, the deviation of the distribution $c_{i,j}(\alpha)$ from a normally distributed random variable is very small and is thus neglected in the following.

We assume the easiest scenario: For any i, j, α with $\alpha \neq K_i \oplus K_j$ the means and the standard deviations are equal and are denoted by a, resp. σ. Furthermore, for the correct $\alpha = K_i \oplus K_j$ the means are equal and are denoted by b and in addition, the standard deviations are equal to σ. For any key candidate $k = (k_1, \cdots, k_{16})$ we can check whether for all key candidates $\tilde{k}$

$$c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) \approx a, \text{ if } \tilde{k}_i \oplus \tilde{k}_j = k_i \oplus k_j$$
$$c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) \approx b, \text{ if } \tilde{k}_i \oplus \tilde{k}_j \neq k_i \oplus k_j$$

As a likelihood measure for any key candidate we seek a function in the single probability density functions as

$$\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) - a)^2}{2\sigma^2}\right), \text{ resp.}$$
$$\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) - b)^2}{2\sigma^2}\right)$$

The cumulative probability density function of two *independent* random variables is just the product of the single probability density functions. Therefore, we are led to use as an evaluation function the product over all single probability density functions. Taking logarithms we get

$$\sum_{\tilde{k}} \left[\sum_{\substack{i<j,\\ \tilde{k}_i \oplus \tilde{k}_j = k_i \oplus k_j}} (c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) - a)^2 + \sum_{\substack{i<j,\\ \tilde{k}_i \oplus \tilde{k}_j \neq k_i \oplus k_j}} (c_{i,j}(\tilde{k}_i \oplus \tilde{k}_j) - b)^2 \right]$$

An equivalent evaluation function is therefore

$$\sum_{i<j} c_{i,j}(k_i \oplus k_j).$$

5 Success Rate of the Algorithm (Variant II)

In this section we want to give theoretical estimates of the success rate and workload for the second variant of the algorithm. The purpose of this section is to find relations between those theoretical estimates and basic properties of the distributions of $c_{i,j}(\alpha)$. To make the derivation as simple as possible, we restrict ourselves to the scenario in the last section, i.e. $c_{i,j}(\alpha)$ for any i, j, α is treated as a realization of a normally distributed random variable with mean a, resp. b, and standard deviation σ. Furthermore, for any key candidate the evaluation function

$$B_s = \sum_{i<j\leq s} c_{i,j}(k_i \oplus k_j)$$

is considered as a sum of *independent* random variables. Therefore, B_s is a normally distributed random variable. For a randomly chosen key candidate we have the expectation value

$$\mathrm{E}(c_{i,j}(k_i \oplus k_j)) = \frac{255}{256}a + \frac{1}{256}b \approx a$$

since b is assumed to be only slightly larger than a. Therefore, the mean and standard deviation of B_s are $\binom{s}{2}a$, resp. $\sqrt{\binom{s}{2}}\sigma$. For the correct key, $B_s((K_1, \ldots, K_s))$ is normally distributed with mean $\binom{s}{2}b$.

For having B_s near to its mean value, we want to avoid small values in $\binom{s}{2}$. In practice, we set in Variant II of the algorithm $g_s = -\infty$ for $s \leq 4$. For the ease of presentation we want to assume

$$B_s((K_1, \ldots, K_s)) \geq \binom{s}{2} b \text{ for } s \geq 5$$

Therefore, we set here $g_s = \binom{s}{2}b$.

5.1 An Upper Bound of the Remaining Entropy

Variant II of the algorithm only finds key candidates for which $B_s \geq g_s$ for *all* s, $5 \leq s \leq 16$. In every step, the set S_s is a subset of all key candidates for

which the condition $B_s \geq g_s$ is fulfilled. The size A_s of this larger set can be approximated by

$$\#S_s \leq A_s = 2^{(s-1)8}\,\mathbf{P}\left(B_s \geq \binom{s}{2} b\right)$$

and $\log_2(A_{16})$ is an upper bound for the remaining entropy.[2] $A_{16} \approx 1$ means that the correct key has been found more or less uniquely, and $\max_s A_s$ is an upper bound for the workload of variant II of Algorithm 1. The inequality of integrals

$$\int_x^\infty e^{-t^2/2}dt \leq \int_x^\infty \frac{t}{x}e^{-t^2/2}dt = \frac{1}{x}e^{-x^2/2}$$

can be used to give an upper bound for the standardized normal distribution $\mathcal{N}_{0,1}$:

$$\mathcal{N}_{0,1}(x,\infty) \leq \frac{1}{x\sqrt{2\pi}}e^{-x^2/2}$$

We set

$$\tau = \frac{b-a}{\sigma}$$

and derive

$$\begin{aligned} A_s &= 2^{(s-1)8}\,\mathbf{P}\left(\frac{B_s - \binom{s}{2}a}{\sigma\sqrt{\binom{s}{2}}} \geq \tau\sqrt{\binom{s}{2}}\right) \\ &\leq 2^{(s-1)8}\frac{1}{\tau\sqrt{2\pi\binom{s}{2}}}e^{-\frac{1}{2}\binom{s}{2}\tau^2} = \frac{1}{\tau\sqrt{2\pi\binom{s}{2}}}2^{(s-1)8-\frac{1}{2\ln(2)}\binom{s}{2}\tau^2} \end{aligned}$$

This approximation of $\log_2(A_s)$ has roughly the form of a parabola in s. The condition $A_{16} \approx 1$ corresponds to the equation

$$\tau = \frac{b-a}{\sigma} \approx \sqrt{2\ln(2)} \approx 1.2$$

Some results are provided in Table 1. This can be interpreted in that we can expect that for $\frac{b-a}{\sigma} \geq 1$, Variant II of Algorithm 1 is successful with workload $\leq 2^{36}$ and remaining entropy ≤ 29.

5.2 A Lower Bound of the Remaining Entropy

We want to analyze variant II of Algorithm 1 step by step. We expect that the size of $\#S_{s+1}$ can be approximated by a *conditional* probability of the form

$$\begin{aligned} \#S_{s+1} \approx \#S_s 2^8 \mathbf{P}\Bigg(B_{s+1} \geq \binom{s+1}{2} b \mid B_s \geq \binom{s}{2} b, \\ B_{s-1} \geq \binom{s-1}{2} b, \ldots, B_5 \geq \binom{5}{2} b\Bigg) \end{aligned}$$

[2] Since one key byte cannot be determined by the algorithm, the accurate remaining entropy is more properly $\log_2(A_{16}) + 8$.

Table 1. Upper bounds for the remaining entropy

$\frac{b-a}{\sigma}$	$\log_2(A_{16})$	$\log_2(A_4)$	$\max_{s\geq 5}\log_2(A_s)$
1.4	0	24	15
1.2	0	24	23
1.1	10	24	29
1.0	29	24	36
0.9	45	24	46

Table 2. Bounds for the remaining entropy

$\tau = \frac{b-a}{\sigma}$	$\log_2(A_{16})$	$\log_2(A_4)$	$\max_{s\geq 5}\log_2(A_s)$	Lower bound of $\log_2(\#S_{16})$	Lower bound of $\max_{s\geq 5}\log_2(\#S_s)$
1.4	0	24	15	0	14
1.2	0	24	23	0	21
1.1	10	24	29	2	26
1.0	29	24	36	21	32
0.9	45	24	46	37	41

Note that

$$X_s = \frac{B_s - \binom{s}{2}a}{\sigma}$$

is the sum of $\binom{s}{2}$ $\mathcal{N}_{0,1}$-distributed independent random variables. Therefore, X_s represents a Gaussian random walk. We write the conditional probability in the form

$$\mathbf{P}\left(B_{s+1} \geq \binom{s+1}{2}b \mid B_s \geq \binom{s}{2}b, B_{s-1} \geq \binom{s-1}{2}b, \ldots, B_5 \geq \binom{5}{2}b\right)$$
$$= \mathbf{P}\left(X_{s+1} \geq \binom{s+1}{2}\tau \mid X_s \geq \binom{s}{2}\tau, X_{s-1} \geq \binom{s-1}{2}\tau, \ldots, X_5 \geq \binom{5}{2}\tau\right)$$

The probability

$$\mathbf{P}\left(X_{s+1} \geq \binom{s+1}{2}\tau, X_s \geq \binom{s}{2}\tau, X_{s-1} \geq \binom{s-1}{2}\tau, \ldots, X_5 \geq \binom{5}{2}\tau\right)$$

can be interpreted as the probability of a Gaussian random walk with at least linear growth at special steps. We get a lower bound of the conditional probability if we omit the conditions on all $s' < s$.

$$\#S_{s+1} \geq V_s 2^8 \mathbf{P}\left(B_s + \sum_{1\leq i\leq s} c_{i,s+1}(k_i \oplus k_{s+1}) \geq \binom{s+1}{2}b \mid B_s \geq \binom{s}{2}b\right)$$

for $s \geq 5$ and $\#S_5 = A_5$. Since $\binom{5}{2} = 10$, we use for $\#S_5 = A_5$ the formula

$$\#S_5 = 2^{32}\mathcal{N}_{0,1}\left(\sqrt{10}\frac{b-a}{\sigma}, \infty\right)$$

The probability $\mathbf{P}\left(B_s + \sum_{1\le i\le s} c_{i,s+1}(k_i \oplus k_{s+1}) \ge \binom{s+1}{2}b \mid B_s \ge \binom{s}{2}b\right)$ only depends on s and $\tau = \frac{b-a}{\sigma}$. This probability and therefore all lower bounds of $\#S_{s+1}$ can be calculated numerically. Table 2 extends Table 1 above. We can expect that the remaining entropy and the workload of variant II of Algorithm 1 are within the limits of this table.

5.3 Probability of the Event $B_s(K_1, \ldots, K_s)) \ge \binom{s}{2}b$

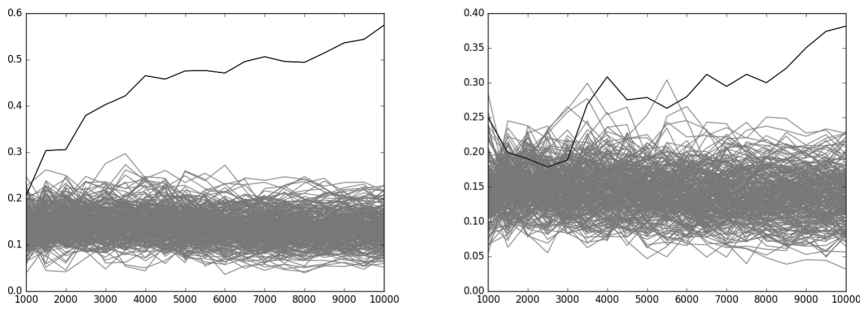
We consider the event

$$B_s((K_1, \ldots, K_s)) \ge \binom{s}{2}b \text{ for } all\ s = 16, 15, \ldots, 5$$

Note, that the probability of this event does not depend on b and σ, it is just a real number. On first sight, one could believe that the probability of this event is 2^{-12}. But since $B_{s-1}((K_1, \ldots, K_{s-1}))$ is a subsum of $B_s((K_1, \ldots, K_s))$, the probability of $B_s(K_1, \ldots, K_s)) \ge \binom{s}{2}b$ is larger than $\frac{1}{2}$ if we already know that $B_{s-1}((K_1, \ldots, K_{s-1})) \ge \binom{s-1}{2}b$. We compute an approximation of this probability by a simulation of normally distributed random variables. We get

$$\mathbf{P}\left(B_s((K_1, \ldots, K_s)) \ge \binom{s}{2}b \text{ for } all\ s = 16, 15, \ldots, 5\right) \approx 0.15.$$

To this end, the assumption $B_s(K_1, \ldots, K_s)) \ge \binom{s}{2}b$ for all s is not too restrictive.



(a) $c_{0,1}(\alpha)$ (b) $c_{0,9}(\alpha)$

Fig. 1. Correlation of $c_{i,j}(\alpha)$ vs # of traces. The correct value of α is shown in black.

6 Experiments

Experimental Setup. Our setup consists of an AES-128 based software implementation running on an Atmel ATMEGA328P-PU. The S-Boxes are realized as lookup tables and stored in the program memory of the ATMEGA. The S-Boxes are masked using the method presented in [AG01]. This masking is known

to be not leakage-free, and as shown in [CFG+11,MME10], also Canright S-Boxes [Can05] are susceptible to correlation attacks. Hence we anticipate that our results are representative. Moreover, the masking used [AG01] is straightforward to implement. The ATMEGA was setup on a custom prototype board, and powered by a lab-grade power supply at 3.3 Volt. We used a LeCroy HDO6104 to record the power consumption of the ATMEGA with a resistor against ground. We recorded N traces of AES-128 encryption. The plaintext was chosen at random, and we attacked the masked_subbytes procedure of the first AES round (Fig. 2).

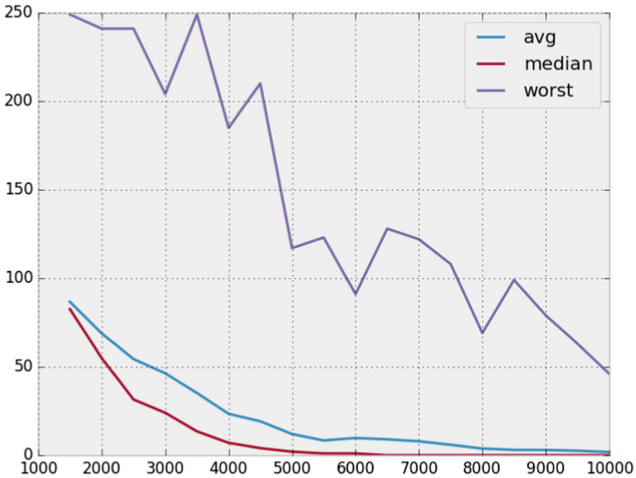


Fig. 2. Number of traces vs. ranking positions of $c_{i,j}(\alpha)$ for correct α.

Practical Results. We recorded 10000 traces with randomly generated plaintext values. Our implementation follows closely [MME10] and we compute the correlation w.r.t. each possible value of one $C_{i,j,\alpha,t}$. Figure 3a shows the resulting correlation of $C_{0,1,\alpha,t}$ for one AES round, i.e. in our setup $t = 0, ..., 25809$. Correlation peaks at the end of the S-Box for the correct value are clearly visible. However a correlation peak is not apparent for every pair of key byte positions i, j; for example considering the correlation values $C_{9,11,\alpha,t}$ as depicted in Fig. 3b, no clear peak is observable for the correct value α. Nevertheless when ranking all possible values $c_{9,11}(\alpha)$, the correct value is still at position 18. Fewer traces result in less collisions and more noise, and the rankings become more fuzzy. To give two examples, the rankings for the correct value of α for 2500 and 10000 traces are as shown in Fig. 4a and b. If more traces are available, the correlation values $c_{i,j}(\alpha)$ for the correct value α become very distinct from those for incorrect values of α, as illustrated in Fig. 1a and b. For $N = 10000$, the correct value

Table 3. Full key recovery for $W = 1500$ using Algorithm 1 (Variant I).

	1500 ... 3500	4000	4500	5000	5500	6000	6500
$1 \rightarrow 16$	–	✓	✓	✓	✓	✓	✓
$16 \rightarrow 1$	–	–	–	–	–	–	–
random	–	1/5	3/5	3/5	3/5	3/5	3/5
	7000	7500	8000	8500	9000	9500	10000
$1 \rightarrow 16$	–	✓	✓	✓	✓	✓	✓
$16 \rightarrow 1$	–	–	–	–	–	–	–
random	3/5	4/5	3/5	4/5	5/5	5/5	5/5

of α often shows on rank 1, but there are some outliers. For $N = 2500$, there are few rankings with position 1.

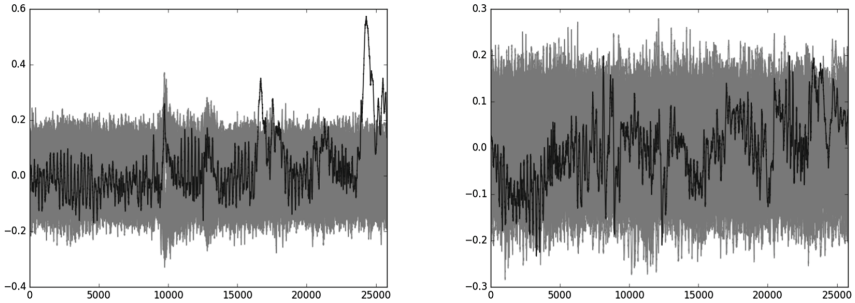
For Variant II of Algorithm 1 one needs to choose appropriate input values $g_2, \cdots, g_{16}$. This requires prior knowledge about the quality of the rankings $c_{i,j}(\alpha)$. If such knowledge is not available, appropriate values could also be estimated by manual analysis of the rankings $c_{i,j}(\alpha)$ and/or by visual inspection of $C_{i,j,\alpha,t}$. For example the comparison of Fig. 3a and b indicates that for $c_{0,1}(\alpha)$ the ranking for the correct value of α is very likely at one of the top positions, whereas for $c_{9,11}(\alpha)$ this is likely not the case. On the other hand, Variant I of Algorithm 1 requires no prior knowledge at all. Also, if Variant I succeeds, we can always choose the bound g_{s+1} in step s for Variant II in such a way that the choice of Variant I is simulated. This is why we have here chosen to implement Variant I of the algorithm.

As mentioned in Sect. 3, the success of the algorithm depends on the order in which the next key byte position is chosen, the parameter W of candidates that are kept in each iteration, and of course the number of available traces N. As mentioned in Sect. 2, in particular the order of choosing the next key byte position is important, as the next example illustrates:

Example 1. For simplicity, suppose we have only a key consisting of three bytes, and suppose one key byte can take only value 0 or 1. Let $c_{i,j}(\alpha)$ be as follows:

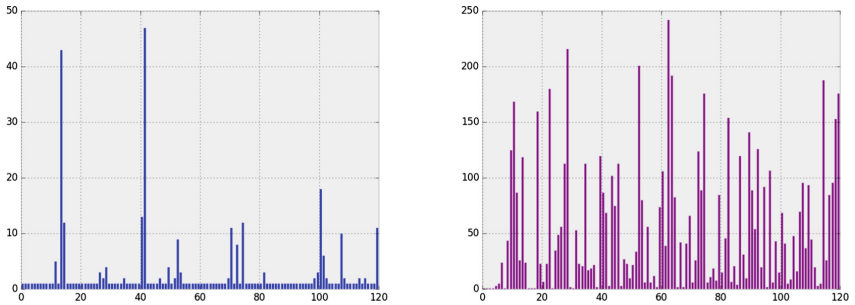
$$c_{0,1}(0) = 0.4 \qquad c_{0,1}(1) = 0.1 \qquad c_{1,2}(0) = 0.4$$
$$c_{1,2}(1) = 0.1 \qquad c_{0,2}(0) = 0.1 \qquad c_{0,2}(1) = 0.8$$

Suppose that we set $W = 2$. Assume the order $0 < 1 < 2$. We start with $S_1 = \{0, 1\}$. Since $c_{0,1}(0) > c_{0,1}(1)$ and $0 \oplus 0 = 0$ as well as $1 \oplus 1 = 0$, we yield the set $S_2 = \{00, 11\}$. Algorithm 1 terminates with $S_3 = \{001, 110\}$. On the other hand, it is not difficult to verify that for the key byte order $2 < 1 < 0$, Algorithm 1 terminates with the set $S_3 = \{100, 011\}$. Note also that Algorithm 1 is nondeterministic in general: If we set $W = 1$ in the second step during the run with order $0 < 1 < 2$, we have $B((00)) = B((11)) = 0.4$, and it is open which partial key to keep.



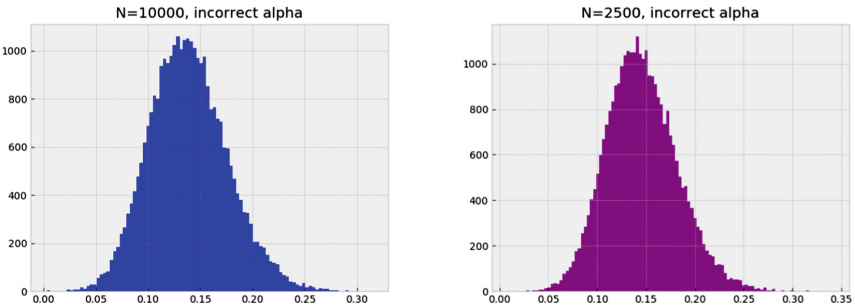
(a) $C_{0,1,\alpha,t}$ (b) $C_{9,11,\alpha,t}$

Fig. 3. Correlation $C_{i,j,\alpha,t}$ for each timepoint t within one trace. The correct value of α is denoted in black.



(a) $N = 10000$ (b) $N = 2500$

Fig. 4. Ranking positions for correct α.



(a) $N = 10000$ (b) $N = 2500$

Fig. 5. Distribution of $c_{i,j}(\alpha)$ for incorrect values of α.

We executed Algorithm 1 for $N = 1500$ up to $N = 10000$ in steps of 500 traces, and for both $W = 1500$ and $W = 10000$. As for the dependency on the order, we considered the natural order of starting at key byte position 1 and moving upward to position 16 (denoted by $1 \rightarrow 16$ in the following), the reverse order of starting at position 16 and moving down to 1 (denoted by $16 \rightarrow 1$), and on five randomly chosen orders for each value of N. Tables 3 and 4[3] show results for the full recovery of the key for $N = 1500...10000$ and the various orders. As one can see, the full key is in the computed set with good probability if at least 4000 traces are available. The probability can be increased, if a larger parameter $W = 10000$ is chosen. For less than 4000 traces, Table 5 shows the maximum number of correctly recovered key bytes in the computed set. For example, when choosing the order $1 \rightarrow 16$, then in the case of 2500 traces, the computed set contains a key where 12 key bytes are correctly identified. In other words, the entropy is significantly reduced, and it is not difficult to devise an algorithm that exploits the fact that one can assume that a certain amount of key byte position are correctly identified.

Table 4. Full key recovery for $W = 10000$ using Algorithm 1 (Variant I).

	1500 ... 3500	4000	4500	5000	5500	6000	6500
$1 \rightarrow 16$	–	✓	✓	✓	✓	✓	✓
$16 \rightarrow 1$	–	–	–	–	✓	✓	✓
random	–	1/5	3/5	3/5	3/5	3/5	3/5
	7000	7500	8000	8500	9000	9500	10000
$1 \rightarrow 16$	–	✓	✓	✓	✓	✓	✓
$16 \rightarrow 1$	✓	✓	–	✓	✓	✓	✓
random	3/5	4/5	3/5	4/5	5/5	5/5	3/5

In order to further interpret these results, we investigated the distribution $c_{i,j}$ in α for all (i, j). For each (i, j) we computed the expected value as well the standard deviation, which were very similar. Figure 5a and b show histograms of all $255 \cdot 120$ $c_{i,j}$ with incorrect value α for $N = 10000$ and $N = 2500$, respectively. Expected value and standard deviation are 0.139 ± 0.0374 for $N = 10000$ and 0.146 ± 0.0363 for $N = 2500$. As derived above, we expect a theoretical bound $\frac{b-a}{\sigma} \approx \sqrt{2\ln(2)} \approx 1.2$. This is the smallest value b, for which we can expect that the evaluation function B succeeds. For our experimental data we yield $\frac{b-a}{\sigma} = \frac{0.336-0.139}{0.0374} \approx 5.3$ for $N = 10000$ and $\frac{b-a}{\sigma} = \frac{0.195-0.146}{0.0363} \approx 1.3$ for $N = 2500$. Apparently, the parameters for $N = 2500$ are very close to the theoretical threshold. This fits with our theoretical observations in previous sections and the experimental data. Aside from the above results, we also experimentally

[3] Note that the miss for $N = 8000$ and $16 \rightarrow 1$ in Table 4 is precisely due to the nondeterministic behavior of Algorithm 1, as illustrated in Example 1.

verified our findings using a hardened security controller with a hardware based AES implementation, and the results were comparable. However, we refrain from providing more specific data here.

7 Related Work

Suppose we are given a key $K_1, ..., K_n$ with n independent subkeys, and a side channel attack yields for each K_i ranks for possible key values. Using these ranks, one can define a probability measurement function for each subkey. The *key enumeration problem* [VCGRS13] is then to enumerate complete keys from the most probable to the least probable one, in order to find the key of a device as quickly as possible. A direct solution is to enumerate all possible keys, compute their probability (multiplying the probabilities of the subkeys) and then sort all keys according to their probability. As this is infeasible in practice, more efficient algorithms have been proposed [VCGRS13].

In our scenario, we do not have ranks (resp. probability measurement functions) for all sixteen K_i. Rather, we obtain 120 ranks for each $K_i \oplus K_j$ for $1 \leq i, j \leq 16$. Our task is to reconstruct the key from these values. Algorithms that solve the key enumeration problem are thus not directly applicable. One naive way to still apply existing key search algorithms in our scenario would be to (1) select a number of $K_i \oplus K_j$ with their ranks which have a low remaining entropy, i.e. select $K_1 \oplus K_2, K_2 \oplus K_3, \ldots, K_{15} \oplus K_{16}$. (2) recover $K_1 \oplus K_2, K_2 \oplus K_3, \ldots, K_{15} \oplus K_{16}$ using an efficient key enumeration algorithm, and for each candidate (3) search among the remaining 256 possibilities for the correct key.

Note that it is not clear how to effectively select $K_i \oplus K_j$. One could for example simply take $K_1 \oplus K_2, K_2 \oplus K_3, \ldots, K_{15} \oplus K_{16}$. How to assess the leakage then? The problem of *key ranking* [GGP+15, MOOS15, MMOS16] is concerned with quantifying the time complexity required to brute force a key given the leakages, especially if the number of keys to enumerate is beyond practical computational limits. Applying e.g. the algorithm of [MMOS16] for our test set of traces with $N = 4000$ when we interpret $K_1 \oplus K_2, K_2 \oplus K_3, \ldots, K_{15} \oplus K_{16}$ as our key, the estimated rank is approximately 2^{50}. However with our algorithm we are able to recover the actual AES key within less than two minutes (cf. Table 3). Directly recovering the key (and not $K_i \oplus K_j$) by making use of all $K_i \oplus K_j$ was the motivation for our algorithm. We anticipate however, that by either employing voting techniques [Bog08] or by a more sophisticated choice of $K_i \oplus K_j$ and repeating steps (1)–(3) one could obtain similar results as with our algorithm, but we did not further pursue this direction.

In [MS16] the attack of [MME10] is generalized into *Moments-Correlating Collision DPA*. There, moments are correlated with samples rather than moments with moments, as done in the original attack. Using that, they derive a measure N_{sr} (Eq. 1 in [MS16]) that provides a metric to quantify the number of measurements needed to perform a key recovery with a given success rate. Note however that this measure quantifies the leakage w.r.t. $K_i \oplus K_j$. As seen

Table 5. Partial key recovery for $W = 1500$ using Algorithm 1 (Variant I).

	1500	2000	2500	3000	3500
$1 \rightarrow 16$	3	5	12	13	14
$16 \rightarrow 1$	2	2	4	1	5
random #1	1	5	3	7	9
random #2	1	5	2	5	14
random #3	2	3	5	12	5
random #4	2	3	5	7	8
random #5	2	2	1	8	12

in the experiments, the leakage can be quite unsteady, i.e. the correct value of α is in the lower ranks for some i, j, but for other combinations the rank is much higher. Here we provide an algorithm that still works in this scenario and reveal the correct key, and our measure τ thus gives a indicates on whether the key can be recovered or not.

8 Conclusion and Future Work

We have shown how to reduce the remaining key entropy of the attack introduced by [MME10] by providing a practical, easy-to-implement algorithm. Our theoretical analysis shows that this algorithm exploits the leakage in a natural way. Moreover, we provide a way to assess the leakage of a device w.r.t. the attack, which could be used e.g. in a Common Criteria security evaluation. Our practical evaluation supports the theoretical analysis. In particular we show that using our algorithm, a full recovery of the AES key is possible with only few available traces. That is, the key can be recovered in a setting where no visual clues w.r.t. the correct ranking are available and the attack as described by [MME10] would not have been applicable. A practical comparison with the metric derived in [MS16] remains future work.

Acknowledgements. We would like to thank Sven Freud for creating the circuit board for power analysis, and Tobias Senger for his help with implementing the masking scheme. We also thank the anonymous reviewers for their helpful comments.

References

[AARR03] Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM side—channel(s). In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_4

[AG01] Akkar, M.-L., Giraud, C.: An implementation of DES and AES, secure against some attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44709-1_26

[BCO04] Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2

[Bog08] Bogdanov, A.: Multiple-differential side-channel collision attacks on AES. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 30–44. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85053-3_3

[Can05] Canright, D.: A very compact S-Box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_32

[CB08] Canright, D., Batina, L.: A very compact "Perfectly Masked" S-box for AES. In: Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 446–459. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68914-0_27

[CFG+11] Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Improved collision-correlation power analysis on first order protected AES. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 49–62. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_4

[CRR03] Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3

[GGP+15] Glowacz, C., Grosso, V., Poussier, R., Schüth, J., Standaert, F.-X.: Simpler and more efficient rank estimation for side-channel security assessment. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 117–129. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48116-5_6

[GMO01] Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44709-1_21

[KJJ99] Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25

[Koc96] Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9

[LMV04] Ledig, H., Muller, F., Valette, F.: Enhancing collision attacks. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 176–190. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_13

[MME10] Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-enhanced power analysis collision attack. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_9

[MMOS16] Martin, D.P., Mather, L., Oswald, E., Stam, M.: Characterisation and estimation of the key rank distribution in the context of side channel evaluations. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 548–572. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_20

[MOOS15] Martin, D.P., O'Connell, J.F., Oswald, E., Stam, M.: Counting Keys in Parallel After a Side Channel Attack. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 313–337. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_13

[MS16] Moradi, A., Standaert, F.: Moments-correlating DPA. In: Proceedings of 2016 TIS SEC Workshop, pp. 5–15 (2016)

[Nat01] National Institute of Standards and Technology: FIPS PUB 197. Advanced Encryption Standard, Technical report (2001)

[QS01] Quisquater, J.-J., Samyde, D.: ElectroMagnetic Analysis (EMA): measures and counter-measures for smart cards. In: Attali, I., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45418-7_17

[SLFP04] Schramm, K., Leander, G., Felke, P., Paar, C.: A collision-attack on AES. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 163–175. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_12

[SWP03] Schramm, K., Wollinger, T., Paar, C.: A new class of collision attacks and its application to DES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 206–222. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39887-5_16

[VCGRS13] Veyrat-Charvillon, N., Gérard, B., Renauld, M., Standaert, F.-X.: An optimal key enumeration algorithm and its application to side-channel attacks. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 390–406. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35999-6_25

Safe Trans Loader: Mitigation and Prevention of Memory Corruption Attacks for Released Binaries

Takamichi Saito[1(✉)], Masahiro Yokoyama[1], Shota Sugawara[1], and Kuniyasu Suzaki[2]

[1] Meiji University, Kawasaki, Kanagawa, Japan
saito@cs.meiji.ac.jp
[2] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract. A variety of countermeasures against memory corruption attacks have been proposed to implement within compilers, linkers, operating systems, and libraries. However, according to our survey, a certain number of executable binaries in Linux distributions are not protected by the countermeasures, even when the countermeasures are applied to these binaries. Further, the countermeasures have some problems including the way of application, the scope of attacks, and the runtime overhead. For example, some require source code or need to update the kernel or specific libraries. These requirements are not acceptable for everyone. In this paper, we propose an application-level loader called Safe Trans Loader (STL) that mitigates or prevents memory corruption attacks. The STL can be applied to already released executable binaries in an operational phase. Note that the STL replaces vulnerable library functions with safe substitute functions when it loads the protected binary. These safe substitute functions mitigate or prevent stack-based buffer overflow attacks, heap-based buffer overflow attacks, and use-after-free attacks. Since the STL has minimal dependencies on the execution environment, it does not require specific changes to the existing operating system or library. Further, through our evaluation, the runtime overhead of the STL is only 1.24%.

Keywords: Memory corruption · Stack-based buffer overflow
Heap-based buffer overflow · Use-after-free · Mitigation
Prevention · Loader

1 Introduction

Various countermeasures against memory corruption attacks such as stack-based buffer overflow (SBF) attacks, heap-based buffer overflow (HBF) attacks, and use-after-free (UAF) attacks have appeared. Some of these countermeasures such as address space layout randomization (ASLR) [25], data execution prevention (DEP) [20], and stack smashing protection (SSP) [24], have been implemented in major OSs and compilers; usually, these countermeasures are enabled by default.

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 68–83, 2018.
https://doi.org/10.1007/978-3-319-97916-8_5

To investigate the effectiveness of these countermeasures, we selected three 32-bit Linux distributions, i.e., CentOS, openSUSE, and Ubuntu. In each of these distributions, we examined the application status of four countermeasures; relocation read-only (RELRO), SSP, position-independent executable (PIE), and automatic fortification. These countermeasures are offered by GNU Compiler Collection (GCC). In the study, the ELF binaries are retrieved from directories set to the root user's default PATH environment variable.

From our survey shown in Table 1, we revealed that there were a certain number of binaries that the countermeasures are not applied. For example, only 3% binaries are applied Full RELRO in openSUSE. Further, we examined how the 79 vulnerable library functions are used in each Linux distribution. These 79 vulnerable functions are expected to be replaced with substitute safer functions of glibc 2.25 by automatic fortification of GCC. In brief, we revealed that 75% of the Linux binaries used at least one vulnerable library function. Reasons here can be divided into two cases. First, some binaries have no risk of memory corruption. Second, some vulnerable library functions do not meet the requirement of replacement by automatic fortification, even when they should be replaced. The latter one is the problem to be solved. In this case, vulnerabilities including CVE-2009-2957 and CVE-2017-14493 are not removed from the binaries.

Table 1. Appreciation situations of countermeasures on GCC

Countermeasure	Ubuntu 14.04 (1,159 binaries)	CentOS 7.3 (1,601 binaries)	openSUSE 13.2 (2,207 binaries)
Partial RELRO	83%	75%	97%
Full RELRO	13%	25%	3%
SSP	74%	91%	65%
PIE	15%	26%	11%
Automatic fortification	74%	85%	65%

As above, some released binaries are not safe because countermeasures do not work effectively. However, even when users try to apply countermeasures to such binaries, it is challenging to recompile because users often can not have the source code. Besides, many countermeasures that do not require compilation need to change the settings of the user's environment. Therefore, it is crucial for users that the countermeasure is easy to apply to released binaries.

In addition of the countermeasures against memory corruption attacks, software diversity [16,34,35] and control flow integrity [30,39,40] are proposed as countermeasures against code reuse attacks [4,28]. However, these countermeasures have the following issues.

Inability to be applied during the operational phase
From the viewpoint of the software life cycle, most countermeasures need to be used during the development phase. More specifically, the application of countermeasures in compilers and linkers requires recompilation, and it cannot be applied to executable binaries without source code.

Having a necessity to change the execution environment
If we want to use countermeasures implemented in an OS, we likely need to change the OS. If we cannot change the OS, we cannot get the benefits of these countermeasures. Further, countermeasures implemented in libraries makes interoperability problems when we use it to be replaced.

Inability to select a binary to apply
Most countermeasures can be applied either to all binaries or not at all. This implies that we cannot select a specific binary and apply countermeasures. For examples, it is difficult for a novice user to use some countermeasure such as Libsafe because he/she need to know the mechanism of a dynamic library.

High runtime overhead
According to Szekeres et al., countermeasures typically require being at most 5% to 10% runtime overhead in order to be widespread [29].

In this study, we propose the Safe Trans Loader (STL) as an application-level loader that can prevent memory corruption attacks. The STL replaces vulnerable library functions with safe substitute functions at load time. We implemented a prototype of the STL for executable and linkable format (ELF) binaries in 32-bit Linux OSs, and evaluated it. The contributions of this paper are summarized as follows:

- The STL can be individually applied to already released executable binaries in the operational phase. Since the STL works at the application level, it can be applied without changing the execution environment.
- The STL prevents three major types of attacks, i.e., SBF attacks, HBF attacks and UAF attacks. In other words, the STL integrates conventional countermeasures against each attack that are scattered on many layers into one application-level loader (for example, Libsafe is a library-based countermeasure against only SBF attacks). Because of using application-level loader approach, even novice users can apply all countermeasures easily at one time.
- The runtime overhead of the STL is approximately 1.24% on average, and it meets the requirement of [29].

2 Threat Model

We assume SBF, HBF, and UAF attacks that execute arbitrary code as our threat models. The purpose then is to mitigate or prevent these attacks through the application of the STL.

More specifically, SBF and HBF attacks refer to attacks caused by vulnerable library functions including `strcpy()`, `memcpy()`, `gets()`, and other similar

such functions. Our researches have revealed that there are a certain number of binaries which is not applied countermeasures and that binaries still used these vulnerable library functions. In particular, the use of vulnerable functions poses a potential threat, e.g., vulnerabilities are reported as CVE-2017-14492 [9] and CVE-2017-14493 [10] even today. Further, to mitigate sophisticated memory corruption attacks, it is desirable to apply multiple compiler-based countermeasures to executable binaries. Therefore, we define our threat model as SBF and HBF attacks against binaries with the potential threats or known vulnerabilities, and binaries without countermeasures.

The thread model of UAF attacks is almost the same as that of the SBF and HBF attacks. The purpose of the STL is to mitigate exploitation of dangling pointers and execution of arbitrary code. Therefore, we define our threat model as UAF attacks against binaries with known vulnerabilities, and binaries without countermeasures.

3 Proposed Method

3.1 Design of STL

Figure 1 illustrates how the STL is applied to an executable binary. The STL loads a protected binary at the application level on behalf of the OS-level loader. When loading the protected binary, if a vulnerable library function is found in the protected binary, STL replaces it with a safer function that includes countermeasures against each attack. In our prototype of the STL, we scoped 34 vulnerable library functions in glibc such as `strcpy()`, `gets()`, and so on. Table 2 shows the list of these vulnerable library functions. Note that we selected these vulnerable library functions listed by [26,32], and the functions replaced by automatic fortification.

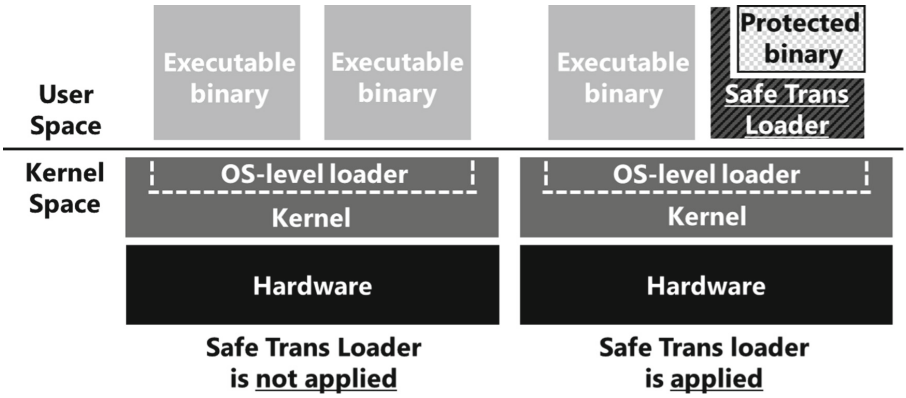


Fig. 1. Illustrating how the STL is applied to an executable binary

Table 2. 34 vulnerable library functions in glibc

No	Function name	Related vulnerability	No	Function name	Related vulnerability
1	strcpy	SBF, HBF	18	recv	SBF, HBF
2	strncpy	SBF, HBF	19	mbstowcs	SBF, HBF
3	stpcpy	SBF, HBF	20	wcstombs	SBF, HBF
4	strcat	SBF, HBF	21	getwd	SBF, HBF
5	strncat	SBF, HBF	22	getcwd	SBF, HBF
6	memcpy	SBF, HBF	23	readlink	SBF, HBF
7	gets	SBF, HBF	24	poll	SBF, HBF
8	fgets	SBF, HBF	25	scanf	SBF
9	read	SBF, HBF	26	fscanf	SBF
10	fread	SBF, HBF	27	sscanf	SBF
11	realpath	SBF, HBF	28	vscanf	SBF
12	sprintf	SBF, HBF	29	vsscanf	SBF
13	snprintf	SBF, HBF	30	vfscanf	SBF
14	vsprintf	SBF, HBF	31	malloc	HBF
15	vsnprintf	SBF, HBF	32	realloc	HBF
16	memmove	SBF, HBF	33	calloc	HBF
17	memset	SBF, HBF	34	free	HBF, UAF

3.2 Operational Flow of the STL

The STL receives the path to the protected binary at runtime. The STL loads the protected binary in the following steps:

(1) Reading the protected binary
To obtain the information about the protected binary required for the steps that follow, the STL reads the protected binary into its own heap area.

(2) Analyzing the ELF header of the protected binary
After reading the protected binary into its own heap area, the STL analyzes the ELF header of the binary, thereby obtaining the information necessary for loading.

(3) Mapping the protected binary
The STL scans the program headers of the protected binary from the offset obtained in step (2). During this scan, if the STL finds a program header corresponding to a LOAD segment, it maps the LOAD segment into its own virtual memory based on the information described in the program header. Similarly, if the STL finds a program header corresponding to a DYNAMIC segment, it loads the shared library based on the information described in the program header and performs the relocation of the library. At this time, if the shared library has already been loaded by the OS-level loader when the STL

was loaded, the shared library is not loaded, instead of being shared with the STL. During this relocation, the STL replaces functions by overwriting each entry within the .got.plt section with the address of the corresponding safer function.

(4) Creating shadow memory

In a safer function against HBF attacks, checking the boundary of the destination buffer in the heap area is performed. To do this, the size information of the buffer is required. Therefore, the STL creates shadow memory that stores the size information of the buffer.

(5) Starting the execution of the protected binary

The STL frees the heap area where the protected binary was read in step (1), then jumps to the entry point of the protected binary obtained in step (2). When the vulnerable library function is called during the execution of the protected binary, the corresponding safer function replaced in step (3) is executed.

Figure 2(a) shows an overview of the virtual memory after the steps (1) through (5) are successfully performed.

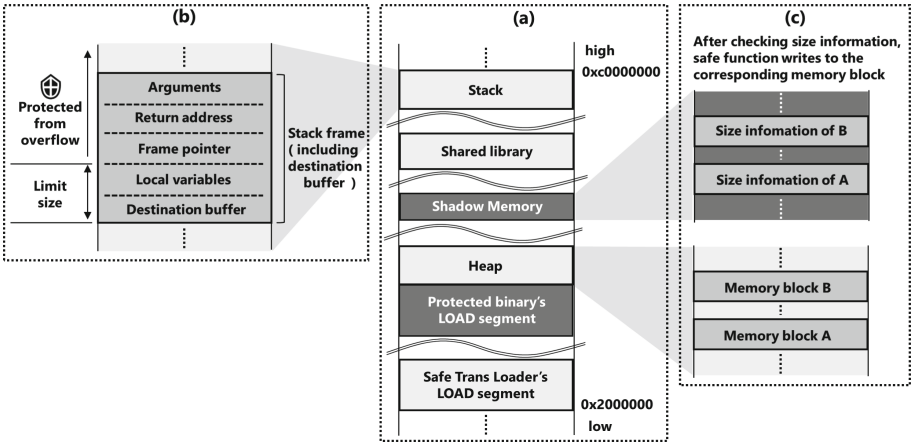


Fig. 2. Virtual memory overview of STL

3.3 Safer Functions

3.3.1 Functions Against SBF Attacks

In a safer function against SBF attacks, checking the boundary of the destination buffer in the stack area is performed. To accomplish this, the safer function calculates the limit size of the buffer using the frame pointer, an approach that refers to Libsafe [3].

When the safer function is called, first of all, the frame pointers in the stack frame are traced, and the stack frame where the destination buffer exists is

identified. Next, the limit size of the destination buffer is calculated; this limit size spans from the address of the destination buffer, which is passed as an argument to the function, to the position of the frame pointer in the stack frame. After that, the size of the character string to be written is compared with the limit size. If the size of the character string does not exceed the limit size, the write operation is performed. Otherwise, execution of the protected binary stops.

As a result, even if a character string whose size exceeds the limit size is given to the safer function, overwriting data following the frame pointer is prevented, as illustrated in Fig. 2(b).

3.3.2 Functions Against HBF Attacks

In a safer function against HBF attacks, checking the boundary of the destination buffer in the heap is performed. To accomplish this, the safer function uses shadow memory, an approach referring to AddressSanitizer [27]. AddressSanitizer [27] associates 8 bytes of the memory block in the stack or heap with 1 byte of shadow memory to store various information of the memory block. This requires an area of approximately 512 MB, not small in a 32-bit environment. Our proposed method associates 8 bytes of the memory block in the heap with 1 bit of shadow memory to store only the size information. Therefore, at the beginning of execution, the STL creates shadow memory of approximately 48 MB. The size information is denoted by setting each 1 bit of shadow memory corresponding to the start address and the end address of a memory block when the memory block is allocated. The size information means the writable range of a memory block in the heap.

In our proposed method, when the dynamic memory allocator is called, the corresponding safer function is executed. This safer function allocates a memory block in the heap and stores the size information of the memory block into shadow memory. Similarly, when an HBF vulnerable function is called, the corresponding safer function is executed. The safer function first calculates the address of shadow memory that stores the size information of the destination buffer in following two steps. First, for every 64 bytes of the memory block in the heap, the address of corresponding 1 byte in shadow memory is calculated as

$$(\text{Address\_of\_the\_destination\_buffer} \gg 6) + 0\text{x}30000000.$$

Second, the position of 1 bit (in the 1 byte) denoting the size information for every 8 bytes of the 64 bytes in the heap is calculated.

After this calculation, the size information is obtained from shadow memory. Then, the size of the character string to be written is compared with the size obtained from shadow memory. If the size of the character string does not exceed the size of the destination buffer, the write operation is performed, as illustrated in Fig. 2(c). Otherwise, execution of the protected binary stops. Note that in the safer function called instead of the `free()`, clearing the size information in the corresponding shadow memory is also performed.

3.3.3 Functions Against UAF Attacks

According to [36], UAF attacks exploit the property of reusing a memory block shortly after it is released by `free()`. Focusing on this property, our safer function delays the release of memory blocks until `free()` is called a specific number of times.

When the safer function is called instead of the `free()`, first, the pointer passed as the argument to `free()` is saved in a queue defined on the STL. This queue manages pointers to memory blocks to be released. When the safer function is called from the second time, if there is room in the queue, the pointer is added to the queue. If there is no room in the queue, the oldest pointer is then released. By delaying the memory release, as the memory block is not available for reuse immediately, thus UAF attacks can be mitigated.

3.4 Limitations

In our proposed countermeasure against SBF attacks, when local variables are placed between the destination buffer and the frame pointer on the stack, rewriting local variables cannot be detected. To solve this problem, it is necessary to combine local variable protection of SSP. The defense method of SBF includes the process of specifying a stack frame to which a destination buffer belongs by tracing a frame pointer; therefore, it is sometimes impossible to mitigate such an attack on an executable binary that does not have a frame pointer. In our proposed countermeasure against HBF attacks, the STL can successfully prevent overwriting that exceed the allocated memory block; however, the STL cannot prevent overflows inside memory blocks, such as structures. Finally, in our proposed countermeasure against UAF attacks, delaying calls of `free()` may be bypassed if attacks are made in consideration of the number of the delay times.

Note that attacks on a vulnerability caused by functions statically linked and functions defined by programmers themselves are out of protection scope of our proposed method.

4 Evaluation

4.1 Environment

In evaluating our prototype of the STL, we performed our measurements using a 32-bit Ubuntu 14.04 LTS distribution on an Intel Xeon E5620@2.40 GHz with 8 GB of RAM. Also, note that we used GCC4.8.4 and glibc2.19.

4.2 Effectiveness Against Attacks

Summarized in Table 3, we selected seven vulnerable applications, including three sample programs and four real-world applications, to verify the effectiveness of our proposed method. Sample programs were developed by referring to

example code presented on the Common Weakness Enumeration (CWE) website [11–13]. We inputted data that overwrites the return address of vulnerable applications as SBF attacks. Similarly, as HBF attacks, we inputted data that overwrites beyond the boundary of an allocated memory block. Also, as UAF attacks, we attempted to reuse memory blocks that immediately after the released and then to abuse a dangling pointer. As a result, the STL prevented all attacks to these binaries.

Table 3. Evaluation against real-world exploits

Vulnerability	CVE	Binary	Vulnerable function	Mitigation
SBF	N/A	Sample program	strcpy	✓
SBF	2013-4256 [8]	Network Audio System1.9.3	strcat	✓
SBF	2017-14493 [10]	dnsmasq2.70	memcpy	✓
HBF	N/A	Sample program	strcpy	✓
HBF	2009-2957 [7]	dnsmasq2.49	strncat	✓
HBF	2017-14492 [9]	dnsmasq2.70	sprintf	✓
UAF	N/A	Sample program	free	✓

4.3 Runtime Overhead

To determine how much runtime overhead the STL imposes on target applications, we measured overhead in SPEC CPU2006. Table 4 shows the runtime performance overhead of the STL; the Original/STL column shows the execution time of each binary when it is executed being not applied/applied the STL. Most of the execution time is the processing time that is taken when SPEC CPU inputs test data to each binary. So it is considered that the overhead whose range is from −1 to 1 is due to the tolerance of this processing time. From this table, the overhead of the STL was 1.24% on average. The worst slowdown was observed to be 4.60% for perlbench. As factors of this overhead, the initialization processing time of the STL and processing time of replacing safer functions can be considered. More specifically, initialization processing here refers to the time taken to perform steps (1) through (5) of the operational flow described in Sect. 3.2 above. Regarding initialization processing of the STL, the time taken for this processing was measured while executing each binary. As the largest time was 7.644 ms for xalancbmk among the benchmarks, we found that initialization processing time does not significantly affect the overhead on SPEC CPU2006.

Regarding the processing time of the safer function, the number of calls of 24 safer functions was measured while executing each binary. The total number of calls per second of the 24 safer functions was 1,524,712 times for perlbench, which had the longest runtime overhead, and 979 times for mcf, which had the smallest runtime overhead.

From these results, we conclude that the runtime overhead of the STL is primarily due to the processing time required by the countermeasure functions.

Table 4. Runtime overhead for each binary on SPEC CPU2006

Benchmark	Original [s]	STL [s]	Overhead [%]
400.perlbench	819.836245	857.567272	+4.602264
401.bzip2	1702.466336	1702.084314	−0.022439
403.gcc	752.265223	786.510522	+4.552291
429.mcf	434.128381	429.883889	−0.977704
445.gobmk	1044.364834	1044.893538	+0.050624
456.hmmer	2696.668994	2697.087355	+0.015514
458.sjeng	1184.304631	1180.371794	−0.332080
462.libquantum	2041.778502	2054.988188	+0.646970
464.h264ref	1856.133358	1921.322944	+3.512118
471.omnetpp	756.754092	785.687587	+3.823368
473.astar	1127.723652	1120.250665	−0.662661
483.xalancbmk	1494.22282	1527.536706	+2.229513
Average	15910.64707	16108.18477	+1.241544

4.4 Memory Overhead

To determine how much memory overhead the STL imposes on target applications, we also measured memory overhead using SPEC CPU2006. Memory usage was measured by referring to the output of `/proc/<PID>/status`, in particular, the VmPeak and VmHWM fields, which we obtained every second. The VmPeak and VmHWM fields are used to obtain peak values for virtual memory size and resident set size, respectively.

Table 5 summarizes our measurement results. The table shows that the average increases of VmPeak and VmHWM were 89 MB and 48 MB, respectively, and the increase of VmHWM was smaller than that of VmPeak. Also, the table shows that the increases in VmPeak and VmHWM for gcc and libquantum were larger than others. We concluded this is caused by a decrease in memory usage efficiency due to the delayed `free()` calls that is a countermeasure against UAF attacks. When we turned off this delay countermeasure and measured memory overhead, the increases in VmPeak for both gcc and libquantum dropped to approximately 49 MB.

5 Related Work

5.1 Countermeasures Against Buffer Overflow

SSP detects SBF attacks by inserting a “canary” underneath the frame pointer, then checking whether the canary has been altered at the end of functions.

Table 5. Memory utilization for each binary on SPEC CPU2006

Benchmark	VmPeak [MB]			VmHWM [MB]		
	Original	STL	Increase	Original	STL	Increase
400.perlbench	551	604	54	550	566	17
401.bzip2	853	906	53	850	863	13
403.gcc	812	1085	274	810	1054	244
429.mcf	840	888	48	839	843	4
445.gobmk	23	75	51	22	28	6
456.hmmer	27	75	48	26	29	3
458.sjeng	177	225	48	176	180	4
462.libquantum	98	370	272	97	328	230
464.h264ref	66	119	53	63	69	7
471.omnetpp	104	154	50	103	109	6
473.astar	300	352	52	291	306	15
483.xalancbmk	316	375	59	313	334	21
Average	347	436	88	345	393	48

Bounds checking [2,15,17,21,38] saves the object size and memory address at the time of object generation, then detects all buffer overflow attacks by checking the size of the object at the time of use. However, as these countermeasures require source code, they cannot be applied to already distributed binaries. Further, bounds checking techniques are strong, but runtime overhead is high.

AddressSanitizer [27] detects out-of-bounds memory accesses on the heap, stack, and global objects at runtime by using shadow memory that stores the information regarding object sizes. AddressSanitizer requires specified memory allocators and code instrumentation at compile-time and also imposes high runtime overhead.

There are some library-based countermeasures that require no source code to apply. For example, SafeStr [31] provides some safe string-manipulation functions. Programmers perform secure coding by using this library. Also Libsafe [3] is a library-based countermeasure that, unfortunately, is no longer in active development and is not maintained. Further, when Libsafe is applied to a binary with the setuid bit set, environment variable `LD_PRELOAD` is ignored; therefore functions cannot be replaced. As a workaround for this problem, Libsafe provides an application method for the entire system, but this method requires changing the settings of the execution environment. Moreover, Libsafe itself cannot prevent HBF attacks such as AddressSanitizer.

StackArmor [6] statically rewrites a binary before execution and mitigates SBF attacks by randomly arranging the data on the stack at runtime. However, it has high overhead.

5.2 Countermeasures Against Use-After-Free

CETS [22], DANGNULL [19], and FreeSentry [37] are compiler-based countermeasures against abusing of dangling pointers. CETS allocates an area for managing reference information when allocating a memory area. By confirming such reference information when accessing the memory area with a pointer, CETS detects dangling pointers at execution time. DANGNULL and FreeSentry both manage reference information regarding pointers, updating such reference information each time a pointer operation is performed. If a dangling pointer occurs as a result of pointer manipulation, these countermeasures nullify the pointer. However, similar to bounds checking, as these compiler-based countermeasures require source code, they cannot be applied to already distributed binaries. AddressSanitizer also detects UAF bugs by using shadow memory.

In addition to above, Cling [1] and Dieharder [23] are used as secure memory allocators in libraries. Cling mitigates attacks that exploit dangling pointers by limiting the reuse of freed memory areas only to specific conditions. Dieharder mitigates attacks that exploit dangling pointers by allocating memory from random positions on the heap. Similar to Libsafe, which we described in the previous section when these countermeasures are applied to binaries with the setuid bit set, the environment variable `LD_PRELOAD` is ignored; therefore, functions cannot be replaced.

5.3 Other Countermeasures

Other countermeasures include Taint Analysis [5,14], Software Diversity [16,34,35], Code-pointer integrity (CPI) [18], Control-flow integrity (CFI) [30,39,40], RELRO, DEP [20]. But these countermeasures require source code. Besides, CFI prevents the execution of illegal instructions by verifying the validity of indirect branch and return address, but can not avert memory destruction itself. ASLR [25] mitigates memory corruption attacks by randomizing stacks, heaps, and shared libraries at the execution time, but can not prevent the attack itself.

6 Discussion

6.1 Comparison

Table 6 compares our proposed method with existing countermeasures regarding their range of defenses, an overhead, and an application method. The STL mitigates three types of memory corruption attacks, i.e., SBF, HBF, and UAF attacks with low overhead. Some countermeasures conduct stricter memory checks than the STL does, but they have problems that they cannot be applied to distributed binaries or that they have high runtime overhead. We conclude that the STL is a practical and useful countermeasure regarding mitigating variable attacks at low cost during the operational phase.

Table 6. Comparing STL with other countermeasures

Countermeasure	Attacks			Distributed binary	Runtime overhead
	SBF	HBF	UAF		
STL (our proposed)	✓	✓	✓	✓	1.24%
SSP	✓				1%
Libsafe [3]	✓			✓	0%
StackArmor [6]	✓				16%
AddressSanitizer [27]	✓	✓	✓		73%
Bounds checking [38]	✓	✓			49%
CETS [22]			✓		48%
DANGNULL [19]			✓		80%
FreeSentry [37]			✓		42%
Cling [1]			✓	✓	−4%
DieHarder [23]			✓	✓	20%

6.2 Consideration from the Viewpoint of Software Life Cycle

In general, countermeasures can be applied in software development phase and operation phase. As countermeasures applied in the development phase, there are some methods at coding time. For example, they are methods using safe libraries [31] and methods using source code analysis tools [33] to prevent the vulnerability from being created. But these countermeasures are not perfect. Furthermore, even if software developers take extreme caution, they can not avoid containing vulnerabilities in distributed binaries, and there are cases where vulnerabilities are discovered during the operation phase. Besides, as shown in the Sect. 1, threats may still exist in executed binaries to which countermeasures in compilers are not applied or even in applied binaries. Therefore, countermeasures applied in the development phase have limitations.

On the other hand, as countermeasures applied in the operation phase, there are methods that applied to the software activation time or at the dynamic linking time. The advantage of the countermeasures applied in the operation phase is that users can apply countermeasures by themselves to the software which is already released with vulnerabilities or whose vulnerabilities are discovered after the release. Moreover, in case of our approach, unlike other countermeasures in the operation phase, the STL can be individually applied for each software, so there is little influence on the entire system such as OS switching and library exchange. Furthermore, the STL can be one of the solutions in the case where the patch is not provided due to the termination of the system support or the case where the failure occurs when the patch is applied.

7 Conclusions

In our approach, even a novice user can select a target binary separately to apply the STL. Moreover, it can be applied to already released executable binaries in the operational phase. The STL can mitigate or prevent attacks by replacing target functions with substitute functions at runtime for each binary. Further, we evaluated the performance of the STL using the SPEC CPU2006, finding that the runtime overhead of the STL was approximately 1.24% on average. Our Future tasks include solving the limitation described in Sect. 3.4, and verifying the effectiveness of UAF attacks of the STL using real-world binaries.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Number 18K11305. We are deeply grateful to Y. Kaneko, T. Uehara, Y. Sumida, Y. Hori, T. Baba, H. Miyazaki, B. Wang, R. Watanabe, and S. Kondo for this work.

References

1. Akritidis, P.: Cling: a memory allocator to mitigate dangling pointers. In: Proceedings of the 19th USENIX Conference on Security. In: USENIX Security 2010, p. 12 (2010)
2. Akritidis, P., Costa, M., Castro, M., Hand, S.: Baggy bounds checking: an efficient and backwards-compatible defense against out-of-bounds errors. In: Proceedings of the 18th Conference on USENIX Security Symposium, SSYM 2009, pp. 51–66 (2009)
3. Baratloo, A., Singh, N., Tsai, T.: Transparent run-time defense against stack smashing attacks. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC 2000, p. 21 (2000)
4. Bittau, A., Belay, A., Mashtizadeh, A., Mazières, D., Boneh, D.: Hacking blind. In: Proceedings of the 2014 IEEE Symposium on Security and Privacy, SP 2014, pp. 227–242 (2014)
5. Bosman, E., Slowinska, A., Bos, H.: Minemu: the world's fastest taint tracker. In: Proceedings of the 14th International Conference on Recent Advances in Intrusion Detection, RAID 2011, pp. 1–20 (2011)
6. Chen, X., Slowinska, A., Andriesse, D., Bos, H., Giuffrida, C.: StackArmor: comprehensive protection from stack-based memory error vulnerabilities for binaries. In: NDSS (2015)
7. CVE: CVE-2009-2957. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-2957
8. CVE: CVE-2013-4256. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-4256
9. CVE: CVE-2017-14492. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-14492
10. CVE: CVE-2017-14493. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-14493
11. CWE: CWE-121: Stack-based buffer overflow. http://cwe.mitre.org/data/definitions/121.html
12. CWE: CWE-122: Heap-based buffer overflow. http://cwe.mitre.org/data/definitions/122.html

13. CWE: CWE-416: Use after free. http://cwe.mitre.org/data/definitions/416.html
14. Davi, L., Sadeghi, A.R., Winandy, M.: ROPdefender: a detection tool to defend against return-oriented programming attacks. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011, pp. 40–51 (2011)
15. Dhurjati, D., Adve, V.: Backwards-compatible array bounds checking for C with very low overhead. In: Proceedings of the 28th International Conference on Software Engineering, ICSE 2006, pp. 162–171 (2006)
16. Hiser, J., Nguyen-Tuong, A. Co, M., Hall, M., Davidson, J.W.: ILR: where'd my gadgets go? In: Proceedings of the 2012 IEEE Symposium on Security and Privacy, SP 2012, pp. 571–585 (2012)
17. Jones, R.W.M., Kelly, P.H.J.: Backwards-compatible bounds checking for arrays and pointers in C programs. In: Proceedings of the 3rd International Workshop on Automatic Debugging (AADEBUG 1997), no. 1, pp. 13–26 (1997)
18. Kuznetsov, V., Szekeres, L., Payer, M., Candea, G., Sekar, R., Song, D.: Code-pointer integrity. In: Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation, OSDI 2014, pp. 147–163 (2014)
19. Lee, B., et al.: Preventing use-after-free with dangling pointers nullification. In: NDSS (2015)
20. Microsoft: A Detailed Description of the Data Execution Prevention (DEP) Feature in Windows XP Service Pack 2, Windows XP Tablet PC Edition 2005, and Windows Server 2003. https://support.microsoft.com/en-us/help/875352/a-detailed-description-of-the-data-execution-prevention-dep-feature-in
21. Nagarakatte, S., Zhao, J., Martin, M.M., Zdancewic, S.: SoftBound: highly compatible and complete spatial memory safety for C. SIGPLAN Not. **44**(6), 245–258 (2009)
22. Nagarakatte, S., Zhao, J., Martin, M.M., Zdancewic, S.: CETS: compiler enforced temporal safety for C. SIGPLAN Not. **45**(8), 31–40 (2010)
23. Novark, G., Berger, E.D.: DieHarder: securing the heap. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, pp. 573–584 (2010)
24. OSDev: Buffer overflow protection. https://wiki.osdev.org/Stack_Smashing_Protector
25. PaX: ASLR (Address Space Layout Randomization) - of PaX (2003). http://pax.grsecurity.net/docs/aslr.txt
26. Seacord, R.: Secure Coding in C and C++. SEI Series in Software Engineering (2013)
27. Serebryany, K., Bruening, D., Potapenko, A., Vyukov, D.: AddressSanitizer: a fast address sanity checker. In: Proceedings of the 2012 USENIX Conference on Annual Technical Conference, USENIX ATC 2012, p. 28 (2012)
28. Snow, K.Z., Monrose, F., Davi, L., Dmitrienko, A., Liebchen, C., Sadeghi, A.R.: Just-in-time code reuse: on the effectiveness of fine-grained address space layout randomization. In: Proceedings of the 2013 IEEE Symposium on Security and Privacy, SP 2013, pp. 574–588 (2013)
29. Szekeres, L., Payer, M., Wei, T., Song, D.: SoK: eternal war in memory. In: Proceedings of the 2013 IEEE Symposium on Security and Privacy, SP 2013, pp. 48–62 (2013)
30. Tice, C., et al.: Enforcing forward-edge control-flow integrity in GCC & LLVM. In: Proceedings of the 23rd USENIX Conference on Security Symposium, SEC 2014, pp. 941–955 (2014)

31. US-CERT: SafeStr (2006). https://www.us-cert.gov/bsi/articles/knowledge/coding-practices/safestr
32. Viega, J., McGraw, G.: Building Secure Software: How to Avoid Security Problems the Right Way (Paperback). Addison-Wesley Professional Computing Series. Addison-Wesley, Reading (2011)
33. Wagner, D., Foster, J.S., Brewer, E.A., Aiken, A.: A first step towards automated detection of buffer overrun vulnerabilities. In: Network and Distributed System Security Symposium, pp. 3–17 (2000)
34. Wartell, R., Mohan, V., Hamlen, K.W., Lin, Z.: Binary stirring: Self-randomizing instruction addresses of legacy x86 binary code. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS 2012, pp. 157–168 (2012)
35. Williams-King, D., et al.: Shuffler: fast and deployable continuous code re-randomization. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI 2016, pp. 367–382 (2016)
36. Yamauchi, T., Ikegami, Y.: HeapRevolver: delaying and randomizing timing of release of freed memory area to prevent use-after-free attacks. In: Chen, J., Piuri, V., Su, C., Yung, M. (eds.) NSS 2016. LNCS, vol. 9955, pp. 219–234. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46298-1_15
37. Younan, Y.: Freesentry: protecting against use-after-free vulnerabilities due to dangling pointers. In: 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, 8–11 February 2015
38. Younan, Y., Philippaerts, P., Cavallaro, L., Sekar, R., Piessens, F., Joosen, W.: Paricheck: an efficient pointer arithmetic checker for C programs. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, pp. 145–156 (2010)
39. Zhang, C., et al.: Practical control flow integrity and randomization for binary executables. In: Proceedings of the 2013 IEEE Symposium on Security and Privacy, SP 2013, pp. 559–573 (2013)
40. Zhang, M., Sekar, R.: Control flow integrity for cots binaries. In: Proceedings of the 22nd USENIX Conference on Security, SEC 2013, pp. 337–352 (2013)

Public-Key Primitives

Estimated Cost for Solving Generalized Learning with Errors Problem via Embedding Techniques

Weiyao Wang[1(✉)], Yuntao Wang[1,2], Atsushi Takayasu[1,3], and Tsuyoshi Takagi[1]

[1] Department of Mathematical Informatics, The University of Tokyo, Tokyo, Japan
weiyao_wang@mist.i.u-tokyo.ac.jp
[2] Graduate School of Mathematics, Kyushu University, Fukuoka, Japan
[3] National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract. Estimating for the computational cost of solving *learning with errors (LWE)* problem is an indispensable research topic to the lattice-based cryptography in practice. For this purpose, the *embedding* approach is usually employed. The technique first constructs a basis matrix by embedding an LWE instance. At this stage, Kannan's and Bai-Galbraith's embeddings are believed to be the most efficient approaches for the standard and the binary LWE with secret vectors in $\mathbb{Z}_q^n$ and $\{0,1\}^n$, respectively. Indeed, both methods work well with sufficiently many LWE samples. After the embedding phase, solving the unique shortest vector problem (uSVP) in the lattice spanned by the basis matrix results in solving the LWE. Recently, there are several lattice-based schemes whose secret vectors have special distributions, e.g., small elements and/or sparse vectors, have been proposed to realize efficient implementations. In this paper, to capture such settings and more, we study the LWE problem in a general setting. We analyze the LWE problem whose secret vectors are sampled from arbitrary distributions. Furthermore, we also study the problem when the number of samples is restricted. We believe that our work provides more general understanding of the hardness of LWE. Moreover, we propose a *half-twisted embedding* that contains the existing two embedding methods as special cases. This proposal enables us to analyze the hardness of LWE in a generic manner and sometimes provides improved attacks.

1 Introduction

Shor's proposal [26] of a quantum algorithm for computing integer factorizations and discrete logarithms in polynomial time means that most of the current public key cryptographic schemes will become vulnerable after the advent of practical quantum computers. Therefore, researches on post-quantum cryptographic schemes and their practical implementations have received a remarkable amount of attention in the last few years. One of the most promising candidates is *lattice-based cryptography*, whose security is usually based on the *learning with*

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 87–103, 2018.
https://doi.org/10.1007/978-3-319-97916-8_6

errors (LWE) problem [22]. Let n denote the lattice dimension and q denote the prime modulus. The LWE problem with m samples is defined in terms of a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, a random secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, and an error vector $\mathbf{e} \in \mathbb{Z}_q^m$. Given $(\mathbf{A}, \mathbf{c} := \mathbf{A}\mathbf{s} + \mathbf{e} \mod q) \in (\mathbb{Z}_q^{m \times n}, \mathbb{Z}_q^m)$, the goal is to find a secret vector $\mathbf{s}$. In general, the secret vector $\mathbf{s}$ and error vector $\mathbf{e}$ are sampled from a uniform distribution in $\mathbb{Z}_q^n$ and a discrete Gaussian distribution over the integers, respectively.

Towards a practical implementation, we want to study the concrete hardness of the LWE problem. Among the several algorithms for solving the LWE, such as BKW [3] and algebraic methods [2], the *embedding method* is believed to be the most adaptable approach. The method first constructs a basis matrix $\mathbf{B}$ by setting an LWE instance $(\mathbf{A}, \mathbf{c})$ as its elements, and then solves the unique shortest vector problem (uSVP) in the lattice spanned by $\mathbf{B}$. The latter process, i.e., solving the uSVP, is usually performed with the BKZ algorithm [23], which is a blockwise generalization of the LLL algorithm [19]. The BKZ algorithm utilizes a β-dimensional SVP solver, where β is called the blocksize. Since the BKZ is an approximate SVP algorithm, the process does not always succeed. The BKZ algorithm becomes inefficient for the larger blocksize while it outputs shorter vectors. To study the concrete hardness of the LWE, we want to use as small a blocksize as possible. Thus far, several significant progress have been made to study the trade-off between the size of the blocksize β and the success probability of solving uSVP with the BKZ algorithm [4,6,15].

In this paper, we rather study the first process of the embedding approach, i.e., given an LWE instance $(\mathbf{A}, \mathbf{c})$, constructing the matrix $\mathbf{B}$. There are two major embedding methods. Kannan's embedding [17] aims at solving the "standard" LWE whose secret vector $\mathbf{s}$ is uniformly at random in $\mathbb{Z}_q^n$. Given an LWE instance, Kannan's embedding constructs a matrix $\mathbf{B}_{\mathsf{Kan}} \in \mathbb{Z}_q^{(m+1)\times(m+1)}$. The other method is Bai-Galbraith's embedding [7] that focuses on a binary LWE whose secret vector $\mathbf{s}$ is uniformly at random in $\{0,1\}^n$. Bai-Galbraith's embedding applies a *twist* to Kannan's embedding and constructs a matrix $\mathbf{B}_{\mathsf{BG}} \in \mathbb{Z}_q^{(n+m+1)\times(n+m+1)}$. The additional dimension n enables attackers to solve uSVP more efficiently by exploiting the fact that the elements of the secret $\mathbf{s}$ are small.

To solve the LWE, we should use an appropriate embedding method, where the choice sensitively depends on the LWE distribution. The security of standard public key encryption schemes, e.g., Regev encryption [22] and dual-Regev encryption [16], is based on the standard LWE. To achieve more efficient implementations, recent lattice-based schemes rely on the LWE problem with specific distributions of secret vectors $\mathbf{s}$. For example, some schemes [5,8,9] use the same distribution as that of the error vectors $\mathbf{e}$, while other schemes [6,10,13,14,21] use sparse distributions from small domains like binary LWE. Hence, in this paper, we study the LWE problem with general distributions.

Moreover, our work has an additional orthogonal extension that is rarely studied in other papers; i.e., there is a restriction on the number of LWE samples m. Previous analyses of embedding techniques assumed that there are sufficiently many samples, and utilized optimal numbers of them. Hence, we do not as yet

understand the hardness of LWE very much when only a few samples are given. Furthermore, such a restriction reflects practical scenarios: we cannot usually obtain sufficiently many samples for the same secret vector $\mathbf{s}$. To capture this scenario, Bindel et al. recently studied [8] the hardness of LWE with a restricted number of samples when the secret vectors and error vectors have the same distributions.

Our Contributions. In this paper, we provide updated analyses on the hardness of LWE in general settings. As we claim, our definition extends the standard one in two ways. First, the secret vectors $\mathbf{s}$ follow a distribution $\chi_{\mathbf{s}}$ of mean 0 and standard deviation $\sigma_{\mathbf{s}}$, and the error vectors $\mathbf{e}$ follow a distribution $\chi_{\mathbf{e}}$ of an analogous definition. The extension covers several existing variants of LWE as special cases. Second, the number of LWE samples m is a priori bounded.

To estimate the hardness of the LWE in general settings, we follow embedding approaches. A naive attempt would be to apply Kannan's embedding or Bai-Gralbraith's embedding with a matrix $\mathbf{B}_{\mathsf{Kan}} \in \mathbb{Z}_q^{(m+1)\times(m+1)}$ or $\mathbf{B}_{\mathsf{BG}} \in \mathbb{Z}_q^{(n+m+1)\times(n+m+1)}$, and then compare the estimated costs of solving the uSVP. Instead, we unify these analyses by introducing a *half-twisted embedding*. Given an LWE instance $(\mathbf{A}, \mathbf{c})$ and a *twist factor* n_{T}, the half-twisted embedding constructs a matrix $\mathbf{B}_{\mathsf{HT}} \in \mathbb{Z}_q^{(n_{\mathsf{T}}+m+1)\times(n_{\mathsf{T}}+m+1)}$. The twist factor n_{T} bridges Kannan's matrix $\mathbf{B}_{\mathsf{Kan}}$ and Bai-Galbraith's matrix $\mathbf{B}_{\mathsf{BG}}$. When we do not apply any twists, i.e., $n_{\mathsf{T}} = 0$, the half-twisted embedding becomes the same as Kannan's embedding, i.e., $\mathbf{B}_{\mathsf{HT}} = \mathbf{B}_{\mathsf{Kan}}$. When we apply full-twists, i.e., $n_{\mathsf{T}} = n$, the half-twisted embedding becomes the same as Bai-Galbraith's embedding, i.e., $\mathbf{B}_{\mathsf{HT}} = \mathbf{B}_{\mathsf{BG}}$. Hence, we use our half-twisted embedding method and study the hardness of LWE in a generic manner.

In this paper, we follow Alkim et al.'s estimate [6] to study the computational cost of solving uSVP. Since our definition has more parameters than the standard one, we carefully determine the minimum BKZ blocksize β and an optimal twist factor n_{T}. As predicted, the twist factor becomes $n_{\mathsf{T}} = 0$ for a large standard deviation $\sigma_{\mathbf{s}}$ and $n_{\mathsf{T}} = n$ otherwise. However, during the analysis, we find an additional benefit of the half-twisted embedding. For some parameters, e.g., the small number of samples m, the twist factor becomes optimal when $0 < n_{\mathsf{T}} < n$. Therefore, our half-twisted embedding technique reduces the computational cost of solving LWE with certain parameters.

Next, we show some numerical results of Frodo, which is an LWE-based scheme. The original Frodo scheme uses the same distribution of secret vectors $\mathbf{s}$ as error vectors $\mathbf{e}$. However, we also consider this case when the standard deviation of the secret vectors is large. We find that the numerical results support our theoretical analysis. For example, when $\sigma_{\mathbf{s}}/\sigma_{\mathbf{e}} = 2$, our analysis shows that the half-twisted embedding provides an improved attack when the number of samples $m \in (846, 1609)$. Our numerical results update the bit-security of the cases in which the number of samples $m = 872, 900, 1200$, and 1400.

Organization. We start in Sect. 2 with some basic notions of lattices and the BKZ algorithm. In Sect. 3, we formally define the LWE problem studied in this

paper, and review embeddings and Alkim et al.'s estimate to study the complexity of solving the problem. After that, we propose the half-twisted embedding in Sect. 4 and analyze the hardness of LWE in a general setting with the help of some numerical examples in Sect. 5.

2 Lattices and BKZ Algorithm

Here, we recall lattices and the BKZ algorithm which we will use for solving the LWE problem. Let $\mathbb{R}$ and $\mathbb{Z}$ denote the sets of real numbers and integers, respectively. Let $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ denote the ring of integers modulo q. Let $\mathbf{I}_m$ denote an $m \times m$ identity matrix, and let $\mathbf{0}$ be an zero vector or zero matrix.

2.1 Lattices and Lattice Problems

Lattices. A full rank d-dimensional lattice Λ is an additive discrete subgroup of $\mathbb{R}^d$, and one can be generated from a matrix $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_d\}^\top \in \mathbb{R}^{d\times d}$, where $\mathbf{b}_i$'s are linearly independent column vectors. The matrix $\mathbf{B}$ is called a basis of Λ, and we write $\Lambda(\mathbf{B})$ for a lattice generated by a basis $\mathbf{B}$. The same lattice $\Lambda(\mathbf{B})$ can be represented using different bases, and one can be obtained from another by multiplying it by unimodular matrix.

The volume of $\Lambda(\mathbf{B})$ is defined as $\mathrm{Vol}(\Lambda) = \det\Lambda = |\det(\mathbf{B})|$, which is independent of bases. We define $\mathbf{b}_1^* = \mathbf{b}_1$ and $\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^*$ which are the corresponding Gram-Schmidt vectors, where $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle$.

Let j and k be two integers which satisfy $1 \le j \le k \le d$. The sublattice is defined as a lattice whose basis is $\mathbf{B}_{[j,k]} = \{\pi_j(\mathbf{b}_j), ..., \pi_j(\mathbf{b}_k)\}^\top \in \mathbb{R}^{(k-j+1)\times d}$, where $\pi_j(\cdot)$ is the orthogonal projection over $(\mathbf{b}_1, ..., \mathbf{b}_{j-1})$. Let $\Lambda(\mathbf{B}_{[1,k]})$ denote the lattice generated by first k basis vectors $\{\mathbf{b}_1, ..., \mathbf{b}_k\}$, and let $\Lambda(\mathbf{B}_{[j,k]})$ denote the $(k-j+1)$-rank sublattice that is the projection of $\Lambda(\mathbf{B}_{[1,k]})$ onto first $j-1$ basis vectors.

The successive minima $\lambda_i(\Lambda(\mathbf{B}))$ denotes the smallest radius of a zero-centered ball that contains i linearly independent vectors. By this definition, the norm of the shortest vector can be represented by $\lambda_1(\Lambda(\mathbf{B}))$. The Gaussian Heuristic predicts

$$\lambda_1(\Lambda(\mathbf{B})) \approx \sqrt{\frac{d}{2\pi e}}\mathrm{Vol}(\Lambda(\mathbf{B}))^{1/d}. \tag{1}$$

Lattice Problems. As we mentioned, the LWE problem can be reduced to a lattice problem. Let $\gamma \ge 1$, where γ is a real number.

Given a basis of Λ, the *γ-shortest vector problem (SVP_γ)* asks us to find $\mathbf{s} \in \Lambda(\mathbf{B})$ such that $0 < \|\mathbf{s}\| \le \gamma \cdot \lambda_1(\Lambda)$. Given a basis of Λ with $\lambda_2(\Lambda(\mathbf{B})) > \gamma \cdot \lambda_1(\Lambda(\mathbf{B}))$ guaranteed, the *γ-unique shortest vector problem ($uSVP_\gamma$)* asks us to find the non-zero shortest vector $\mathbf{s} \in \Lambda$. Moreover, given a basis of $\Lambda(\mathbf{B})$ and a target vector $\mathbf{t} \in \mathbb{R}^d$ such that the distance between $\mathbf{t}$ and $\Lambda(\mathbf{B})$ can be bounded by $\lambda_1(\Lambda(\mathbf{B}))/\gamma$, the *$\gamma$-bounded distance decoding (BDD_γ)* problem asks us to find vector $\mathbf{b} \in \Lambda(\mathbf{B})$ closest to $\mathbf{t}$. Lattice problems are harder to solve for smaller γ.

2.2 BKZ Algorithm

Two kinds of algorithm can be used to solve SVP_γ and uSVP_γ: exact algorithms and approximate algorithms. Exact algorithms, such as the enumeration algorithm [25] and the sieve algorithm [1], output the exact shortest vector but require (super) exponential time. In contrast, approximate algorithms, such as the LLL algorithm [19], output relatively short vectors in practical time. In high-dimensional spaces, approximate algorithms are usually feasible. The BKZ algorithm proposed by Schnorr and Euchner [23] trades off computing time with output quality using exact algorithms and the LLL algorithm as its subroutines.

The BKZ algorithm takes a basis $\mathbf{B}$ and a blocksize β and outputs a BKZ-β reduced basis by repeatedly applying an exact β-rank SVP solver (e.g. sieve or enumeration) to each local block $\mathbf{B}_{[j,k]}$ for $j = 1$ to $d-1$ and $k = \min(j+\beta-1, d)$. The BKZ algorithm treats the SVP solver as a subroutine to find the shortest vector in each sublattice $\Lambda(\mathbf{B}_{[j,k]})$, and outputs $\mathbf{x} = (x_j, ..., x_k) \in \mathbb{Z}^{k-j+1}$ such that $\|\pi_j(\Sigma_{i=j}^{k} x_i \mathbf{b}_i)\| = \lambda_1(\Lambda(\mathbf{B}_{[j,k]}))$.

Output Quality of BKZ Algorithm. We can increase the output quality of the BKZ algorithm by setting a larger β. To evaluate the output quality of BKZ algorithms, the root Hermite factor is defined as $\delta = (\|\mathbf{b}_1\|/\mathrm{Vol}(\Lambda(\mathbf{B}))^{1/d})^{1/d}$, where $\|\mathbf{b}_1\|$ is a short vector outputted by an approximate algorithm. Hermite-factor regime [12] assumes that a BKZ-β reduced basis $\mathbf{B}$ satisfies $\|\mathbf{b}_1\| \approx \delta_0^d \cdot \mathrm{Vol}(\Lambda(\mathbf{B}))^{1/d}$, where $\delta_0 = (((\pi\beta)^{1/\beta}\beta)/(2\pi e))^{1/(2(\beta-1))}$.

Furthermore, the geometric series assumption (GSA) [24] assumes that the BKZ-β reduced basis $\mathbf{B}$ satisfies

$$\|\mathbf{b}_i^*\| = \alpha^{i-1} \cdot \|\mathbf{b}_1\|, \tag{2}$$

where α is a GSA constant.

Under the GSA and Hermite factor, we can easily get $\alpha = \delta_0^{-2d/(d-1)} \approx \delta_0^{-2}$.

Complexity of BKZ Algorithm. As we claim, the BKZ algorithm calls a β-rank SVP solver as its subroutine. By choosing different SVP solvers, its complexity estimates can be varied. In this paper, we consider two exact algorithms: the enumeration algorithm with classical computing and the sieve algorithm with quantum computing. There exists a trade-off between space and time complexity. The sieve algorithm proposed by Ajtai et al. [1] has exponentially large space complexity $2^{O(\beta)}$, whereas the enumeration algorithm proposed by Schnorr and Euchner [25] only has polynomial space. In contrast, the sieve algorithm has smaller time complexity $2^{O(\beta)}$ compared with the enumeration algorithm which has the time complexity $2^{O(\beta^2)}$.

If we choose the sieve algorithm, the time complexity of the BKZ algorithm can be estimated as $\beta 2^{c_0\beta}$, where $c_0 = 0.265$ for quantum sieving [18]. This estimate is considered to be a lower bound. For the enumeration algorithm with classical computing, we use an alternative estimate formula $2^{c_1\beta\log_2\beta + c_2\beta + c_3}$, where $c_1 = 0.18728, c_2 = -1.0192, c_3 = 16.1$ [20].

2.3 Applying BKZ Algorithm to uSVP$_\gamma$

Since the BKZ algorithm is an approximate algorithm, we may fail to solve uSVP$_\gamma$ if BKZ cannot output the shortest vector. In 2008, Gama and Nguyen [15] examined the gap between $\lambda_1(\Lambda)$ and $\lambda_2(\Lambda)$, and they observed the BKZ algorithm can output the shortest vector with high probability when $\lambda_2(\Lambda)/\lambda_1(\Lambda) \geq \tau\delta_0^d$, where $\tau < 1$ depends on the blocksize β of the algorithm.

3 LWE Problem and Complexity Analysis

We define the generalized LWE problem in Sect. 3.1. In Sect. 3.2, we show how to reduce an LWE problem to a uSVP$_\gamma$ by using Kannan's and Bai-Galbraith's embeddings. In Sect. 3.3, we recall how to estimate the complexity of the LWE problem by using Alkim et al.'s estimate.

3.1 Definition of Generalized LWE Problem

As we claimed, the standard LWE problem proposed by Regev [22] chooses secret vectors $\mathbf{s} \in \mathbb{Z}_q^n$ uniformly. In contrast, the binary LWE chooses the secret vector $\mathbf{s}$ uniformly from $\{0, 1\}$.

Regev [22] shows a quantum reduction from GapSVP (or SIVP) to LWE. Therefore, LWE-based schemes are secure under the assumption that GapSVP (or SIVP; two worst-case lattice problems), is hard to solve by quantum computers. Brakerski et al. [11] proved that the binary LWE problem is also a hard problem. We will extend the standard-LWE problem and the binary LWE problem to our generalized LWE setting.

Definition 1 (Generalized LWE problem). *Let $n \in \mathbb{Z}$ denote the dimension of the LWE problem. Let $m \in \mathbb{Z}$ denote the sample number and $q \in \mathbb{Z}$ denote the modulus. Let $\chi_{\mathbf{s}}$ and $\chi_{\mathbf{e}}$ denote two independent probability distributions on $\mathbb{Z}_q$ with mean 0 and standard deviations $\sigma_{\mathbf{s}} \in \mathbb{R}$ and $\sigma_{\mathbf{e}} \in \mathbb{R}$, respectively.*

Let $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ be a matrix whose entries are sampled uniformly. Let $\mathbf{s} \in \mathbb{Z}_q^n$ be a secret vector whose entries are sampled from $\chi_{\mathbf{s}}$, and let $\mathbf{e} \in \mathbb{Z}_q^m$ be an error vector whose entries are sampled from $\chi_{\mathbf{e}}$. Finally, Let $\mathbf{c} \in \mathbb{Z}_q^m$ which is calculated by $\mathbf{c} := \mathbf{As} + \mathbf{e} \mod q \in \mathbb{Z}_q^m$.

The LWE problem is to recover vector $\mathbf{s}$ from $(\mathbf{A}, \mathbf{c}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$.

3.2 Reduction from LWE to uSVP$_\gamma$

As claimed above, we can solve the LWE problem by reducing it to a uSVP$_\gamma$ and then applying the BKZ algorithm. In this section, we recall two embeddings to reduce an LWE problem to a uSVP$_\gamma$.

Kannan's Embedding. In 1987, Kannan [17] proposed an embedding technique which reduces the LWE problem to the uSVP$_\gamma$ problem. Using it, we transform the LWE problem into a BDD problem in which the target vector $\mathbf{t} = \mathbf{c}^T$ and the basis of $\Lambda(\mathbf{B}_{\mathsf{BDD}})$ is

$$\mathbf{B}_{\mathsf{BDD}} := \begin{bmatrix} \mathbf{I}_n & \mathbf{A}' \\ \mathbf{0} & q\mathbf{I}_{m-n} \end{bmatrix} \in \mathbb{Z}_q^{m \times m}, \tag{3}$$

where $[\mathbf{I}_n | \mathbf{A}']$ denotes the reduced row echelon matrix of $\mathbf{A}^\top$, which can be easily calculated by Gaussian elimination. Next, we reduce the BDD problem to a uSVP$_\gamma$ in a lattice $\Lambda(\mathbf{B}_{\mathsf{Kan}})$ with a basis matrix

$$\mathbf{B}_{\mathsf{Kan}} := \begin{bmatrix} \mathbf{B}_{\mathsf{BDD}} & \mathbf{0} \\ \mathbf{c}^\top & 1 \end{bmatrix} = \left[\begin{array}{cc|c} \mathbf{I}_n & \mathbf{A}' & \mathbf{0} \\ \mathbf{0} & q\mathbf{I}_{m-n} & \\ \hline & \mathbf{c}^\top & 1 \end{array}\right] \in \mathbb{Z}_q^{(m+1)\times(m+1)}, \tag{4}$$

Recall that the lattice $\Lambda(\mathbf{B}_{\mathsf{Kan}})$ contains all linear combinations of the vectors in $\mathbf{B}_{\mathsf{Kan}}$. The equation $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} \mod q$ can be written as $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} + q\mathbf{k}$, where $\mathbf{k} \in \mathbb{Z}_q^m$. There exists a row vector $[-\mathbf{s}^\top | -\mathbf{k}^\top | 1] \in \mathbb{Z}_q^{(m+1)}$ such that the shortest vector in $\Lambda(\mathbf{B}_{\mathsf{Kan}})$ is $[-\mathbf{s}^\top | -\mathbf{k}^\top | 1] \cdot \mathbf{B}_{\mathsf{Kan}} = [\mathbf{e}^\top | 1] \in \mathbb{Z}_q^{(m+1)}$. Therefore, we have $\lambda_1(\Lambda(\mathbf{B}_{\mathsf{Kan}})) = \sqrt{\nu^2 \|\mathbf{s}\| + \|\mathbf{e}\| + 1} \approx \sqrt{n+m+1}\sigma_{\mathbf{e}}$.

Therefore, there exists a vector $\mathbf{v}$ such that $\mathbf{v} \cdot \mathbf{B}_{\mathsf{Kan}} = [\mathbf{e}^\top | 1]$, where $\mathbf{e} = \mathbf{c} - \mathbf{A}\mathbf{s}$. The norm of vector $[\mathbf{e}^\top | 1]$ is $\sqrt{\|\mathbf{e}\|^2 + 1} \approx \sqrt{m}\sigma_{\mathbf{e}}$. If this norm is smaller than the norm of the shortest vector estimated by Gaussian Heuristic, this LWE instance can be solved, and there exist a gap between the first and second successive minima $\lambda_2(\Lambda(\mathbf{B}_{\mathsf{Kan}}))/\lambda_1(\Lambda(\mathbf{B}_{\mathsf{Kan}}))$, which simply corresponds to the definition of uSVP$_\gamma$.

We estimate $\lambda_1(\Lambda(\mathbf{B}_{\mathsf{Kan}}))$ as $\sqrt{\|\mathbf{e}\|^2 + 1} \approx \sqrt{m}\sigma_{\mathbf{e}}$. The lattice $\Lambda(\mathbf{B}_{\mathsf{Kan}})$'s volume is q^{m-n}, and $\lambda_2(\Lambda(\mathbf{B}_{\mathsf{Kan}}))$ can be estimated by Gaussian Heuristic:

$$\lambda_2(\Lambda(\mathbf{B}_{\mathsf{Kan}})) \approx \lambda_1(\Lambda(\mathbf{B}_{\mathsf{BDD}})) \approx \mathrm{GH}(\Lambda(\mathbf{B}_{\mathsf{Kan}})) = \sqrt{\frac{m+1}{2\pi e}} q^{(m-n)/(m+1)}.$$

This result indicates that $[\mathbf{e}^\top | 1]$ is an unusually short vector compared with other short vectors. Corresponding to LWE problem' parameters, a larger q or a smaller σ will enlarge $\lambda_2(\Lambda(\mathbf{B}_{\mathsf{Kan}}))/\lambda_1(\Lambda(\mathbf{B}_{\mathsf{Kan}}))$, and a larger γ for uSVP$_\gamma$ can be applied, which leads an easier LWE problem.

Bai-Galbraith's Embedding. In 2014, Bai and Galbraith [7] proposed another embedding that can solve binary LWE more efficiently. Unlike Kannan's embedding, Bai-Galbraith's reduced an LWE problem to a uSVP$_\gamma$ problem in a lattice $\mathbf{B}_{\mathsf{BG}}$ with a basis matrix

$$\mathbf{B}_{\mathsf{BG}} := \begin{bmatrix} \nu\mathbf{I}_n & \mathbf{A}^\top & \mathbf{0} \\ \mathbf{0} & q\mathbf{I}_{m-n} & \mathbf{0} \\ \mathbf{0} & \mathbf{c}^\top & 1 \end{bmatrix} \in \mathbb{Z}_q^{(n+m+1)\times(n+m+1)}, \tag{5}$$

where ν is an embedding factor to re-balance the norm of the shortest vector. Usually, we pick $\nu = \sigma_{\mathbf{e}}/\sigma_{\mathbf{s}}$.

The basic idea of Bai-Galbraith's method is to enlarge the gap between first and second successive minima by constructing a matrix with larger volume. The analysis is similar to Kannan's one. There exists a row vector $[-\mathbf{s}^\top | -\mathbf{k}^\top | 1] \in \mathbb{Z}_q^{(m+1)}$ such that the shortest vector in $\Lambda(\mathbf{B}_{\mathsf{BG}})$ is $[-\mathbf{s}^\top | -\mathbf{k}^\top | 1] \cdot \mathbf{B}_{\mathsf{BG}} = [-\nu\mathbf{s}^\top | \mathbf{e}^\top | 1] \in \mathbb{Z}_q^{(m+1)}$, whose norm is $\lambda_1(\Lambda(\mathbf{B}_{\mathsf{BG}})) = \sqrt{\nu^2\|\mathbf{s}\| + \|\mathbf{e}\| + 1} \approx \sqrt{n+m+1}\sigma_{\mathbf{e}}$. If $\sigma_{\mathbf{s}} < \sigma_{\mathbf{e}}$, the volume of the lattice $\mathrm{Vol}(\Lambda(\mathbf{B}_{\mathsf{BG}}))$ can be enlarge to $\nu^n q^m$, and $\lambda_2(\Lambda(\mathbf{B}_{\mathsf{BG}}))$ can be estimated by Gaussian Heuristic:

$$\lambda_2(\Lambda(\mathbf{B}_{\mathsf{BG}})) \approx \sqrt{\frac{n+m+1}{2\pi e}}(\nu^n q^m)^{1/(n+m+1)}. \tag{6}$$

3.3 Complexity Analysis Using Alkim et al.'s Estimate

After reducing the LWE problem to a uSVP_γ, we should estimate the cost of the BKZ algorithm to solve this uSVP_γ. Since the BKZ algorithm is an approximation algorithm, it requires a proper blocksize β in order to solve uSVP_γ with a high success probability.

Alkim et al.'s Estimate. In 2016, Alkim et al. predicted [6] that a uSVP_γ can be solved if the geometric series assumption and Hermite-factor assumption hold and

$$\sqrt{\beta/d}\|\mathbf{v}\| \approx \sqrt{\beta}\sigma_{\mathbf{e}} \leq \delta_0^{2\beta-d}\mathrm{Vol}(\Lambda(\mathbf{B}))^{1/d}, \tag{7}$$

where $\delta_0 = (((\pi\beta)^{1/\beta}\beta)/(2\pi e))^{1/(2(\beta-1))}$, and $\mathbf{v}$ denotes the unique shortest vector in lattice.

The basic idea of Alkim et al.'s estimate relies on some predictions which have been examined in practice.

First, we write the shortest vector $\mathbf{v}$ in the normalized d-dimension basis $(\mathbf{b}_1^*/\|\mathbf{b}_1^*\|, \mathbf{b}_2^*/\|\mathbf{b}_2^*\|, ..., \mathbf{b}_d^*/\|\mathbf{b}_d^*\|)$. Alkim et al.'s estimate requires it to obey a spherical distribution, which is a uniform distribution on a $\|\mathbf{v}\|$ radius ball. After that, we can estimate that $\|\pi_{d-\beta}(\mathbf{v})\| \approx \sqrt{\beta/d}\|\mathbf{v}\|$. Therefore, the left side of inequality (7) estimates the norm of $\pi_{d-\beta}(\mathbf{v})$.

Second, by using GSA and Hermite-factor regime in Sect. 2.2, the right side of inequality (7) estimates the norm of $\mathbf{b}_{d-\beta+1}^*$. Alkim et al.'s estimate predicts that the projection of the shortest vector can be recovered on index $d-\beta+1$ with high probability if $\|\pi_{d-\beta}(\mathbf{v})\| \leq \|\mathbf{b}_{d-\beta+1}^*\|$ holds.

Finally, Alkim et al.'s estimate predicts that we can recover the shortest vector soon after its projection is found. Moreover, this prediction has been theoretically proved by Albrecht et al. [4].

As we mentioned, Alkim et al.'s estimate has been widely used to calculate the complexity of solving LWE problems. There are some parameters with which Alkim et al.'s estimate cannot be satisfied even if we choose the largest blocksize β. Here, we write m_{Ks} for the smallest number of samples used for a Kannan's embedding; it will be used in the analysis that follows.

Complexity of LWE Problems. As we claimed in Sect. 2, the complexity of the BKZ algorithm can be estimated from the blocksize β. Therefore, attackers should choose the smallest blocksize β that satisfies inequality (7). Accordingly, the complexity of LWE problems can be estimated using the complexity of the BKZ algorithm with a blocksize β recommended by Alkim et al.

4 Half-Twisted Embedding

In this section, we introduce the half-twisted embedding by combining Kannans and Bai-Galbraiths embeddings. For the LWE problem in Definition 1, we need to recover the secret vector $\mathbf{s} \in \mathbb{Z}^n$ from $\mathbf{A} \in \mathbb{Z}^{m \times n}$ and $\mathbf{c} \in \mathbb{Z}_q^m$, where n is the dimension of the LWE problem and m is the number of samples. Let a positive number $n_\mathsf{T} \in [0, n]$ denote a twist factor which bridges Kannans embedding and Bai-Galbraiths embedding. We divide $\mathbf{A} = [\mathbf{A}_\mathsf{BG} | \mathbf{A}_\mathsf{Kan}]$ into matrices $\mathbf{A}_\mathsf{BG} \in \mathbb{Z}^{m \times n_\mathsf{T}}$ and $\mathbf{A}_\mathsf{Kan} \in \mathbb{Z}^{m \times (n - n_\mathsf{T})}$, and divide $\mathbf{s} = [\mathbf{s}_\mathsf{BG} | \mathbf{s}_\mathsf{Kan}]$ into vectors $\mathbf{s}_\mathsf{BG} \in \mathbb{Z}_q^{n_\mathsf{T}}$ and $\mathbf{s}_\mathsf{Kan} \in \mathbb{Z}_q^{(n - n_\mathsf{T})}$.

Our method is divided into two parts:

1. Applying Kannan's embedding to $\mathbf{A}_\mathsf{Kan}$.
   Notice that $\mathbf{A}_\mathsf{Kan}$ is a $(n - n_\mathsf{T})$-column matrix. Kannan's embedding constructs an $(m+1)$-dimensional lattice from $\mathbf{A}_\mathsf{Kan}$. The basis $\mathbf{C}_\mathsf{Kan}$ is

$$\mathbf{C}_\mathsf{Kan} = \left[\begin{array}{cc|c} \mathbf{I}_{(n-n_\mathsf{T})} & \mathbf{A}_\mathsf{Kan}{}' & \mathbf{0} \\ \mathbf{0} & q\mathbf{I}_{m-(n-n_\mathsf{T})} & \\ \hline \multicolumn{2}{c|}{\mathbf{c}^T} & 1 \end{array}\right] \in \mathbb{Z}_q^{(m+1)\times(m+1)}, \tag{8}$$

   where $[\mathbf{I}_{(n-n_\mathsf{T})} | \mathbf{A}_\mathsf{Kan}{}']$ denotes the reduced row echelon form of $\mathbf{A}_\mathsf{Kan}^\top$, which can be easily calculated by Gaussian elimination.
2. Applying Bai-Galbraith's embedding to $\mathbf{A}_\mathsf{BG}$.
   $\mathbf{A}_\mathsf{BG}$ is a n_T-column matrix. Bai-Galbraith's embedding constructs an $(n_\mathsf{T} + m + 1)$-dimensional lattice from $\mathbf{A}_\mathsf{BG}$. The basis $\mathbf{C}_\mathsf{BG}$ is

$$\mathbf{C}_\mathsf{BG} = \left[\begin{array}{c|c|c} \nu\mathbf{I}_{n_\mathsf{T}} & \mathbf{A}_\mathsf{BG}^\top & \mathbf{0} \\ \hline \mathbf{0} & \multicolumn{2}{c}{\mathbf{C}_\mathsf{Kan}} \end{array}\right] \in \mathbb{Z}_q^{(n_\mathsf{T}+m+1)\times(n_\mathsf{T}+m+1)}, \tag{9}$$

   where $\nu = \sigma_\mathbf{e}/\sigma_\mathbf{s}$ is Bai-Galbraith's embedding factor.

By using our embedding, we can reduce the LWE problem to a uSVP$_\gamma$ problem in a lattice $\Lambda(\mathbf{B}_\mathsf{HT})$ with a basis matrix $\mathbf{B}_\mathsf{HT} = \mathbf{C}_\mathsf{BG}$.

$$\mathbf{B}_\mathsf{HT} = \left[\begin{array}{c|cc|c} \nu\mathbf{I}_{n_\mathsf{T}} & \multicolumn{2}{c|}{\mathbf{A}_\mathsf{BG}^\top} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I}_{(n-n_\mathsf{T})} & \mathbf{A}_\mathsf{Kan}{}' & \mathbf{0} \\ & \mathbf{0} & q\mathbf{I}_{m-(n-n_\mathsf{T})} & \\ \hline \mathbf{0} & \multicolumn{2}{c|}{\mathbf{c}^T} & 1 \end{array}\right] \in \mathbb{Z}_q^{(n_\mathsf{T}+m+1)\times(n_\mathsf{T}+m+1)}. \tag{10}$$

Our matrix construction is just a variant of traditional embeddings. Our embedding is equivalent to Kannans embedding for $n_\mathsf{T} = 0$, and it is the same

as Bai-Galbraiths embedding for $n_\mathsf{T} = n$. Therefore, our method can be used to analyze which embedding is more efficient or find some intermediate condition that may perform better.

The lattice $\Lambda(\mathbf{B}_{\mathsf{HT}})$ contains a vector $[-\nu\mathbf{s}_{\mathsf{BG}}|\mathbf{e}|1]$ with an expected norm $\sqrt{n_\mathsf{T}+m+1}\sigma_\mathbf{e}$. This lattice has volume $\nu^{n_\mathsf{T}}q^{m-(n-n_\mathsf{T})}$. Then $\lambda_2(\Lambda(\mathbf{B}_{\mathsf{HT}}))$ can be estimated by Gaussian Heuristic:

$$\lambda_2(\Lambda(\mathbf{B}_{\mathsf{HT}})) \approx \sqrt{\frac{n_\mathsf{T}+m+1}{2\pi e}}(\nu^{n_\mathsf{T}}q^{m-(n-n_\mathsf{T})})^{1/(n_\mathsf{T}+m+1)}. \tag{11}$$

This result denotes that $[\nu\mathbf{s}_{\mathsf{BG}}|\mathbf{e}|1]$ is an unusual short vector compared with the other short vectors i any solvable LWE instances.

5 Generalized Analysis of LWE Problem

We will start by applying Alkim et al.'s estimate to the half-twisted embedding in Sect. 5.1. In Sect. 5.2, we show that the half-twisted embedding is good for a generalized analysis of the LWE problem. In Sect. 5.3, we show some numerical results of our analysis.

5.1 Applying Alkim et al.'s Estimate to Half-Twisted Embedding

Here, we construct a formula for estimating the complexity of the half-twisted embedding by using Alkim et al.'s estimate in Sect. 3.3. The hardness of the LWE problem depends on the largest number of samples m_{limit}, dimension n, modulus q, standard deviation of the secret vector $\sigma_\mathbf{s}$ and standard deviation of the error vector $\sigma_\mathbf{e}$. Alkim et al.'s estimate tries to find the smallest blocksize β of the BKZ algorithm for given fixed parameters $(q, \sigma_\mathbf{s}, \sigma_\mathbf{e})$. The half-twisted embedding divides up the dimension n into n_T and $(n-n_\mathsf{T})$. Therefore, our goal is to construct the formula $F(m, n_\mathsf{T}, \beta)$ depending on the number of samples m, twist factor n_T and blocksize β.

By applying Alkim et al.'s estimate, we have

$$\sqrt{\beta}\sigma_\mathbf{e} \le \delta_0^{2\beta-d}\mathrm{Vol}(\Lambda(\mathbf{B}_{\mathsf{HT}}))^{1/d} = \delta_0^{2\beta-m-n_\mathsf{T}-1}(\sigma_\mathbf{e}/\sigma_\mathbf{s})^{\frac{n_\mathsf{T}}{n_\mathsf{T}+m+1}}q^{\frac{m-(n-n_\mathsf{T})}{n_\mathsf{T}+m+1}}, \tag{12}$$

where $\delta_0 = (((\pi\beta)^{1/\beta}\beta)/(2\pi e))^{1/(2(\beta-1))}$. We can obtain $F(m, n_\mathsf{T}, \beta)$ from this inequality, where

$$\begin{aligned} F(m, n_\mathsf{T}, \beta) = &\left(\frac{1}{2} + \frac{(\beta+1)(n_\mathsf{T}+m+1-2\beta)}{2(\beta-1)\beta}\right)\log\beta + \frac{m+1}{n_\mathsf{T}+m+1}\log\sigma_\mathbf{e} \\ &+ \frac{n_\mathsf{T}}{n_\mathsf{T}+m+1}\log\sigma_\mathbf{s} - \frac{n_\mathsf{T}+m-n}{n_\mathsf{T}+m+1}\log q - \frac{n_\mathsf{T}+m+1}{2\beta}\log\pi \\ &- \frac{n_\mathsf{T}+m-1}{2(\beta-1)}\log 2e + \log 2\pi e. \end{aligned} \tag{13}$$

Therefore, given $m_{\text{limit}}, n, q, \sigma_{\mathbf{s}}, \sigma_{\mathbf{e}}$, we can find a pair of $\{m, n_{\mathsf{T}}, \beta\}$ such that the blocksize β of the BKZ algorithm becomes the smallest one with respect to $F(m, n_{\mathsf{T}}, \beta)$. We adjust the number of samples m and the twist factor n_{T} to minimize the blocksize β in the inequality $F(m, n_{\mathsf{T}}, \beta) \leq 0$. In practice, we only have to consider the boundary condition and get an implicit function $F(m, n_{\mathsf{T}}, \beta) = 0$, which is much easier for us to analyze.

5.2 Analysis on Half-Twisted Embedding

Here, we analyze the complexity of the half-twisted embedding by using formula (13). Formula (13) is relatively complicated, and our analysis will start from an assumption and show the conditions under which our half-twisted embedding works better than Kannan's and Bai-Galbraith's embeddings.

First, we say that an embedding method performs better in the sense of the following definition.

Definition 2. *Consider the following three embeddings:*

(a) Kannan's embedding in Sect. 3.2
(b) Bai-Galbraith's embedding with factor $\nu = \sigma_{\mathbf{e}}/\sigma_{\mathbf{s}}$ in Sect. 3.2
(c) Half-twisted embedding in Sect. 4

We say that one embedding "performs better" if Alkim et al.'s estimate in Sect. 5.1 outputs a smaller blocksize β of the BKZ algorithm than other embeddings.

Recall that the embedding techniques reduce the LWE problem to a uSVP$_\gamma$, and the most efficient algorithm for solving the unique SVP is a BKZ algorithm of blocksize β. In Sect. 2.2, we show that the complexity of BKZ algorithm mainly depends on the blocksize β. For any LWE problem, we can estimate its complexity from the blocksize recommended by Alkim et al.'s estimate. Therefore, a smaller blocksize means the LWE problem is easier to solve, so we can say that this method performs better.

In Sect. 5.1, we got formula (13) which is complicated for analyzing. To simplify our calculation, we will base our analysis on the following assumption.

Assumption 1. *$F(m, n_{\mathsf{T}}, \beta)$ in Eq. (13) satisfies the following conditions (a) $\frac{\partial^2 \beta}{\partial m \partial n_{\mathsf{T}}} > 0$, (b) $\frac{\partial F(m_{\mathsf{Ks}}, 0, m_{\mathsf{Ks}}+1)}{\partial n_{\mathsf{T}}} < 0$, where m_{Ks} denotes the smallest number of samples used for Kannan's embedding.*

In the following, we will explain that when two conditions (a) and (b) hold in the setting of LWE problem setting. Recall that the smallest such number under condition (b) for Kannan's embedding was discussed in Sect. 3.3.

First, let us deal with condition (a). We write F_m, $F_{n_{\mathsf{T}}}$, F_β for the partial derivative of function (13), and we write $F_{n_{\mathsf{T}}m}$ for the second-order partial derivative. In an actual calculation, we can get $F_\beta < 0$. Let $\{m_{\mathsf{ADPS}}, 0, \beta\}$ be the pair of

parameters based on Alkim et al.'s estimate without any restriction on the number of samples. Then, for an LWE instance that satisfies $\sigma_{\mathbf{s}}/\sigma_{\mathbf{e}} < q^{\frac{2(n+1)}{n+m_{\mathsf{ADPS}}+1}}$, it is very easy to get $\partial F_{n_\mathsf{T}}/\partial\beta < 0$, since $F_{n_\mathsf{T}m} > 0$. Finally, we have

$$\frac{\partial^2\beta}{\partial m\partial n_\mathsf{T}} = -\frac{F_{n_\mathsf{T}m}F_\beta - (\partial F_{n_\mathsf{T}}/\partial\beta)F_m}{F_\beta^2} > 0 \tag{14}$$

Next, we deal with condition (b). Since m_{Ks} denotes the smallest number of samples for Kannan's attack, an enumeration should be used for this LWE instance, which means $\beta = m_{\mathsf{ADPS}} + 1$. Then, we have $F(m_{\mathsf{Ks}}, 0, m_{\mathsf{Ks}} + 1) = 0$. Another condition $F_{n_\mathsf{T}}(m_{\mathsf{Ks}}, 0, m_{\mathsf{Ks}} + 1) < 0$ can be easily achieved by setting $\sigma_{\mathbf{s}} < q/\sqrt{m_{\mathsf{Ks}} + 1}$. In practice, m_{Ks} is a large number, so we get

$$F_{n_\mathsf{T}}(m_{\mathsf{Ks}}, 0, m_{\mathsf{Ks}} + 1) \approx \frac{\frac{1}{2}\log(m_{\mathsf{Ks}} + 1) - \log(q\sigma_{\mathbf{s}}) - F(m_{\mathsf{Ks}}, 0, m_{\mathsf{Ks}} + 1)}{(m_{\mathsf{Ks}} + 1)} < 0. \tag{15}$$

Therefore, Assumption 1 holds for an LWE instance satisfying $\sigma_{\mathbf{s}} < \min\left(q^{\frac{2(n+1)}{n+m_{\mathsf{ADPS}}+1}}\sigma_{\mathbf{e}}, q/\sqrt{m_{\mathsf{Ks}} + 1}\right)$.

Now we can state Theorem 1 under this assumption.

Theorem 1. *Under Assumption 1, the half-twisted method performs better for a number of samples m in the interval $(m_{\mathsf{low}}, m_{\mathsf{up}})$ if $\sigma_{\mathbf{e}} < \sigma_{\mathbf{s}}$ holds, where m_{low} and m_{up} can be computed from $F(m, n_\mathsf{T}, \beta)$, $\sigma_{\mathbf{s}}$, $\sigma_{\mathbf{e}}$ and q.*

Proof. Let m_{limit} denote the largest number of samples we can choose for an attack. Consider an LWE instance defined as in Definition 1 with the parameter settings $\chi_{\mathbf{s}}$, $\chi_{\mathbf{e}}$, m, n, and q. We can easily get

$$\frac{\partial\beta}{\partial m} - \frac{\partial\beta}{\partial n_\mathsf{T}} = -\frac{F_m - F_{n_\mathsf{T}}}{F_\beta} = -\frac{\log\sigma_{\mathbf{e}} - \log\sigma_{\mathbf{s}}}{(n_\mathsf{T} + m + 1)F_\beta}. \tag{16}$$

Since $\sigma_{\mathbf{e}} < \sigma_{\mathbf{s}}$, we have $\frac{\partial\beta}{\partial m} < \frac{\partial\beta}{\partial n_\mathsf{T}}$.

By Assumption 1, since $\frac{\partial\beta}{\partial n_\mathsf{T}} = -F_{n_\mathsf{T}}(m_{\mathsf{Ks}}, 0, m_{\mathsf{Ks}}+1)/F_\beta(m_{\mathsf{Ks}}, 0, m_{\mathsf{Ks}}+1) < 0$, we can decrease β by increasing n_T. Therefore, there exist at least one m such that Bai-Galbraith's embedding or ours performs better than Kannan's one.

We suppose that only Kannan's and Bai-Galbraith's embeddings will be recommended, which means (13) outputs only $n_\mathsf{T} = 0$ or $n_\mathsf{T} = n$. This hypothesis can be easily rejected from the continuity of (13). Therefore, there exist at least one m such that our embedding performs better than Kannan's and Bai-Galbraith's embedding.

Here, we prove that such an m exists in the interval $(m_{\mathsf{low}}, m_{\mathsf{up}})$. We assume that there exists a pair $\{m_{\mathsf{up}}, 0, \beta_{\mathsf{up}}\}$ which satisfies $\partial\beta/\partial n_\mathsf{T} = 0$. Consider the case that $m = m_{\mathsf{up}}$. By Assumption 1, we have $\frac{\partial^2\beta}{\partial m\partial n_\mathsf{T}} > 0$. Therefore, when we simply decrease m, n_T will be increased so as to find a better pair for smaller β. As n_T increases to n, $\partial\beta/\partial n_\mathsf{T}$ becomes negative even if we continue to

decrease m. Therefore, there is an interval $(m_{\mathsf{low}}, m_{\mathsf{up}})$. For all $m \in (m_{\mathsf{low}}, m_{\mathsf{up}})$, our method can attack this LWE instance with a smaller blocksize.

If we don't have any pair $\{m, 0, \beta\}$ that satisfies $\partial\beta/\partial n_{\mathsf{T}} = 0$, all pairs must satisfy $\partial\beta/\partial n_{\mathsf{T}} > 0$ by continuity. Therefore, Kannan's method performs better for all pairs. This is contradict to Assumption 1. □

Now we show our analysis for the general cases. We suppose an LWE instance with a sufficient number of samples in the sense of the following definition.

Definition 3. *Consider a half-twisted embedding for an LWE instance.*

We say that the number of samples m of is "sufficient" for an LWE instance if the Alkim et al.'s estimate in Sect. 5.1 outputs a blocksize $\beta \leq n_{\mathsf{T}} + m + 1$.

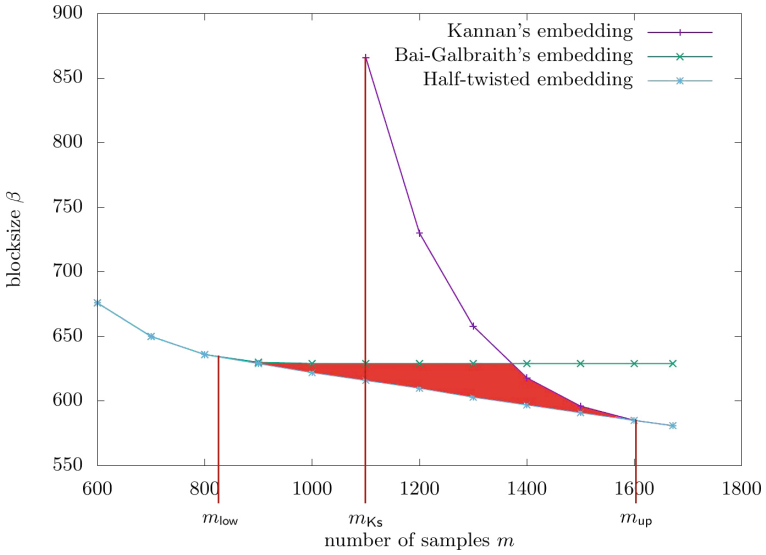


Fig. 1. Numerical result for case of $\sigma_{\mathbf{s}}/\sigma_{\mathbf{e}} = 2$. The other settings are the same as in Table 1. The horizontal axis represents the number of samples, and the vertical axis represents the best blocksize β to choose. The red area represents the improvement. m_{low} and m_{up} are from Theorem 1, and m_{Ks} is as defined in Sect. 3.3. (Color figure online)

Corollary 1 which follows directly from our Theorem 1.

Corollary 1. *Consider an LWE instance with a sufficient number of samples.*

If $\sigma_{\mathbf{e}} = \sigma_{\mathbf{s}}$, all embeddings performs same.
If $\sigma_{\mathbf{e}} > \sigma_{\mathbf{s}}$, Bai-Galbraith's embedding performs better.
If $\sigma_{\mathbf{e}} < \sigma_{\mathbf{s}}$ holds, under Assumption 1,
(a) Kannan's embedding performs better for $m > m_{\mathsf{up}}$,

(b) the half-twisted embedding performs better for $m_{\mathsf{low}} < m < m_{\mathsf{up}}$*,*
(c) Bai-Galbraith's embedding performs better for other cases $m < m_{\mathsf{low}}$
where m_{low} *and* m_{up} *are defined as in Theorem 1.*

Proof. For any LWE instance with $\sigma_{\mathbf{e}} > \sigma_{\mathbf{s}}$, we can easily get $\frac{\partial \beta}{\partial m} > \frac{\partial \beta}{\partial n_{\mathsf{T}}}$ from (16). Since m can easily be decreased, we have $\frac{\partial \beta}{\partial m} \leq 0$. Therefore, n_{T} should be chosen as large as possible since $\frac{\partial \beta}{\partial n_{\mathsf{T}}} < 0$.

For any LWE instance with $\sigma_{\mathbf{e}} < \sigma_{\mathbf{s}}$, the corollary follows directly from Theorem 1. □

5.3 Numerical Results of Our Analysis

Here, we show some numerical results of the half-twisted embedding for an LWE instance of Theorem 1 that satisfies $\sigma_{\mathbf{e}} < \sigma_{\mathbf{s}}$ and $m \in (m_{\mathsf{low}}, m_{\mathsf{up}})$.

Table 1. Numerical result. β_{Kan} and β_{BG} are recommend parameters for Kannan's and Bai-Galbraith's embedding. "q-sieve" denotes the bit complexity estimated by the sieve algorithm, and "enum" denotes the bit complexity estimated by the enumeration algorithm. "$\mathrm{bits}_{\mathrm{old}}$" is estimated by using Kannan's and Bai-Galbraith's embedding, and "$\mathrm{bits}_{\mathrm{new}}$" additionally considers our half-twisted embeddings.

$\sigma_{\mathbf{s}}/\sigma_{\mathbf{e}}$	m_{low}	m_{up}	m	β_{Kan}	β_{BG}	β_{HT}	q-sieve/enum	
							$\mathrm{bits}_{\mathrm{old}}$	$\mathrm{bits}_{\mathrm{new}}$
1	852	1716	1672	581	581	581	164/424	164/424
			1200	730	581	581	164/424	164/424
			900	–	581	581	164/424	164/424
			872	–	581	581	164/424	164/424
2	846	1609	1672	581	629	581	164/424	164/424
			1600	585	629	585	165/427	165/427
			1400	618	629	**597**	174/460	**168/439**
			1200	730	629	**610**	176/471	**171/452**
			900	–	630	**629**	177/472	**176/471**
			872	–	631	**630**	177/473	177/**472**
20	829	1329	1672	581	789	581	164/424	164/424
			1600	585	789	585	165/427	165/427
			1200	730	792	**702**	203/573	**196/544**
			1000	–	816	**792**	226/663	**220/638**
			900	–	843	**838**	234/692	**232/687**
			872	–	853	**851**	236/703	236/**700**

In order to analyze parameter sizes used in practice, we choose Frodo's paranoid parameter [9] $q = 32771 \geq 2^{15}, n = 864, \sigma_{\mathbf{e}} = \sqrt{1.75}$, and calculate

$m_{\mathsf{up}}, m_{\mathsf{low}}$ by using the formula shown in Sect. 5.3. Frodo is a key exchange protocol that chooses the secret vector from the same distribution of error vectors, which means $\sigma_{\mathbf{s}} = \sigma_{\mathbf{e}}$. In Frodo, the number of samples can be between 0 and $n + 8$. Therefore, it is a limited sample case. Moreover, we enlarge the standard deviation of the secret vector to analyze some unusual cases. These cases decrease the success probability but make the problem harder.

Figure 1 plots the case of $\sigma_{\mathbf{s}}/\sigma_{\mathbf{e}} = 2$. The red area represents our improvement. It is clear that we get a speed-up around the intersection of the two embeddings by using the half-twisted embedding. Our estimate seems smoother as a result of it combining Kannan's and Bai-Galbraith's embeddings.

Table 1 shows that out techniques get a smaller β. β_{Kan} and β_{BG} are optimal β for the two original embedding techniques. We propose β_{HT} for solving LWE problem by using our technique. We show the bit complexity of the BKZ algorithm with blocksize β using the estimate in Sect. 2.2. "q-sieve" in the table is the bit complexity of the BKZ algorithm estimated by the sieve algorithm, and "enum" is the bit complexity of the BKZ algorithm estimated by the enumeration algorithm. "$\mathrm{bits}_{\mathrm{old}}$" is estimated by using Kannan's and Bai-Galbraith's embedding, and "$\mathrm{bits}_{\mathrm{new}}$" additionally considers our half-twisted embedding. We also find that sometimes $n_{\mathsf{T}} < n$ is chosen when $m = m_{\mathsf{low}}$. This result can be explained in terms of the discrete property of β. Since we have to choose a larger integer $\beta > \beta_{\mathsf{low}}$ in almost all cases, n_{T} can be made smaller without reducing the success probability.

6 Conclusion

We proposed half-twisted embedding and estimated the cost of the LWE problem in a generic manner using Alkim et al.'s estimate. If the number of LWE samples m is restricted, the half-twisted embedding updates the complexity estimate for an LWE problem which satisfies $\sigma_{\mathbf{e}} < \sigma_{\mathbf{s}} < \min\left(q^{\frac{2(n+1)}{n+m_{\mathsf{ADPS}}+1}}\sigma_{\mathbf{e}}, q/\sqrt{m_{\mathsf{Ks}}+1}\right)$ in Sect. 5.2. Moreover, for an LWE problem with known parameters, we showed which embedding provides an attack with lower complexity compared with other embeddings.

Our analysis still has some drawbacks. Although Alkim et al.'s estimate is believed to be reliable, our analysis should be experimentally verified. However, some of our heuristics hold only for high dimensional cases, which means a further study will require the large-scale computing. Moreover, our analysis considers only three embeddings. We will consider more methods for solving LWE problem as our future work.

Acknowledgement. This work was supported by JSPS KAKENHI Grant Number JP17H06571, and JST CREST Grant Number JPMJCR14D6, Japan. The second author is supported by a JSPS fellowship for Young Scientists (JP17J01987).

References

1. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the STOC 2001, pp. 601–610. ACM (2001)
2. Albrecht, M.R., Cid, C., Faugère, J., Fitzpatrick, R., Perret, L.: Algebraic algorithms for LWE problems. ACM Commun. Comput. Algebra **49**(2), 62 (2015)
3. Albrecht, M.R., Cid, C., Faugère, J., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. Des. Codes Crypt. **74**(2), 325–354 (2015)
4. Albrecht, M.R., Göpfert, F., Virdia, F., Wunderer, T.: Revisiting the expected cost of solving uSVP and applications to LWE. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 297–322. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_11
5. Alkim, E., et al.: Revisiting TESLA in the quantum random oracle model. In: Lange, T., Takagi, T. (eds.) PQCrypto 2017. LNCS, vol. 10346, pp. 143–162. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59879-6_9
6. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange–a new hope. In: Proceedings of the USENIX Security 2016, pp. 327–343. USENIX Association (2016)
7. Bai, S., Galbraith, S.D.: Lattice decoding attacks on binary LWE. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 322–337. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08344-5_21
8. Bindel, N., Buchmann, J.A., Göpfert, F., Schmidt, M.: Estimation of the hardness of the learning with errors problem with a restricted number of samples. IACR Cryptology ePrint Archive 2017/140 (2017)
9. Bos, J.W., et al.: Frodo: take off the ring! practical, quantum-secure key exchange from LWE. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security 2016, pp. 1006–1018. ACM (2016)
10. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In: IEEE Symposium on Security and Privacy 2015, pp. 553–570. IEEE Computer Society (2015)
11. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé.: Classical hardness of learning with errors. In: STOC 2013, pp. 575–584 (2013)
12. Chen, Y.: Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe. Ph.D. thesis, Paris 7 (2013)
13. Cheon, J.H., Kim, D., Lee, J., Song, Y.S.: Lizard: cut off the tail! // practical post-quantum public-key encryption from LWE and LWR. IACR Cryptology ePrint Archive 2016/1126 (2016)
14. Ding, J., Xie, X., Lin, X.: A simple provably secure key exchange scheme based on the learning with errors problem. IACR Cryptology ePrint Archive 2012/688 (2012)
15. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_3
16. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the STOC 2008, pp. 197–206. ACM (2008)
17. Kannan, R.: Minkowski's convex body theorem and integer programming. Math. Oper. Res. **12**(3), 415–440 (1987)
18. Laarhoven, T.: Search problems in cryptography: from fingerprinting to lattice sieving. Ph.D. thesis, Eindhoven University of Technology (2015)

19. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**(4), 515–534 (1982)
20. Micciancio, D., Walter, M.: Fast lattice point enumeration with minimal overhead. SODA **2015**, 276–294 (2015)
21. Peikert, C.: Lattice cryptography for the internet. In: Mosca, M. (ed.) PQCrypto 2014. LNCS, vol. 8772, pp. 197–219. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11659-4_12
22. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the STOC 2005, pp. 84–93. ACM (2005)
23. Schnorr, C.: A hierarchy of polynomial time lattice basis reduction algorithms. Theor. Comput. Sci. **53**, 201–224 (1987)
24. Schnorr, C.: Lattice reduction by random sampling and birthday methods. In: Proceedings of the STACS 2003, pp. 145–156. ACM (2003)
25. Schnorr, C., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Program. **66**, 181–199 (1994)
26. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)

(Short Paper) How to Solve DLOG Problem with Auxiliary Input

Akinaga Ueda(✉), Hayato Tada, and Kaoru Kurosawa

Ibaraki University, Mito, Japan
{17nm703x,kaoru.kurosawa.kk}@vc.ibaraki.ac.jp

Abstract. Let $\mathbb{G}$ be a group of prime order p with a generator g. We first show that if $p-1 = d_1 \dots d_t$ and $g, g^x, g^{x^{(p-1)/d_1}}, \dots, g^{x^{(p-1)/(d_1 \dots d_{t-1})}}$ are given, then x can be computed in time $O(\sqrt{d_1} + \dots + \sqrt{d_t})$ exponentiations. Further suppose that $p-1 = d_1^{e_1} \dots d_t^{e_t}$, where $d_1, \dots, d_t$ are relatively prime. We then show that x can be computed in time $O(e_1\sqrt{d_1} + \dots + e_t\sqrt{d_t})$ exponentiations if g and $g^{x^{(p-1)/d_i}}, \dots, g^{x^{(p-1)/d_i^{e_i-1}}}$ are given for $i = 1, \dots, t$.

Keywords: Discrete logarithm · Auxiliary inputs · Cheon algorithm

1 Introduction

Let $\mathbb{G}$ be a group of prime order p with a generator g. The discrete logarithm problem (DLP) is to find x from g and g^x, and its hardness is a basis of many cryptographic schemes. It is well known that the DLP is solved in time $O(\sqrt{p})$ exponentiations [3].

On the other hand, "the DLP with auxiliary inputs" is a problem to find x from g, g^x and $g^{x^e}, \dots, g^{x^{e^t}}$. The examples include weak DH problem [5], strong DH problem [6], bilinear DH inversion problem [7] and bilinear DH exponent problem [8]. Their hardness is the basis of recent schemes with advanced functionalities such as traitor tracing schemes [5], short signature schemes [6], ID-based encryption schemes [7] and broadcast encryption schemes [8].

In the DLP with auxiliary inputs, however, the auxiliary inputs $g^{x^e}, \dots, g^{x^{e^t}}$ could weaken the difficulty. From this point of view, Cheon [2] showed two algorithms for this problem. For $p-1 = ed$, his first algorithm can find x from (g, g^x, g^{x^d}) in time

$$O(\sqrt{p/d} + \sqrt{d})$$

exponentiations. If $e \approx d \approx \sqrt{p}$, it can find x in time $O(p^{1/4})$ exponentiations.

For $p-1 = d_1 \dots d_t$ such that $d_1, \dots, d_t$ are relatively prime, his second algorithm can find x from g and

$$g^{x^{(p-1)/d_1}}, \dots, g^{x^{(p-1)/d_t}} \tag{1}$$

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 104–113, 2018.
https://doi.org/10.1007/978-3-319-97916-8_7

in time

$$O(\sqrt{d_1} + \ldots + \sqrt{d_t})$$

exponentiations. If $d_1 \approx \ldots \approx d_t \approx p^{1/t}$, then it can find x in time $O(p^{1/2t})$ exponentiations.

In this paper, we show a generalization Cheon's two algorithms. In Cheon's first algorithm, we observe that the auxiliary input is written as

$$g^{x^d} = g^{x^{(p-1)/e}},$$

where $p-1 = ed$. Our first algorithm is a generalization such that $p-1 = d_1 \ldots d_t$ and the auxiliary input is

$$g^{x^{(p-1)/d_1}}, \ldots, \ g^{x^{(p-1)/(d_1 \ldots d_{t-1})}}.$$

(Here $d_1, \ldots, d_t$ are not necessarily relatively prime.) It can find x in time

$$O(\sqrt{d_1} + \ldots + \sqrt{d_t})$$

exponentiations. If $d_1 \approx \ldots \approx d_t \approx p^{1/t}$, then x can be found in time $O(t \cdot p^{1/2t})$ exponentiations. Cheon's first algorithm is obtained as a special case for $t = 2$.

In Cheon's second algorithm, the auxiliary is given by Eq. (1), where $p-1 = d_1 \ldots d_t$ and $d_1, \ldots, d_t$ are relatively prime. Our second algorithm is a generalization such that

$$p - 1 = d_1^{e_1} \ldots d_t^{e_t},$$

where $d_1, \ldots, d_t$ are relatively prime, and the auxiliary input is

$$g^{x^{(p-1)/d_i}}, \ldots, g^{x^{(p-1)/d_i^{e_i - 1}}},$$

for $i = 1, \ldots, t$. It can output x in time

$$O(e_1\sqrt{d_1} + \ldots + e_t\sqrt{d_t})$$

exponentiations. If

$$e_1 \approx \ldots \approx e_t \approx e \text{ and } d_1 \approx \ldots \approx d_t \approx p^{1/et},$$

then x can be found in time $O(et \cdot p^{1/2et})$ exponentiations. Cheon's second algorithm is obtained as a special case for $e_1 = \ldots = e_t = 1$.

den Boer [1] showed that if $p-1$ is a product of small primes and we can have access to the DH oracle, then the DLP is solved efficiently. He applied Pohlig-Hellman algorithm to the "DLP over the exponent" by utilizing the DH oracle. Cheon solved the "DLP over the exponent" by utilizing the auxiliary inputs instead of the DH oracle. We follow the same approach.

2 Discrete Logarithm Problem over the Exponent

Let $\mathbb{G}$ be a group of prime order p with a generator g. Let α be an element in $\mathbb{Z}_p$ of order n. The discrete logarithm problem (DLP) over the exponent is to find x from g, α and $y = g^{\alpha^x}$. This problem can be solved in time $O(\sqrt{n})$ exponentiations [2] (Tables 1 and 2).

Table 1. Our first algorithm

	Cheon	Proposed
$p-1$	$p-1=ed$	$p-1=d_1 \dots d_t$
The auxiliary input (exp part)	$x^d = x^{(p-1)/e}$	$x^{(p-1)/d_1}, \dots, x^{(p-1)/(d_1 \dots d_{t-1})}$
Time	$O(\sqrt{d}+\sqrt{e})$ If $\lvert d\rvert = \lvert e\rvert$, $O(p^{1/4})$	$O(\sqrt{d_1}+\dots+\sqrt{d_t})$ If $\lvert d_1\rvert = \dots = \lvert d_t\rvert$, $O(t \cdot p^{1/2t})$

Table 2. Our second algorithm

	Cheon	Proposed
$p-1$	$p-1=d_1 \dots d_t$	$p-1=d_1^{e_1} \dots d_t^{e_t}$
The auxiliary input (exp part)	$x^{p-1/d_1}, \dots, x^{p-1/d_t}$	$(x^{p-1/d_1}, \dots, x^{p-1/d_1^{e_1}})$, $\dots, (x^{p-1/d_t}, \dots, x^{p-1/d_t^{e_t}})$
Time	$O(\sqrt{d_1}+\dots+\sqrt{d_t})$ If $\lvert d_1\rvert = \dots = \lvert d_t\rvert$, $O(t \cdot p^{1/2t})$	$O(e_1\sqrt{d_1}+\dots+e_t\sqrt{d_t})$ If $\lvert d_1\rvert = \dots = \lvert d_t\rvert$ and $\lvert e_1\rvert = \dots = \lvert e_t\rvert$, $O(et \cdot p^{1/2et})$

2.1 Deterministic Algorithm

We can use Baby-step Giant-step algorithm to solve the DLP over the exponent as follows. Let $s = \lceil \sqrt{n} \rceil$. Then x is written as $x = sk+r$ for some $0 \le k < s$ and $0 \le r < s$. Then it holds that

$$y = g^{\alpha^x} = g^{\alpha^{sk+r}},$$

and

$$yg^{-\alpha^r} = g^{\alpha^{sk}}. \tag{2}$$

Therefore we first compute two lists such that

$$B = (yg^{-\alpha^0}, yg^{-\alpha^1}, \dots, yg^{-\alpha^s})$$
$$G = (g^{\alpha^s}, g^{\alpha^{2s}}, \dots, g^{\alpha^{s(s-1)}})$$

Next we find $B \cap G$, which is a solution of Eq. (2). Hence this problem can be solved by in time $O(\sqrt{n})$ exponentiations.

2.2 Probabilistic Algorithm

The DLP probabilistically can be solved by using Pollard's kangaroo method with distinguished points [4]. One can use this method to solve the DLP over the exponent. The advantage is that it needs smaller amount of memory than the Baby-step Giant-step algorithm.

3 Generalization of Cheon's First Algorithm

Let $\mathbb{G}$ be a group of prime order p with a generator g. Consider p such that

$$p - 1 = d_1 d_2 \dots d_t. \tag{3}$$

Suppose that

$$B_1 = g^{x^{(p-1)/d_1}} \tag{4}$$
$$B_2 = g^{x^{(p-1)/(d_1 d_2)}} \tag{5}$$
$$\vdots$$
$$B_{t-1} = g^{x^{(p-1)/(d_1 \dots d_{t-1})}} \tag{6}$$

are given as auxiliary inputs in addition to g and

$$y = g^x. \tag{7}$$

In this section, we show that x can be computed in time $O(\sqrt{d_1} + \dots + \sqrt{d_t})$ exponentiations. If

$$|d_1| \approx \dots \approx |d_t|,$$

then we can compute x in time $O(t \cdot p^{1/2t})$ exponentiations. Cheon's first algorithm is obtained as a special case of our algorithm for $t = 2$.

Lemma 1. *If $w \in \mathbb{Z}_{d_1 \dots d_t}$, then there exist unique $a_0, \dots, a_{t-1}$ such that*

$$w = a_0 + a_1 d_1 + a_2 (d_1 d_2) + \dots + a_{t-1}(d_1 \dots d_{t-1}) \tag{8}$$

and

$$\begin{aligned} 0 \le\ & a_0 < d_1, \\ 0 \le\ & a_1 < d_2, \\ & \vdots \\ 0 \le\ & a_{t-1} < d_t. \end{aligned}$$

Proof. For $k = 1, \dots, t-1$, let $D_k = d_1 \dots d_k$.

(1) Let a_0 be $a_0 = w \bmod d_1$. Such a_0 is unique and $0 \le a_0 < d_1$.
(2) Suppose that $a_0, \dots, a_k$ are uniquely determined in such a way that

$$w = a_0 + \dots + a_k D_k \bmod D_{k+1}.$$

Then w is written as

$$w = a_0 + \dots + a_k D_k + Q_{k+1} D_{k+1}$$

for some Q_{k+1}. Now define a_{k+1} as

$$a_{k+1} = Q_{k+1} \bmod d_{k+2}.$$

Such a_{k+1} is unique and $0 \leq a_{k+1} < d_{k+2}$. Then Q_{k+1} is written as

$$Q_{k+1} = a_{k+1} + cd_{k+2}$$

for some c. Therefore it holds that

$$\begin{aligned} w &= a_0 + \ldots + a_k D_k + (a_{k+1} + cd_{k+2}) D_{k+1} \\ &= a_0 + \ldots + a_k D_k + a_{k+1} D_{k+1} \bmod D_{k+2} \end{aligned}$$

Consequently by induction, the lemma follows. □

A random element of $\mathbb{Z}_p^*$ is a primitive element of $GF(p)$ with probability

$$\frac{\phi(p-1)}{p-1} > \frac{1}{6 \log \log (p-1)},$$

where ϕ is Euler totient function [2]. Therefore we can easily take a primitive element of $\mathbb{Z}_p$.

Theorem 1. *Let $p - 1 = d_1 d_2 \ldots d_t$. Suppose that $B_1, \ldots, B_{t-1}$ of Eqs. (4)–(6) are given as auxiliary inputs in addition to g and $y = g^x$. Then x can be computed in time $O(\sqrt{d_1} + \ldots + \sqrt{d_t})$ exponentiations.*

Proof. Let α be a primitive element of $GF(p)$. Then x is written as

$$x = \alpha^w \bmod p$$

for some $w \in \mathbb{Z}_{p-1}$. From Lemma 1, w is written like Eq. (8). Our algorithm finds $a_0, \ldots, a_{t-1}$ of Eq. (8) step by step.

(Step 0). We first compute a_0 as follows. From Eq. (8), we have $w = a_0 + d_1 s$ for some s. Therefore

$$\begin{aligned} x^{(p-1)/d_1} &= \alpha^{w(p-1)/d_1} \\ &= \alpha^{(a_0 + d_1 s)(p-1)/d_1} \\ &= \alpha^{a_0 (p-1)/d_1} \\ &= \beta_1^{a_0} \end{aligned}$$

where $\beta_1 = \alpha^{(p-1)/d_1}$. Then it holds that

$$B_1 = g^{x^{(p-1)/d_1}} = g^{\beta_1^{a_0}}.$$

Hence by applying an algorithm for the DLP over the exponent to the above equation, we can find a_0 in time $O(\sqrt{d_1})$ exponentiations because the order of $\beta_1 = \alpha^{(p-1)/d_1}$ is d_1.

(Step k). For $k = 1, \ldots, t-2$, do the following. Suppose that we have found $a_0, \ldots, a_{k-1}$. Then from Lemma 1, w is written as

$$w = A + a_k(d_1 \ldots d_k) + s(d_1 \ldots d_{k+1})$$

for some unknown s, where $A = a_0 + a_1 d_1 + \ldots + a_{k-1}(d_1 \ldots d_{k-1})$. Hence it holds that

$$x = \alpha^w = \alpha^{A + a_k(d_1 \ldots d_k) + s(d_1 \ldots d_{k+1})}.$$

Therefore

$$\begin{aligned} x^{(p-1)/(d_1 \ldots d_{k+1})} &= \alpha^{\{A + a_k(d_1 \ldots d_k) + s(d_1 \ldots d_{k+1})\} \times (p-1)/(d_1 \ldots d_{k+1})} \\ &= \gamma \cdot \alpha^{a_k(p-1)/d_{k+1}} \\ &= \gamma \cdot (\beta_{k+1})^{a_k} \end{aligned}$$

where $\gamma = \alpha^{A(p-1)/(d_1 \ldots d_{k+1})}$ and $\beta_{k+1} = \alpha^{(p-1)/d_{k+1}}$. Then it holds that

$$(B_{k+1})^{1/\gamma} = (g^{x^{(p-1)/(d_1 \ldots d_{k+1})}})^{1/\gamma} = g^{(\beta_{k+1})^{a_k}}.$$

Hence by applying an algorithm for the DLP over the exponent to the above equation, we can find a_k in time $O(\sqrt{d_{k+1}})$ exponentiations because the order of $\beta_{k+1} = \alpha^{(p-1)/d_{k+1}}$ is d_{k+1}.

(Step $t-1$). Suppose that we have found $a_0, \ldots, a_{t-2}$. Then we compute a_{t-1} as follows. From Eq. (8), we have

$$w = A + a_{t-1}(d_1 \ldots d_{t-1})$$

for some known A. Hence it holds that

$$x = g^w = \alpha^{A + a_{t-1}(d_1 \ldots d_{t-1})}.$$

and

$$\begin{aligned} x\alpha^{-A} &= \alpha^{a_{t-1}(d_1 \ldots d_{t-1})} \\ &= \alpha^{a_{t-1}(p-1)/d_t} \\ &= \beta_t^{a_{t-1}} \end{aligned}$$

where $\beta_t = \alpha^{(p-1)/d_t}$. Therefore we have

$$y^{\alpha^{-A}} = g^{x\alpha^{-A}} = g^{\beta_t^{a_{t-1}}}$$

Finally by applying an algorithm for the DLP over the exponent to the above equation, we can find a_{t-1} in time $O(\sqrt{d_t})$ exponentiations because the order of $\beta_t = \alpha^{(p-1)/d_t}$ is d_t.

Finally we compute w from $a_0, \ldots, a_{t-1}$, and then $x = \alpha^w \bmod p$. Consequently we can compute x in time $O(\sqrt{d_1} + \ldots + \sqrt{d_t})$ exponentiations. □

4 Generalization of Cheon's Second Algorithm

Let $\mathbb{G}$ be a group of prime order p with a generator g. Consider p such that

$$p - 1 = d_1^{e_1} \dots d_t^{e_t},$$

where $d_1, \dots, d_t$ are relatively prime. Suppose that for $i = 1, \dots, t$,

$$B_{i,1} = g^{x^{(p-1)/d_i}} \tag{9}$$

$$\vdots$$

$$B_{i,e_i} = g^{x^{(p-1)/d_i^{e_i}}} \tag{10}$$

are given as auxiliary inputs in addition to g. In this section, we show that x can be computed in time $O(e_1\sqrt{d_1} + \ldots + e_t\sqrt{d_t})$ exponentiations. If

$$|d_1| \approx \ldots \approx |d_t| \text{ and } |e_1| = \ldots = |e_t| = e,$$

then we can compute x in time $O(et \cdot p^{1/2et})$ exponentiations. Cheon's second algorithm is obtained as a special case for $e_1 = \ldots e_t = 1$.

In what follows, let α be a primitive element of $GF(p)$. Define $[x]$ as $[x] = g^x$.

Lemma 2. *Consider p such that $p - 1 = d^e \times f$. Suppose that g, α and*

$$B_1 = [(\alpha^w)^{(p-1)/d}], \dots, B_e = [(\alpha^w)^{(p-1)/d^e}] \tag{11}$$

are given. Then we can compute $u = w \bmod d^e$ in time $O(e\sqrt{d})$ exponentiations.

Proof. Since $u = w \bmod d^e$, we can write w as

$$w = u + sd^e \tag{12}$$

for some s. Therefore for $i = 1, \dots, e$, it holds that

$$\begin{aligned} B_i &= [(\alpha^w)^{(p-1)/d^i}] \\ &= [(\alpha^{u+sd^e})^{(p-1)/d^i}] \\ &= [(\alpha^u)^{(p-1)/d^i}] \end{aligned}$$

Also u is written as

$$u = a_0 + a_1 d + \ldots a_{e-1}(d^{e-1}) \tag{13}$$

from Lemma 1, where $0 \leq a_i < d$ for $i = 0, \dots, e-1$. We will find these $a_0, \dots, a_{e-1}$ step by step as in the proof of Theorem 1.

For $k = 0, \dots, e-1$, do the following. Suppose that we have found $a_0, \dots, a_{k-1}$. Then from Eq. (12), w is written as

$$w = u + sd^e = A + a_k(d^k) + s'(d^{k+1})$$

for some unknown s', where $A = a_0 + a_1 d + \ldots + a_{k-1} d^{k-1}$. Hence, we have

$$x = \alpha^w = \alpha^{A + a_k (d^k) + s'(d^{k+1})}.$$

Therefore

$$\begin{aligned} B_{k+1} &= [x^{(p-1)/d^{k+1}}] \\ &= [\alpha^{\{A + a_k (d^k) + s'(d^{k+1})\} \times (p-1)/d^{k+1}}] \\ &= [\gamma \cdot \alpha^{a_k (p-1)/d}] \\ &= [\gamma \cdot \beta^{a_k}] \end{aligned}$$

where $\gamma = \alpha^{A(p-1)/d^{k+1}}$ and $\beta = \alpha^{(p-1)/d}$. Then it holds that

$$(B_{k+1})^{1/\gamma} = [\beta^{a_k}].$$

Hence, by applying an algorithm for the DLP over the exponent to the above equation, we can find a_k in time $O(\sqrt{d})$ exponentiations because the order of $\beta = \alpha^{(p-1)/d}$ is d.

Therefore we can find u of Eq. (13) in time $O(e\sqrt{d})$ exponentiations. □

Theorem 2. *Let*

$$p - 1 = d_1^{e_1} \ldots d_t^{e_t}$$

where $d_1, \ldots, d_t$ are relatively prime. Suppose that $B_{i,1}, \ldots, B_{i,e_i}$ of Eqs. (9)–(10) are given for $i = 1, \ldots, t$ as auxiliary inputs in addition to g. Then x can be computed in time $O(e_1\sqrt{d_1} + \ldots + e_t\sqrt{d_t})$ exponentiation.

Proof. Let α be a primitive element of $GF(p)$. Then x is written as

$$x = \alpha^w \bmod p$$

for some $w \in \mathbb{Z}_{p-1}$. For $i = 1, \ldots, t$, let

$$w_i = w \bmod d_i^{e_i}.$$

Then we can compute w_i in time $O(e_i\sqrt{d_i})$ exponentiations from Lemma 2.

Further we can compute w from $w_1, \ldots, w_t$ by using Chinese remainder theorem. Therefore we can compute w in time $O(e_1\sqrt{d_1} + \ldots + e_t\sqrt{d_t})$ exponentiations. Hence we can find x in time $O(e_1\sqrt{d_1} + \ldots + e_t\sqrt{d_t})$ exponentiations. □

5 Example

Let $p - 1 = 5 \cdot 7 \cdot 8$ and $p = 281$. Suppose that

$$\begin{aligned} B_1 &= g^{x^{(p-1)/5}} \\ B_2 &= g^{x^{(p-1)/5 \cdot 7}} \end{aligned}$$

are given as auxiliary inputs in addition to g and $y = g^x$. Let α be a primitive element of $GF(281)$. Then x is written as

$$x = \alpha^w \bmod 281$$

for some $w \in \mathbb{Z}_{280}$. From Lemma 1, w is written as

$$w = a_0 + a_1 \cdot 5 + a_2 \cdot 5 \cdot 7,$$

where

$$\begin{aligned} 0 \le a_0 < 5, \\ 0 \le a_1 < 7, \\ 0 \le a_2 < 8. \end{aligned}$$

1. We first compute a_0 as follows. w is written as

$$w = a_0 + 5 \cdot s$$

for some s. Therefore

$$\begin{aligned} x^{(p-1)/5} &= \alpha^{w(p-1)/5} \\ &= \alpha^{(a_0+5s)(p-1)/5} \\ &= \alpha^{a_0(p-1)/5} \\ &= \beta_1^{a_0} \end{aligned}$$

where $\beta_1 = \alpha^{(p-1)/5}$. Then it holds that

$$B_1 = g^{x^{(p-1)/5}} = g^{\beta_1^{a_0}}.$$

Hence by applying an algorithm for the DLP over the exponent to the above equation, we can find a_0 in time $O(\sqrt{5})$ exponentiations because the order of $\beta_1 = \alpha^{(p-1)/5}$ is 5.

2. We compute a_1 as follows. w is written as

$$w = a_0 + a_1 \cdot 5 + 5 \cdot 7 \cdot s$$

for some s. Therefore

$$\begin{aligned} x^{(p-1)/5\cdot 7} &= \alpha^{w(p-1)/5\cdot 7} \\ &= \alpha^{(a_0+a_1\cdot 5+5\cdot 7\cdot s)(p-1)/5\cdot 7} \\ &= \gamma \cdot \alpha^{a_1(p-1)/7} \\ &= \gamma \cdot (\beta_2)^{a_1} \end{aligned}$$

where $\gamma = \alpha^{a_0(p-1)/(5\cdot 7)}$ and $\beta_2 = \alpha^{(p-1)/7}$. Then it holds that

$$B_2^{1/\gamma} = (g^{x^{(p-1)/(5\cdot 7)}})^{1/\gamma} = g^{(\beta_2)^{a_1}}.$$

Hence by applying an algorithm for the DLP over the exponent to the above equation, we can find a_1 in time $O(\sqrt{7})$ exponentiations because the order of $\beta_2 = \alpha^{(p-1)/7}$ is 7.

3. We compute a_2 as follows. We have $w = a_0 + a_1 \cdot 5 + a_2 \cdot 5 \cdot 7$. Hence it holds that

$$x = g^w = \alpha^{a_0 + a_1 \cdot 5 + a_2 \cdot 5 \cdot 7}.$$

and

$$\begin{aligned} x\alpha^{-(a_0 + a_1 \cdot 5)} &= \alpha^{a_2 5 \cdot 7} \\ &= \alpha^{a_2 (p-1)/8} \\ &= \beta_3^{a_2} \end{aligned}$$

where $\beta_3 = \alpha^{(p-1)/8}$. Therefore we have

$$y^{\alpha^{-(a_0 + a_1 \cdot 5)}} = g^{x\alpha^{-(a_0 + a_1 \cdot 5)}} = g^{\beta_3^{a_2}}$$

Finally by applying an algorithm for the DLP over the exponent to the above equation, we can find a_2 in time $O(\sqrt{8})$ exponentiations because the order of $\beta_t = \alpha^{(p-1)/8}$ is 8.

Therefore we can compute w in time $O(\sqrt{5} + \sqrt{7} + \sqrt{8})$ exponentiations. Then x is obtained from $x = \alpha^w \bmod p$. Consequently we can compute x in time $O(\sqrt{5} + \sqrt{7} + \sqrt{8})$ exponentiations.

References

1. den Boer, B.: Diffie-Hellman is as strong as discrete log for certain primes. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 530–539. Springer, New York (1990). https://doi.org/10.1007/0-387-34799-2_38
2. Cheon, J.H.: Discrete logarithm problems with auxiliary inputs. J. Cryptol. **23**(3), 457–476 (2010)
3. Galbraith, S.D.: Mathematics of Public Key Cryptography. Cambridge University Press, Cambridge (2012)
4. Pollard, J.M.: Monte Carlo methods for index computation (mod p). Math. Comput. **32**(143), 918–924 (1978)
5. Mitsunari, S., Sakai, R., Kasahara, M.: A new traitor tracing. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **85**(2), 481–484 (2002)
6. 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). https://doi.org/10.1007/978-3-540-24676-3_4
7. 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). https://doi.org/10.1007/978-3-540-24676-3_14
8. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_16

(Short Paper) Parameter Trade-Offs for NFS and ECM

Kazumaro Aoki(✉)

NTT Secure Platform Laboratories, Nippon Telegraph and Telephone Corporation,
3-9-11 Midoricho, Musashino-shi, Tokyo 180-8585, Japan
aoki.kazumaro@lab.ntt.co.jp

Abstract. This paper analyzes two factoring algorithms, NFS (Number Field Sieve) and ECM (Elliptic Curve Method). The previous results only minimize their running times, however, we may need to minimize the storage size or running time with smaller success probability. We provide these trade-offs, $L[s]$ ($s \leq 1/3$) memory requires $L[1-2s]$ running time for NFS, for example. This can be interpreted that NFS requires much more running time when reducing memory complexity.

Keywords: Number Field Sieve · NFS · Elliptic Curve Method
ECM · Time complexity · Memory complexity · Success probability

1 Introduction

The security of RSA, which is one of the most famous digital signature algorithm, depends on the difficulty of the integer factoring problem. After the publication of RSA signature, efforts to solve the integer factoring problem are increased. In the future, a quantum computer could be realized, and the integer factoring problem might be solved in a polynomial time by Shor's algorithm [1]. However, under the current technologies, the number field sieve (NFS) [2, pp. 11–42] is the fastest algorithm to factor a composite whose digit is greater than around 100 digits, which is about 350 bits. Since the current practical implementation of RSA adopts a modulus as 2048 bits or higher, it is important problem to estimate the behaviors of the number field sieve as much as precise.

The running time of the number field sieve is estimated as $L_N[1/3, (64/9)^{1/3}]$ using so-called L subexponential notation defined in Sect. 2.1. After the development of NFS, many improvements are shown. Coppersmith shows asymptotically faster modification [3], but the folklore says that it will be faster the standard NFS when the size of factoring target is larger than 2048 bits. For substeps of NFS, nonmonic rational side polynomial selection [4], bucket sieving [5], self-tuning filtering to construct a matrix [6], and the partially parallel use of block Wiedemann [7] are developed, but none of them does not show the subexponential improvement of the time and memory complexity. To realize the running time for standard NFS, NFS requires the memory complexity of $L_N[1/3, (8/9)^{1/3}]$. The running time can be reduced by parallel computation, while a step in the number field sieve requires

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 114–123, 2018.
https://doi.org/10.1007/978-3-319-97916-8_8

that a storage should be prepared in a small area and this storage cannot be distributed to the world. Of course, there might be a study that can distribute the storage, but these kinds of studies do not seem to succeed. Therefore, though the computational resources are increasing, we can assume that we cannot use the number field sieve to factor a large composite due to the limit of available memory. Fortunately, there exists a trade-off that the number field sieve can require smaller memory by increasing running time. This paper evaluate the trade-off between required memory and running time.

2 Preliminaries

2.1 Notation

Let N be a composite to be factored, and L be a subexponential representation function as

$$L_N[s,c] = \exp((c+o(1))(\log N)^s(\log\log N)^{1-s}).$$

Note that $0 \leq s \leq 1$ and $o(1)$ converges to 0 when $N \to \infty$. Moreover, $L_N[s,c]$ may be abbreviated as $L_N[s]$. Let $\pi(x)$ be the number of primes less than or equal to x, and $\Psi(x,y)$ be the number of y-smooth positive integers less than or equal to x, where we call x as y-smooth when all prime factors of x are less than or equal to y.

Let the error function and the complementary error function [8, Eqs. 7.1.1 and 7.1.2] are shown as follows.

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x \exp(-t^2)dt \quad \mathrm{erfc}(x) = 1-\mathrm{erf}(x)$$

2.2 Formulas

This paper uses Canfield-Erdös-Pomerance and Prime theorems [9, Sects. 1.4.5, 1.1.5, and 1.4.1].

$$\Psi(x,x^{1/u}) = xu^{-u(1+o(1))} \quad \pi(x) = \frac{x}{\log x}(1+o(1)) \quad \sum_{\substack{p:\text{ prime}\\ p\leq x}} \frac{1}{x} \approx \log\log x$$

$L_N[s,c]$ satisfies following equations.

$$\begin{aligned}
N &= L_N[1,1]\\
L_N[s,c_1]+L_N[s,c_2] &= L_N[s,\max\{c_1,c_2\}]\\
L_N[s,c](\log N)^{O(1)} &= L_N[s,c]\\
L_N[s_1,c_1]L_N[s_2,c_2] &= L_N[\max\{s_1,s_2\},c] \quad \text{where } c = \begin{cases} c_1 & s_1 > s_2\\ c_2 & s_1 < s_2\\ c_1+c_2 & s_1 = s_2\end{cases}\\
L_{L_N[s_1,c_1]}[s_2,c_2] &= L_N[s_1s_2, c_1^{s_2}c_2] \quad (0<s_2\leq 1,\ 0\leq c_1)\\
\pi(L_N[s,c]) &= L_N[s,c]\\
\frac{\Psi(x_1,x_2)}{x_1} &= L_N[s_1-s_2, -\frac{c_1}{c_2}(s_1-s_2)] \quad (x_i = L_N[s_i,c_i] \text{ for } i=1,2)
\end{aligned}$$

The complementary error function can be written as

$$\operatorname{erfc}(x) = \frac{\exp(-x^2)}{x\sqrt{\pi}} \sum_{n=0}^{U-1} (-1)^n \frac{(2n-1)!!}{2^n} x^{-2n} + \mathcal{E}_U(x),$$

and the absolute value of its error term is evaluated as

$$|\mathcal{E}_U(x)| \leq \frac{\exp(-x^2)}{x\sqrt{\pi}} \frac{(2U-1)!!}{2^U} x^{-2U}$$

[8, Eqs. 7.1.23 and 7.1.24]. When $U = 1$, we have

$$\operatorname{erfc}(x) = \frac{\exp(-x^2)}{x\sqrt{\pi}} + \mathcal{E}_1(x) \quad \text{and} \quad |\mathcal{E}_1(x)| \leq \frac{\exp(-x^2)}{2x\sqrt{\pi}} x^{-2}.$$

2.3 NFS Algorithm

This section introduces the notations used in this paper. Refer [2, pp. 11–42].

Polynomial Selection. Let the polynomials for the algebraic side and the rational side as $f(x) = \sum_{i=0}^{d} c_i x^i \in \mathbf{Z}[x]$ and $x - M$. Using the shared solution M, the polynomial for the algebraic side is generated by M-adic expansion for N to compute the coefficients c_i $(i = 0, 1, \ldots, d)$ such that $f(M) \equiv 0 \pmod{N}$.

Sieve. We use naïve line sieve. Let the limit be H and we use the following sieving region.

$$-H \leq a < H \qquad 0 \leq b < H \tag{1}$$

Let the norms for rational and algebraic sides as the left side of the following equations: $|a + bM| = \prod_{\substack{p:\text{ prime} \\ p \leq B_{\mathrm{R}}}} p^{e_p}$ and $|b^d f(-a/b)| = \prod_{\substack{p:\text{ prime} \\ p \leq B_{\mathrm{A}}}} p^{f_p}$. We call the pair (a, b) *relation* that satisfies the above equations. We are required to find the relations more than $\pi(B_{\mathrm{R}}) + \pi(B_{\mathrm{A}})$.

The sieve itself runs for each b, (1) compute the first sieving address a, and (2) add $\log p$ for each prime p to memory of the corresponding address. We assume other parts than the one described above as negligible complexity.

Linear Algebra. This step computes non-trivial solution of the linear system that is generated from the sieve, and its system is defined by the matrix whose size is roughly $(\pi(B_{\mathrm{R}}) + \pi(B_{\mathrm{A}})) \times (\pi(B_{\mathrm{R}}) + \pi(B_{\mathrm{A}}))$, and its elements is in $\{0, 1\}$.

Square Root. The method [2, pp. 95–102] is an example of this step.

2.4 Running Time Evaluation of NFS

Let the following parameters whose notation is shown in the previous section and use L as follows.

$$M = L_N[s_M, c_M] \quad H = L_N[s_H, c_H] \quad B_{\mathrm{R}} = L_N[s_{B_{\mathrm{R}}}, c_{B_{\mathrm{R}}}] \quad B_{\mathrm{A}} = L_N[s_{B_{\mathrm{A}}}, c_{B_{\mathrm{A}}}]$$

Note that we assume $0 < s_M,\ s_H,\ s_{B_{\mathrm{R}}},\ s_{B_{\mathrm{A}}} < 1$.

Polynomial Selection. We can choose $M \approx N^{1/d}$, and we only computes M-adic expansion. Therefore, the running time is $(\log N)^{O(1)}$. The same amount can be derived for the required memory. Because this complexity is always lower than the required complexity for the following steps including sieve and linear algebra, we ignore the complexity for polynomial selection.

Note that the degree of the polynomial is $d = \log L_N[1 - s_M, \frac{1}{c_M}]$.

Sieve. The time complexity for the computation of the first sieve step is

$$H(\pi(B_{\mathrm{R}}) + \pi(B_{\mathrm{A}})) \times (\log N)^{O(1)},$$

since we use mod$p$ arithmetic for each b $(0 \leq b < H)$ to compute the first address for all elements in factor bases. The $\log p$ addition can be done in time

$$H(\log\log B_{\mathrm{R}} + \log\log B_{\mathrm{A}})(\log N)^{O(1)},$$

since the number of additions is $\log\log B_{\mathrm{R}} + \log\log B_{\mathrm{A}}$ on average for each a in $-H \leq a < H$. In total,

$$H(\pi(B_{\mathrm{R}}) + \pi(B_{\mathrm{A}}) + H(\log\log B_{\mathrm{R}} + \log\log B_{\mathrm{A}})) \times (\log N)^{O(1)}.$$

Therefore, the time complexity is

$$L_N[s_H, c_H](L_N[s_{B_{\mathrm{R}}}, c_{B_{\mathrm{R}}}] + L_N[s_{B_{\mathrm{A}}}, c_{B_{\mathrm{A}}}] + L_N[s_H, c_H]),$$

and the memory complexity is

$$L_N[s_{B_{\mathrm{R}}}, c_{B_{\mathrm{R}}}] + L_N[s_{B_{\mathrm{R}}}, c_{B_{\mathrm{R}}}] + L_N[s_H, c_H]$$

since we can reuse the memory for each factor base and b.

Linear Algebra. We can use an algorithm for sparse matrix, block Wiedemann algorithm [10] for example, and the time complexity is $O((\pi(B_{\mathrm{R}}) + \pi(B_{\mathrm{A}}))^2)$, and the memory complexity is $O(\pi(B_{\mathrm{R}}) + \pi(B_{\mathrm{A}}))$. Therefore, the total time complexity is

$$L_N[s_{B_{\mathrm{R}}}, 2c_{B_{\mathrm{R}}}] + L_N[s_{B_{\mathrm{A}}}, 2c_{B_{\mathrm{A}}}],$$

and the total memory complexity is

$$L_N[s_{B_{\mathrm{R}}}, c_{B_{\mathrm{R}}}] + L_N[s_{B_{\mathrm{A}}}, c_{B_{\mathrm{A}}}].$$

Square Root. Since we use only $O(d^2)$ arithmetic operations at most for each prime that is in the factor bases, the time and memory complexity is

$$L_N[s_{B_\mathrm{R}}, c_{B_\mathrm{R}}] + L_N[s_{B_\mathrm{A}}, c_{B_\mathrm{A}}].$$

Since this amount is always less than the complexity for linear algebra, we ignore the complexity for computing square roots.

Summary. To summarize all time complexity, we have

$$\begin{aligned} &L_N[s_H, c_H](L_N[s_{B_\mathrm{R}}, c_{B_\mathrm{R}}] + L_N[s_{B_\mathrm{A}}, c_{B_\mathrm{A}}] + L_N[s_H, c_H]) \\ &+L_N[s_{B_\mathrm{R}}, 2c_{B_\mathrm{R}}] + L_N[s_{B_\mathrm{A}}, 2c_{B_\mathrm{A}}], \end{aligned}$$

and memory complexity is

$$L_N[s_H, c_H] + L_N[s_{B_\mathrm{R}}, c_{B_\mathrm{R}}] + L_N[s_{B_\mathrm{A}}, c_{B_\mathrm{A}}].$$

Constraint. In the linear algebra step, we need to find non-trivial solutions. We expect (a, b) in the sieving region that $|a+bM|$ is B_R-smooth and $|b^d f(-a/b)|$ is B_A-smooth as described in Sect. 2.4. The probability that a pair (a, b) is smooth is

$$\frac{\Psi(HM, B_\mathrm{R})}{HM} \frac{\Psi(MH^d, B_\mathrm{A})}{MH^d},$$

since the sizes of norms satisfy

$$\begin{aligned} |a + bM| &\leq (H+1)M \approx HM \\ |b^d f(-a/b)| &= |\sum_{i=0}^{d} c_i a^d b^{d-i}| \leq (d+1)MH^d \approx MH^d, \end{aligned}$$

and the number of elements in the sieving region (1) is $2H^2 \approx H^2$. Thus,

$$H^2 \frac{\Psi(HM, B_\mathrm{R})}{HM} \frac{\Psi(MH^d, B_\mathrm{A})}{MH^d} > \pi(B_\mathrm{R}) + \pi(B_\mathrm{A})$$

should be satisfied since the number of relations (a, b) is roughly

$$H^2 \frac{\Psi(HM, B_\mathrm{R})}{HM} \frac{\Psi(MH^d, B_\mathrm{A})}{MH^d}.$$

Let

$$\begin{aligned} HM &= L_N[s_H, c_H] L_N[s_M, c_M] &&= L_N[s_{N_\mathrm{R}}, c_{N_\mathrm{R}}] \\ MH^d &= L_N[s_M, c_M] L_N[1 - s_M + s_H, c_H/c_M] &&= L_N[s_{N_\mathrm{A}}, c_{N_\mathrm{A}}] \end{aligned}$$

and we have

$$\begin{aligned} &L_N[s_H, 2c_H] L_N[s_{N_\mathrm{R}} - s_{B_\mathrm{R}}, -\frac{c_{N_\mathrm{R}}}{c_{B_\mathrm{R}}}(s_{N_\mathrm{R}} - s_{B_\mathrm{R}})] \\ &\times L_N[s_{N_\mathrm{A}} - s_{B_\mathrm{A}}, -\frac{c_{N_\mathrm{A}}}{c_{B_\mathrm{A}}}(s_{N_\mathrm{A}} - s_{B_\mathrm{R}})] \\ &\geq L_N[s_{B_\mathrm{R}}, c_{B_\mathrm{R}}] + L_N[s_{B_\mathrm{A}}, c_{B_\mathrm{A}}]. \end{aligned}$$

Therefore, the following inequality is the constraint to success NFS.

$$\begin{aligned} & L_N[s_H, 2c_H] \\ \geq\ & L_N[s_{N_\mathrm{R}} - s_{B_\mathrm{R}}, \frac{c_{N_\mathrm{R}}}{c_{B_\mathrm{R}}}(s_{N_\mathrm{R}} - s_{B_\mathrm{R}})] \\ & \times L_N[s_{N_\mathrm{A}} - s_{B_\mathrm{A}}, \frac{c_{N_\mathrm{A}}}{c_{B_\mathrm{A}}}(s_{N_\mathrm{A}} - s_{B_\mathrm{R}})] \times (L_N[s_{B_\mathrm{R}}, c_{B_\mathrm{R}}] + L_N[s_{B_\mathrm{A}}, c_{B_\mathrm{A}}]) \end{aligned}$$

Standard Evaluation. Let the success probability is almost 1 (or a constant that does not depend on N), and we minimize the running time. Then, we need to specify the parameters as

$$\begin{array}{ll} s_M = 2/3, & s_H = s_{B_\mathrm{R}} = s_{B_\mathrm{A}} = 1/3, \\ c_M = (1/3)^{1/3}, & c_H = c_{B_\mathrm{R}} = c_{B_\mathrm{A}} = (8/9)^{1/3}, \end{array}$$

and we have

Time: $L_N[1/3, (64/9)^{1/3}]$
Memory: $L_N[1/3, (8/9)^{1/3}]$.

2.5 ECM Algorithm

Refer to the original paper [11] for details.

3 Evaluation of the Trade-Offs for NFS

3.1 Running Time and Success Probability

This section studies the success probability which is based on the evaluation in Sect. 2.4 with smaller time complexity. The time complexity depends on the sieve and linear algebra, and reaches the smallest when these complexities are the same. The time of sieve depends on the size of sieving region and the number of elements in the factor bases. Therefore, to reduce the time complexity requires to reduce the size of factor bases, and also requires to reduce the sieving region with the same order of the reduction of factor bases. The success probability can be derived using the constraints in Sect. 2.4.

Let the probability that both rational and algebraic norms will be smooth for a point (a, b) in the sieving region be as follows.

$$p_{B_\mathrm{R}} = \frac{\Psi(HM, B_\mathrm{R})}{HM} \quad p_{B_\mathrm{A}} = \frac{\Psi(MH^d, B_\mathrm{A})}{MH^d}$$

Using the size of sieving region $2H^2$, we expect to get $2H^2 p_{B_\mathrm{R}} p_{B_\mathrm{A}}$ relations on average. We assume that the number of relations follows the binomial distribution $B(2H^2, p_{B_\mathrm{R}} p_{B_\mathrm{A}})$. When $2H^2$ is large enough, we can approximate the distribution as the normal distribution $N(\mu, \sigma^2)$ with the following average and variance.

$$\mu = 2H^2 p_{B_\mathrm{R}} p_{B_\mathrm{A}} \quad \sigma^2 = 2H^2 p_{B_\mathrm{R}} p_{B_\mathrm{A}} (1 - p_{B_\mathrm{R}} p_{B_\mathrm{A}})$$

Since the constraint requires that the number of relations should be larger than the size of factor bases, the success probability is $\frac{1}{2}\mathrm{erfc}(x) = \frac{\exp(-x^2)}{x\sqrt{\pi}} + \mathcal{E}_1(x)$, where $x = \frac{\pi(B_\mathrm{R})+\pi(B_\mathrm{A})-\mu}{\sigma}$. Consider the case that $x = L_N[s_x, c_x]$ is subexponential ($s_x > 0$). The success probability $L_N[s_s, c_s]$ cannot be written in subexponential size ($0 \le s_s < 1$, $c_s \le 0$). That is, the success probability will be worse than subexponential size. To keep the success probability *subexponentially* small, the numerator of x holds as $\pi(B_\mathrm{R}) + \pi(B_\mathrm{A}) \approx \mu$. This means

$$\begin{aligned} &L_N[s_{B_\mathrm{R}}, c_{B_\mathrm{R}}] + L_N[s_{B_\mathrm{A}}, c_{B_\mathrm{A}}] \\ &= L_N[s_H, 2c_H] L_N[s_{N_\mathrm{R}} - s_{B_\mathrm{R}}, -\frac{c_{N_\mathrm{R}}}{c_{B_\mathrm{R}}}(s_{N_\mathrm{R}} - s_{B_\mathrm{R}})] \\ &\quad \times L_N[s_{N_\mathrm{A}} - s_{B_\mathrm{A}}, -\frac{c_{N_\mathrm{A}}}{c_{B_\mathrm{A}}}(s_{N_\mathrm{A}} - s_{B_\mathrm{R}})]. \end{aligned}$$

This equals to the constraint in Sect. 2.4. In summary, the success probability will be worse than subexponential order when the time complexity is subexponentially reduced.

3.2 Time Complexity and Memory Complexity

This section estimates the increase of the time complexity with success probability of almost 1 when the memory complexity is reduced.

We first study that the dominant term of the memory complexity. For the sieve step, we consider that we do not prepare sieving memory and factor the norms by ECM for each (a, b). The asymptotic time complexity is not change for this case. ECM can derive factor p with the time complexity of $L_p[1/2, \sqrt{2}]$. The sieve step only requires to find factors of rational and algebraic norms up to B_R and B_A. The relation can be verified in

$$L_N[\frac{s_{B_\mathrm{R}}}{2}, \sqrt{2c_{B_\mathrm{R}}}] + L_N[\frac{s_{B_\mathrm{A}}}{2}, \sqrt{2c_{B_\mathrm{A}}}]$$

for each (a, b). Therefore, the time complexity of the sieve step is

$$\begin{aligned} &2H^2(L_N[\frac{s_{B_\mathrm{R}}}{2}, \sqrt{2c_{B_\mathrm{R}}}] + L_N[\frac{s_{B_\mathrm{A}}}{2}, \sqrt{2c_{B_\mathrm{A}}}]) \\ &= L_N[s_H, 2c_H](L_N[\frac{s_{B_\mathrm{R}}}{2}, \sqrt{2c_{B_\mathrm{R}}}] + L_N[\frac{s_{B_\mathrm{A}}}{2}, \sqrt{2c_{B_\mathrm{A}}}]). \end{aligned}$$

Because the matrix used in linear algebra is generated during the NFS algorithm and it depends on N, a big increase of the time complexity is required whenever redoing the sieve step whenever we need the matrix. That is, the matrix should be stored in memory, and its size is $O(\pi(B_\mathrm{R}) + \pi(B_\mathrm{A}))$. When this size of memory is prepared, the factor bases can be stored in the same area, and the memory for sieve step is automatically prepared.

We estimate s-part in the memory complexity $L_N[s, c]$. Following Sect. 2.4, the memory complexity is $L_N[\max\{s_{B_\mathrm{R}}, s_{B_\mathrm{A}}\}]$ and the time complexity is $L_N[\max\{s_H, s_{B_\mathrm{R}}, s_{B_\mathrm{A}}\}]$. The constraint in Sect. 2.4, requires as follows.

$$L_N[s_H] \geq L_N[\max\{s_{N_\mathrm{R}} - s_{B_\mathrm{R}}, s_{N_\mathrm{A}} - s_{B_\mathrm{A}}, s_{B_\mathrm{R}}, s_{B_\mathrm{A}}\}]$$

Since

$$s_{N_\mathrm{R}} = \max\{s_H, s_M\} \qquad s_{N_\mathrm{A}} = \max\{s_M, 1 - s_M + s_H\},$$

we have

$$\begin{aligned} s_H \geq \max\{&s_H - s_{B_\mathrm{R}}, s_M - s_{B_\mathrm{R}}, s_M - s_{B_\mathrm{A}}, \\ &1 - s_M + s_H - s_{B_\mathrm{A}}, s_{B_\mathrm{R}}, s_{B_\mathrm{A}}\}. \end{aligned}$$

That is, s_H should be larger or equal to all elements in the set for the right hand side "max." To simplify these inequalities and to minimize the running time, we have

$$s_{B_\mathrm{R}} = s_{B_\mathrm{A}} = s, \quad s_H = 1 - 2s, \quad \text{and} \quad s_M = 1 - s$$

for the memory complexity $L_N[s]$ $(0 \leq s \leq 1/3)$, and its corresponding time complexity is $L_N[1 - 2s]$. This trade-off is worth for $1/4 < s \leq 1/3$ since ECM only requires the memory complexity of polynomial size.

Secondly, we evaluate the trade-off when the memory complexity is $L_N[1/3, c]$ $(0 < c \leq (8/9)^{1/3})$. The time complexity is

$$L_N[1/3, \max\{2c_H, 2c_{B_\mathrm{R}}, 2c_{B_\mathrm{A}}\}]$$

using Sect. 2.4 and the first half of this section, and the memory complexity is

$$L_N[1/3, \max\{c_{B_\mathrm{R}}, c_{B_\mathrm{A}}\}] = L_N[1/3, c].$$

Using the constraint in Sect. 2.4, we have

$$\begin{aligned} &L_N[1/3, 2c_H] \\ &\geq L_N[1/3, \frac{c_M}{3c_{B_\mathrm{R}}}] L_N[1/3, \frac{c_M + \frac{c_H}{c_M}}{3c_{B_\mathrm{A}}}] \times (L_N[1/3, c_{B_\mathrm{R}}] + L_N[1/3, c_{B_\mathrm{A}}]) \\ &= L_N[1/3, \frac{c_M}{3c_{B_\mathrm{R}}} + \frac{c_M + \frac{c_H}{c_M}}{3c_{B_\mathrm{A}}} + \max\{c_{B_\mathrm{R}}, c_{B_\mathrm{A}}\}]. \end{aligned}$$

Therefore, we need to satisfy the following.

$$2c_H \geq \frac{c_M}{3c_{B_\mathrm{R}}} + \frac{c_M + \frac{c_H}{c_M}}{3c_{B_\mathrm{A}}} + \max\{c_{B_\mathrm{R}}, c_{B_\mathrm{A}}\}$$

If we set $c = c_{B_\mathrm{R}} \geq c_{B_\mathrm{A}}, c_{B_\mathrm{A}}$ in the right hand side only appears in the denominator, and it is natural to set $c = c_{B_\mathrm{A}}$ since we want to set small c_H for minimizing the time complexity. On the other hand, if we set $c = c_{B_\mathrm{A}} \geq c_{B_\mathrm{R}}, c_{B_\mathrm{R}}$ in the right hand side only appears in the denominator, we can choose $c = c_{B_\mathrm{R}}$. For

both cases, the time complexity is minimized when we choose $c = c_{B_R} = c_{B_A}$. Thus,

$$2c_H \geq \frac{c_M}{3c} + \frac{c_M + \frac{c_H}{c_M}}{3c} + c = \frac{2c_M + \frac{c_H}{c_M}}{3c} + c \geq \frac{2\sqrt{2c_H}}{3c} + c.$$

To simplify this inequality, we have $4c_H^2 - (4c + \frac{8}{9c^2})c_H + c^2 \geq 0$. That is, $c_H \geq \frac{c}{2} + \frac{1}{9c^2} + \frac{1}{3}\sqrt{\frac{1}{c} + \frac{1}{9c^4}}$. In conclusion, the time complexity is

$$L_N[1/3, c + \frac{2}{9c^2} + \frac{2}{3}\sqrt{\frac{1}{c} + \frac{1}{9c^4}}]$$

for the memory complexity of $L_N[1/3, c]$ $(0 < c \leq (8/9)^{1/3})$.

4 Evaluation of the Trade-Offs for ECM

The full paper shows the details and the results are shown in Sect. 5.2.

5 Summary of the Results

5.1 NFS

- The success probability will be exponentially lower when the time complexity reduces as subexponential.
- When the memory complexity is $L_N[s]$ $(0 \leq s \leq 1/3)$, the time complexity is $L_N[1 - 2s]$. Note that this result is worth for $1/4 < s \leq 1/3$, due to the existence of ECM.
- When the memory complexity is $L_N[1/3, c]$ $(0 < c \leq (8/9)^{1/3})$, the time complexity is $L_N[1/3, c + \frac{2}{9c^2} + \frac{2}{3}\sqrt{\frac{1}{c} + \frac{1}{9c^4}}]$.

5.2 ECM

- When the time complexity is $L_p[s, c]$ $(s < 1/2)$, the success probability is $L_p[1 - s, -\frac{1-s}{c}]$.
- When the time complexity is $L_p[1/2, c]$ $(1/\sqrt{2} \leq c \leq \sqrt{2})$, the success probability is $L_p[1/2, -(\sqrt{2} - c)]$.
- When the time complexity is $L_p[1/2, c]$ $(0 < c \leq 1/\sqrt{2})$, the success probability is $L_p[1/2, -\frac{1}{2c}]$.

6 Conclusion

The integer factoring problem relates to the security of cryptography such as RSA. We studied NFS which is the fastest algorithm and ECM which is fast algorithm that does not require large amount of memory. The time complexities of these algorithms were evaluated and minimized the running time as the following condition.

- No limitation of memory usage
- No consideration for parallel computation
- Success probability is almost 1.

Therefore, for the real world, we cannot use the large amount of memory, and we need to know the success probability with smaller time complexity. This paper evaluates the success probability or time complexity with these conditions. The results show that we need much more time complexity when reducing the required memory for NFS. However, our results only show that the upper bound of the average complexity, and actual behavior of the complexity may be different, and experimental verification is expected. We wish that our results are useful when we choose a good parameter set for integer factoring experiments and security evaluation for cryptography with success probability.

References

1. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science (FOCS 1994), pp. 124–134. IEEE Computer Society (1994)
2. Lenstra, A.K., Lenstra Jr., H.W. (eds.): The Development of the Number Field Sieve. LNM, vol. 1554. Springer, Heidelberg (1993). https://doi.org/10.1007/BFb0091534
3. Coppersmith, D.: Modifications to the number field sieve. J. Cryptology **6**(3), 169–180 (1993)
4. Kleinjung, T.: On polynomial selection for the general number field sieve. Math. Comput. **75**(256), 2037–2047 (2006)
5. Aoki, K., Ueda, H.: Sieving using bucket sort. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 92–102. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30539-2_8
6. Papadopoulos, J.: A self-tuning filtering implementation for the number field sieve, CADO workshop on integer factorization. http://cado.gforge.inria.fr/workshop/
7. Aoki, K., Franke, J., Kleinjung, T., Lenstra, A.K., Osvik, D.A.: A kilobit special number field sieve factorization. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 1–12. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76900-2_1
8. Abramowitz, M., Stegun, I.A.: Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables. Applied Mathematics Ser. vol. 55. National Bureau of Standards (1964). Tenth Printing, December 1972, with corrections
9. Crandall, R., Pomerance, C.: Prime Numbers. A Computational Perspective. Springer, New York (2001). https://doi.org/10.1007/0-387-28979-8
10. Coppersmith, D.: Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. Math. Comput. **62**(205), 333–350 (1994)
11. Lenstra Jr., H.W.: Factoring integers with elliptic curves. Ann. Math. **126**(3), 649–673 (1987)

Security in Practice

Is Java Card Ready for Hash-Based Signatures?

Ebo van der Laan[1], Erik Poll[2], Joost Rijneveld[2(✉)], Joeri de Ruiter[2], Peter Schwabe[2], and Jan Verschuren[1]

[1] Netherlands National Communication Security Agency (NLNCSA), The Hague, The Netherlands
{ebo.laan,jan.verschuren}@nlncsa.nl
[2] Digital Security Group, Radboud University, Nijmegen, The Netherlands
{erikpoll,joeri}@cs.ru.nl, joost@joostrijneveld.nl, peter@cryptojedi.org

Abstract. The current Java Card platform does not seem to allow for fast implementations of hash-based signature schemes. While the underlying implementation of the cryptographic primitives provided by the API can be fast, thanks to implementations in native code or in hardware, the cumulative overhead of the many separate API calls results in prohibitive performance for many common applications. In this work, we present an implementation of XMSS^{MT} on the current Java Card platform, and make suggestions how to improve this platform in future versions.

Keywords: Post-quantum cryptography · Hash-based signatures
Java Card · XMSS^{MT}

1 Introduction

Over the past years, cryptographic schemes that promise resilience against quantum cryptanalysis have been getting more and more attention. With the start of NIST's Post-Quantum Standardization project [17] in 2017, the focus is starting to shift from an academic niche towards real-world use.

The most immediate concern with respect to quantum attacks is long-term confidentiality of data: an adversary that records ciphertext today may come back to decrypt it later, when sufficiently large quantum computers are available. Attacks against authentication, on the other hand, would require access to a quantum computer today. Nevertheless, one should be careful not to dismiss work towards practical applications of post-quantum authentication as premature. In this work, we show that even though standardization of post-quantum signature schemes is underway, widespread application still requires bridging serious gaps.

This work has been supported by the European Commission through the ICT program under contract ICT-645622 (PQCRYPTO), and by the Netherlands Organisation for Scientific Research (NWO) through Veni 2013 project 13114. Date: June 14, 2018.

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 127–142, 2018.
https://doi.org/10.1007/978-3-319-97916-8_9

Perhaps the first images that come to mind when considering cryptographic software in the real world are large data centers full of servers that terminate TLS, full disk encryption on laptops, or intricate PKI systems. It is easy to forget that most people carry several cryptographic devices in their pockets: smart cards. With estimates of over 10 billion[1] "secure elements" sold globally in 2018 [7], this is undeniably an important market.

Smart cards are often used as authentication token – in an asymmetric-key setting, the card then stores the private key and uses it to generate signatures. We examine a practical use case: setting up VPN connections using the popular OpenVPN application, for which we implemented a state-of-the-art post-quantum signature scheme. Specifically, we implement the hash-based signature scheme XMSS^{MT} [10,11] (described in Sect. 2) on the Java Card platform. XMSS^{MT} is a stateful signature scheme; a smart card implementation can conveniently record this state alongside the key material, hiding the complexity of the statefulness from the applications that use the functionality offered by the card. Section 3 describes the Java Card platform and OpenVPN use case in more detail.

We are not the first to implement hash-based signatures on a smart card. In 2013, Hülsing, Busold and Buchmann implemented a variant of XMSS on an Infineon-produced smart card [6]; their work makes even on-card key generation practical – something that cannot possibly be said of our implementation. This is done by building upon earlier work [18] where so-called 'BDS traversal' [5] was used on an 8-bit AVR, and expanding it to the multi-tree scheme that would later evolve into XMSS^{MT}. Crucially, their work uses low-level access to the underlying hardware, which is not publicly available or portable across manufacturers.

As alluded to earlier, and as is reflected by the question in the title of this paper, the results of this work are somewhat demoralizing. With signatures taking just shy of a minute (and a subsequent preparation step well over a minute and a half), for many use cases this is impractical; see Sect. 4.2 for a more detailed analysis. The main contribution of this work is clearly not to present speed records, but instead to provide a proof-of-concept and directions on how to improve the situation. Section 4 discusses our implementation of XMSS^{MT}; the issues we identify carry over into Sect. 5, where we provide suggestions for future improvements that could help make hash-based signature schemes more practical on the Java Card platform. The Java Card API has been extended in the past to support new protocols (notably the SAC/PACE protocol used in passports [3,14]), so we can expect future extensions when applications begin to require support for post-quantum cryptography.

Availability of Software. We place all software presented in this paper into the public domain to maximize reusability of our results. It is available for download at https://joostrijneveld.nl/papers/javacard-xmss.

[1] Half of these are SIM cards; financial and governmental applications make up most of the remainder.

2 XMSS^{MT}

In [11], Hülsing, Rausch and Buchmann propose XMSS^{MT}, the current state of the art in (stateful) hash-based signatures. This scheme does not stand on its own, though, as hash-based signatures go back all the way to the 1970 s when they were first described by Merkle [16]. These schemes are characterized by their very conservative security assumptions, relying only on the existence of a secure one-way function; the provably minimal assumption required for any signature scheme to exist [19].

As research is progressing and the need for post-quantum cryptography is becoming more urgent, standardization efforts are starting to take shape. This is not only limited to the aforementioned project by NIST, but also ISO and the IETF have shown interest. The latter is relevant in particular, since at the time of writing a 'Request For Comments' describing XMSS and XMSS^{MT} [10] has just been published. Our implementation is compatible with XMSS^{MT} as specified in RFC 8391, and we refer to this document for a detailed technical specification. We limit the description in this section to that which is required as a preliminary for the discussions in the remainder of this paper.

2.1 WOTS$^+$

Before describing XMSS^{MT}, it is useful to separately define WOTS$^+$ [9], a variant of the Winternitz One-Time Signature (WOTS) scheme.

Parameters. WOTS$^+$ is a one-time signature scheme: a private key must not be used to sign more than one n-byte message, where n is a parameter defined at the time of key generation. Additionally, a parameter w signifies a trade-off between signature size and computation time.

To describe the resulting scheme, we use derived values ℓ_1, ℓ_2 and ℓ, defined as $\ell_1 := \lceil \frac{8n}{\log_2(w)} \rceil$, $\ell_2 := \lfloor \frac{\log_2(\ell_1 \cdot (w-1))}{\log_2(w)} \rfloor + 1$, and $\ell := \ell_1 + \ell_2$. For the remainder of this paper, we fix the parameters[2] $n = 32$ and $w = 16$. This leads to $\ell_1 = 64$ and $\ell_2 = 3$, and thus $\ell = 67$.

Keys. A WOTS$^+$ private key consists of ℓ random values of n bytes. In practice, these are derived from an n-byte seed using a pseudo-random generator. The corresponding public key is derived by applying a so-called chaining function F to the values in private key $w - 1$ times. The result consists of the ℓ chain heads (i.e. the last computed nodes) of n bytes each.

This public key can be compressed to an n-byte value by interpreting the ℓ heads as leaf nodes of a hash tree. Note that this tree is almost a binary tree; this so-called ℓ-tree is constructed by hashing two neighboring nodes to construct a parent node on a higher layer, and simply raising the last node to the next layer

[2] One could consider $w = 4$, to speed up the computation at the cost of additional signature size. While the RFC [10] does not specify a specific parameter set, it does explicitly mention $w = 4$ as an option for this purpose.

if the number of nodes on that layer is odd. The root of this tree is the de facto WOTS$^+$ public key, and we will refer to it as such.

Signatures. Assuming an n-byte message m, this is split into ℓ_1 chunks of $\log_2(w)$ bits, which are interpreted as integers m_1 to m_{ℓ_1}. The chaining function F is then applied m_i times to the i-th value of the private key, and the output is included as part of the signature. Given a message m and such outputs, verification means completing the chains by applying the chaining function an additional $w - 1 - m_i$ times and checking that these values combine to form the public key.

The careful reader will have noticed two issues: there were ℓ (and not ℓ_1) chain heads that make up the public key, and given a signature on a message m, it is easy to forge a signature for some messages m' by simply applying the chaining function $m'_i - m_i$ times[3]. To remedy this, WOTS$^+$ signatures include a checksum, computed by signing the base-w representation of $C = \Sigma_{i=1}^{\ell_1}(w - 1 - m_i)$, i.e. $(C_1, \ldots, C_{\ell_2})$. This prevents forgery, since an increase in any m_i results in a decrease in C and thus at least one C_j.

Functions. In the above description, we have left the chaining function F unspecified, and have not defined how the hash tree is constructed. These functions are instantiated using a tweaked variant of SHA-256 in the parameter sets we consider in this work[4]. To ensure collision resilience and to mitigate multi-function and multi-target attacks [13], each application of this function not only hashes the above-described input, but additionally includes a domain separator, a unique 'address' and a key, as well as applying a mask.

For ease of exposition, we omit the specifics of these constructions here, and only touch upon the relevant aspects in Sect. 5.

2.2 Hash Trees

Having established WOTS$^+$ as a one-time signature scheme, we now expand this into the many-time signature scheme XMSS [4]. In essence, XMSS consists of many instances of WOTS$^+$ and a hash tree to authenticate them.

Keys. Consider a binary hash tree of height h, i.e., a tree with 2^h leaf nodes. We associate a WOTS$^+$ key pair with each leaf, allowing for 2^h signatures. The XMSS private key simply provides a seed from which to generate the WOTS$^+$ private keys; the WOTS$^+$ public keys are then derived by applying the chaining function, as described above. Then, by computing a binary hash tree on top of the WOTS$^+$ public keys, one derives the XMSS public key: the root node of the tree.

Signatures. By the above construction, it is straight-forward to see that an XMSS signature mostly consists of a WOTS$^+$ signature, complemented by an

[3] This requires that $m'_i \geq m_i$ for all i, but this is sufficiently likely even for random m.

[4] A common and often more natural instantiation relies on the Keccak-based SHAKE [2].

index to indicate which WOTS$^+$ private key was used. However, the verifier does not hold the corresponding WOTS$^+$ public key required to verify the signature. Instead they compute what the WOTS$^+$ public key should be, based on the presented signature – note that this process is exactly the same as verifying a WOTS$^+$ signature, omitting actual comparison to the public key. This effectively gives the verifier one leaf node in the hash tree. To compare to the root node (i.e. the XMSS public key), the verifier requires nodes along the path to the root of the tree. This path is referred to as the 'authentication path', and it can be seen that the signer must include h additional nodes. See Fig. 1.

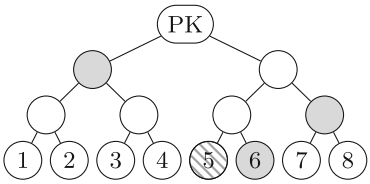


Fig. 1. The authentication path to authenticate the fifth leaf is shown in grey [12].

The State. When introducing the notion of many WOTS$^+$ key pairs linked to one XMSS public key, it is crucial to prevent re-using these one-time key pairs. Conceptually, this is trivially accomplished by iterating through the leaf nodes sequentially. It is important to remark, however, that this implies maintaining a persistent *and changing* state across different signing operations. Depending on the specific usage scenario, this may or may not be a problem – it adds complexity in settings where a key is used by multiple processes or needs to be kept in sync across servers, but is not a concern when a key is embedded in a single system. The latter scenario describes the use case discussed in this paper. In [6], the authors demonstrate that there is a strong synergy between this property and achieving forward security (but this is not part of the 'standard' XMSS scheme).

2.3 Chaining Trees

In order to be able to perform many signatures using the same public key, one could instantiate XMSS with a large tree. This comes at a considerable cost, as the signer needs to compute all leaf nodes when generating a signature. Specialized tree traversal algorithms [5] move a large part of this cost to the key generation, but it remains a limiting factor. This is mitigated in XMSS^{MT}, the multi-tree variant of XMSS. As the name suggests, this scheme makes use of a structure of trees.

On the bottom layer, a WOTS$^+$ key pair is used to sign the message. Along with the WOTS$^+$ signature, the signer supplies an authentication path to the root of that subtree. Rather than interpreting this root as the public key, it is signed using a WOTS$^+$ leaf of a new tree, one layer 'above' the current layer. This signature is authenticated by a path leading to the next root node, et cetera.

Considering d layers of trees of height h/d, this allows for 2^h signatures while only requiring h/d leafs on each layer to be computed to construct the authentication path (as well as opening up a whole new range of time-memory trade-offs with tree traversal [5]).

Note that this trade-off leads to increased signature sizes. While an XMSS signature consists roughly of a WOTS$^+$ signature (i.e. $67 \cdot 32$ bytes) and a number of intermediate nodes (say, $20 \cdot 32$ bytes), an XMSS^{MT} signature consists of multiple WOTS$^+$ signatures. For the sake of simplicity, we now consider XMSS^{MT} to be a direct generalization of XMSS, i.e. XMSS is the specific class of instances where $d = 1$.

3 Java Card Platform and Limitations

Java Card defines a standardized, vendor-independent programming platform for multi-application smart cards produced by different manufacturers. While the specification is controlled by Oracle, many of the large smart-card manufacturers[5] collaborate in the 'Java Card Forum' [8] in defining the platform. The platform has proven popular, with over 20 billion cards sold at the time of its twentieth anniversary in 2016 [20]. Java Card is often found in SIM cards and passports.

As the name suggests, Java Card is based on Java, but with many language features restricted due to the limited resources. This shows prominently in the limited availability of types – a Java Card platform is only required to support 8-bit `bytes` and 16-bit `short`s. Similarly, Java Card inherits the class-based object-oriented style of Java, but using objects is discouraged because of size constraints; moreover, garbage collection is optional for Java Card.

The APIs for Java and Java Card differ vastly. The Java Card API is extremely limited, but does provide a range of high-level methods for standard cryptographic use cases (e.g. signature generation, key storage, block encryption). This enables developers to quickly construct applets to perform basic cryptographic operations. The implementation of the API is left to the smart-card manufacturer, allowing implementations in native code or directly in hardware. This is crucial for performance: the Java Card VM introduces considerable overhead, so implementing cryptographic primitives in Java Card bytecode would be unacceptably slow. Still, considerable overhead remains when calling these API functions, and this turns out to be a recurring theme in the rest of this paper.

An important consideration is the limited amount of memory. Typical Java Cards have in the order of tens of KiB persistent memory (EEPROM or Flash), but the transient (RAM) memory is typically only a few KiB, which is a serious bottleneck. Memory sizes can vary significantly between cards, so memory requirements should be carefully taken into account when developing applications.

[5] At the time of writing, the Java Card Forum consists of Gemalto, Giesecke & Devrient, IDEMIA, Infineon, jNet ThingX, NXP Semiconductors and STMicroelectronics [8].

Java Card is compatible with the ISO 7816 standard. This means that communication is done using APDUs (Application Protocol Data Units). These traditionally support a payload of up to 256 bytes, although recent cards support extended-length APDUs allowing longer payloads.

In this work, we focus on compatibility with Java Card version 2.2.2 to 3.0.4.

3.1 Considerations for the OpenVPN Use Case

This work was done as part of a project involving a Java Card applet to provide authentication when establishing a VPN connection, tightly integrated into OpenVPN. The projected benefit of this was twofold: increased security and increased usability. Smart cards typically provide much more secure storage of the key material. By selecting the Java Card platform, the cross-platform applet can be easily combined with existing deployed systems. The tight integration with OpenVPN aims to improve the user experience: we avoid third-party middleware (which would be required for the use of more generic solutions, such as hardware tokens relying on standards like PKCS#11) and store the configuration files for OpenVPN on the card to simplify the setup process for the user.

This use case implies a set of assumptions and limitations. There is some margin in terms of signing time, as signing operations are fairly infrequent and users would expect some latency when establishing a connection. More importantly, the required throughout is low: after signing once, typical usage scenarios suggest a period of time during which the card is connected and powered, but not used to produce a new signature. Furthermore, we note that key generation can be done during issuance, and even outside of the card (assuming a secure issuance environment – this is a reasonable assumption given that initialization also involves, e.g., PIN codes). In principle, there is a nice match between these properties and the XMSS^{MT} signature scheme. There are many time-memory trade-offs that can be flexibly tweaked, and there is ample opportunity for precomputation either during key generation or idle time. However, it is important to reiterate that memory (in particular the fast RAM) is a scarce resource on the card. The next section details these trade-offs.

4 Implementation

When designing a smart card application, it is important to consider natural 'commands' that divide up and structure the computation. For a traditional RSA-2048 or ECC signature, signing a message could be a single command with a single APDU as response. For XMSS^{MT}, signatures are several kilobytes in size and must be spread out over multiple 256-byte response APDUs. This behavior is typical for hash-based signatures on small devices [12]; they are too large to comfortably fit in RAM but are very sequential in their construction, strongly suggesting an interface where the signature is streamed out incrementally.

There is much repetition of small subroutines to be found in the scheme. After initializing the signing routine by computing a message digest, a signature

consists of a sequence of WOTS$^+$ signatures and authentication paths. Internally, the WOTS$^+$ signatures can be decomposed further into their separate chains. The $\ell = 67$ chains split naturally into 8 sets of 8 chains for the $\ell_1 = 64$ message digest chains, and $\ell_2 = 3$ chains for the checksum. For hashes of 32 bytes and $h/d \leq 8$, authentication paths within a subtree fit into one response APDU, and choosing $h/d > 8$ is not realistic on this platform because of resource constraints[6]. In order to reduce the latency of signature generation, we ensure that all relevant leaf nodes for the authentication path in each subtree on each layer are available in memory. We address this later in this section, and for now only note that maintaining this invariant introduces a preparation step after a signature is produced (and thus: a leaf node is consumed).

Figure 2 represents these states visually. Note that each state is triggered by a command APDU, of which only the initial command contains auxiliary data (i.e. the message). These have been omitted for simplicity.

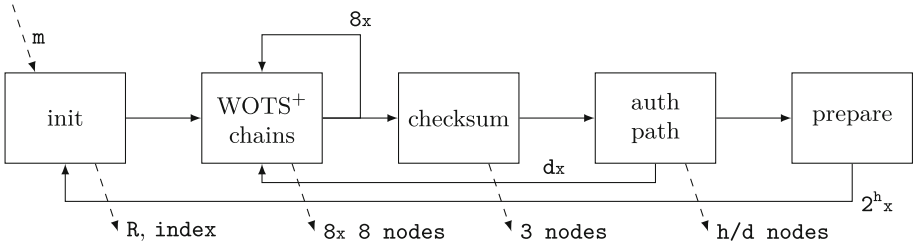


Fig. 2. State diagram of the signing routine.

Indices. As it is crucial that the smart card cannot be coerced into re-using a leaf, the first operation should be incrementing the state index. Because Java Card does not guarantee a native 32-bit integer type, all indices are stored as tuples of two `shorts`, interpreted as 15-bit unsigned values (effectively ignoring the high bit). As a consequence, atomic increments are not possible without use of expensive transactions, and special care has to be taken in case of overflows – the conservative approach skips 2^{15} leafs in case of card tear[7], rather than rolling back. Similar considerations apply when deriving indices of 'next' and 'previous' nodes during state generation. As we have limited h/d previously, internal tree indices can be represented with a single `short`.

WOTS$^+$ Leaf Generation. To generate a WOTS$^+$ leaf, an ℓ-tree must be computed over the heads of all chains. As memory is limited, the natural choice here is to use the treehash algorithm [16, Sect. 7]. Since $\ell = 67$ is not a power of 2, bringing the tree out of balance, there are some special cases to consider. As

[6] This would imply either computing or storing hundreds of WOTS$^+$ leaf nodes per tree layer.

[7] The physical attack of interrupting the power supply to the card, e.g., by removing it from the reader.

the value of ℓ is constant for all parameters we account for, this can be simplified by manually handling these special cases after performing treehash. Altogether, this ensures that we require only 416 bytes of RAM for intermediate results when deriving a WOTS$^+$ public key.

State (Re)generation. At least one WOTS$^+$ computation needs to be performed whenever a message is signed: exactly when signing the message digest. Without proper state management, however, one would be required to compute $d \cdot 2^{h/d}$ WOTS$^+$ leafs to derive the authentication paths. Instead, we keep a persistent array of the leaf nodes of the current tree on each of the d layers. If the secret key is generated off-card, the leaf nodes of the first trees can be preloaded; alternatively, they can be computed during issuance. Similarly, the $d - 1$ WOTS$^+$ signatures that join the subtrees together can also be precomputed and cached. By keeping an additional array of such nodes and signatures for the 'next' tree on every layer[8] and computing one new node whenever one is consumed, it can be easily seen that we are guaranteed to always have all leafs available before they are consumed. Note that this introduces an imbalance in signing time cost (as consuming indices that introduce new nodes on multiple layers adds linear leaf generation cost), but that this computation be performed *after* outputting a signature. Careful administration is required to guarantee that this is not neglected. Intuitively, one might consider decoupling the signing and preparation step, and allow the signature routine to effectively consume the nodes up to the point at which they were prepared. While this is certainly possible, the involved bookkeeping is more complicated than it may seem at first: memory requirements imply re-using arrays, the leafs currently in use cannot be overwritten, and the next layer of leafs needs to be completed precisely when switching to the next subtree. Verifying these conditions combines poorly with the convoluted arithmetic on tuples of `short`s that represent indices.

4.1 Hash Functions

Performance is dominated by the cost of a call to the chaining function in WOTS$^+$ and the hash function in the binary trees. In essence, these functions consist of many applications of SHA-256 to small arrays of data (i.e. 32 to 128 bytes) and some `xor` operations. This is not a particularly common pattern of operations in traditional cryptography – a signature operation typically requires just one hash function call to digest the message, often negligible in the overall performance of the signing operation. Note also that there is significant cost associated with a single call to a hash function that is constant in the length of the input, likely representing the overhead of the function call, as shown in Table 1.

AES-Based Hashing. Instead of using a cryptographic hash function as a building block for the described functions, a block cipher can be used to construct a similar primitive using common constructions such as Davies-Meyer and

[8] This is only required on layers where there is still a 'next' tree, which is trivially false for the top-most tree.

Table 1. 1000 iterations of SHA-256

Data (bytes)	32	64	128	256
Runtime (seconds)	3.94	5.83	8.02	12.40

Table 2. 1000 iterations of AES-128 in ECB mode

Data (bytes)	16	32	64	128	256	1024
Runtime (seconds)	2.97	3.30	3.96	5.30	7.97	23.87

Matyas-Meyer-Oseas (the latter being used in [6]). Some care would need to be taken to transform these to a security level equivalent to the second pre-image resistance derived from SHA-256 in the context of XMSS^{MT}. This would break compatibility with the RFC [10], but in principle this is not unsurmountable.

Some Java Cards appear to be equipped with an AES implementation in hardware, speeding up its performance significantly. This is evidenced by an even larger unbalance between constant and variable costs: encrypting large blocks of data is only marginally more costly than smaller blocks, as shown in Table 2. The base cost of a single call to AES is still significant, however, putting the performance in the same ballpark as SHA-256 on short inputs. Note that these numbers cannot be directly compared to the cost of SHA-256 as listed in Table 1, as multiple iterations of AES would be required for one compression block.

There is another avenue to explore when relying on AES as a primitive, as the Java Card API supports a range of modes of operation for AES. Combining this with the fact that we process a large amount of data at once suggests opportunities for parallel data streams; encrypting a large data stream using AES in ECB mode is functionally equivalent to performing independent AES encryption in parallel – under the same key. This last restriction is crucial, as a message-dependent block is used as key, ruling out precisely the constructions available to turn AES into a compression function. Other modes of operation suffer a similar faith. As a result, there is no clear way to exploit the available AES implementation for parallel data streams.

4.2 Memory Usage and Benchmarks

This section outlines the performance when running the applet on a Java Card. For this, we performed measurements and ran tests on NXP-produced JCOP cards, as well as a card of unclear origin (`ICFabricator=0005`). While this is somewhat indicative of relative performance, we note that measurements may vary wildly when comparing different cards by different manufacturers. Tables 1 and 2 give the individual benchmarks for the primitives on the cards we used.

For a WOTS$^+$ signature operation with the parameters described in Sect. 2.1, we measure an average time of approximately 33 s. In the best case, the preparation step requires one WOTS$^+$ key generation, which requires approximately a minute.

When we consider a realistic parameter set, where $h = 20$ and $d = 4$, i.e. subtrees with 32 leaf nodes, we notice that the cost of authentication path generation starts to come into play. In particular, the access to nodes stored in persistent memory makes this more costly than a back-of-the-envelope computation would predict[9]. For these parameters, a signature takes roughly 54 s in the best case: every 32nd signature adds an additional WOTS$^+$ signature generation, every 256th signature adds two WOTS$^+$ signatures, et cetera. Similarly, preparation takes 85 s in the best case. Varying to $d = 5$ results in a slightly shorter signing time, coming in at 50 s in the best case (but more frequently requires new WOTS$^+$ signatures).

Besides a small number of bytes to store the keys and index, the requirements on persistent memory follow from the storage of WOTS$^+$ signatures and leaf nodes: $32 \cdot \ell \cdot (d-1)$ bytes for the WOTS$^+$ signatures, and $32 \cdot (2 \cdot d - 1) \cdot 2^{\frac{h}{d}}$ bytes for the leaf nodes. For $d = 4$, this comes down to $6432 + 7168 = 13600 = 13.28$ KiB. Similarly, for $d = 5$, this adds up to $8576 + 4608 = 13.18$ KiB. Note that increasing d also increases signature size by additional WOTS$^+$ signatures, but decreasing d while maintaining $h = 20$ sharply increases the memory requirements for node storage, as well as the cost of (off-card) key generation.

Considering the signing states described in Sect. 4, in particular in Fig. 2, it can be easily seen that the signature is output in stages as computation progresses. With the WOTS$^+$ chain computation taking up most of the computation, splitting this over eight APDUs levels out communication costs.

5 Java Card API Recommendations and Considerations

In the previous section, we touched upon several issues with implementing XMSS^{MT} (and hash-based signatures in general) using the current Java Card API (i.e. version 3.0.5 or below). This section discusses potential extensions to improve support for hash-based signatures. In the past the Java Card API has been extended to support new cryptographic algorithms[10]. If and when hash-based signatures become widely used in the future, one would expect extensions of the API for this, either as proprietary extensions of manufactures or ultimately as extensions of the standard.

An important design choice in such an API is the level of abstraction. One can opt for low-level methods providing more fine-grained (i.e. more primitive) operations, or for higher levels of abstractions, where the API methods provide bigger building blocks, or possibly even a complete signatures scheme[11]. Below we present four alternatives with an increasing level of abstraction.

[9] A WOTS$^+$ signature costs 536 applications of the chaining function on average, versus 63 hash function calls in the tree.

[10] For example, version 3.0.5 introduces support for SAC/PACE [3,14], a protocol used in electronic passports.

[11] For example, in the case of the PACE protocol, the choice has been made not to provide a generic API method for elliptic curve point addition, which would enable applet developers to implement PACE, but rather to provide more higher-level operations to directly provide PACE as primitive.

Generally speaking, a more fine-grained API is likely to be easier to implement for manufacturers and offers more flexibility to applet developers. On the other hand, higher level, more monolithic API methods make it easier for developers that are less versed in the relevant cryptography to make the correct choices, allow for faster implementation in hardware, and enable manufacturers to provide more comprehensive side-channel countermeasures. Also, an API implementation may need memory to record state between API calls and scratchpad memory to record temporary results. Given that transient memory is extremely scarce, it is not acceptable that API methods need large amounts of RAM.

Another factor to take into consideration when making an abstraction level trade-off is the fact that standardization efforts are still ongoing, and a high-level API leads to less agility to account for future scheme changes.

5.1 Parallel Hashing

Performance of hash-based signatures is completely dependent on the ability to efficiently compute many hash digests over small amounts of data. While this can be sped up by implementing the hash function in hardware, Sect. 4.1 illustrates that this is only part of the solution. More critically, the execution time depends on being able to exploit the extreme levels of parallelism that are inherent to hash-based signatures. Here parallelism does not necessarily imply parallel execution, but rather independent parallel data streams.

The current interface to hash functions is provided in the form of the `MessageDigest` class. After instantiating an object for a specific digest function, say SHA-256, a user can add additional data by calling the `update(byte[] inBuff, short inOffset, short inLength)` method, and obtain the final digest by calling `doFinal(byte[] inBuff, short inOffset, short inLength, byte[] outBuff, short outOffset)`.

We propose duals of these methods, following an almost identical API: `updateParallel(byte[] inBuff, short inOffset, short inBlockLength, short numberOfBlocks)`, and `doFinalParallel(byte[] inBuff, short inOffset, short inBlockLength, short numberOfBlocks, byte[] outBuff, short outOffset)`. Here, `inBuff` provides `numberOfBlocks` sequential inputs of `inBlockLength` bytes, and output is written to `outBuff` analogously.

Providing an inconsistent number of inputs (i.e. different `numberOfBlocks`) for `update` and `doFinal` calls could be treated as an error but it may be beneficial to instead fix the `numberOfBlocks` at the time of construction of the `MessageDigest` object. For hash-based signatures this decision is equivalent, as the relevant hash function calls all require arguments of the same form. Both options have serious effects on the underlying implementations, as these modifications suggest maintaining a (runtime-determined) number of intermediate hash function states. If this proves to be infeasible, a natural restriction would be to drop the parallel `updateParallel` method[12]. While this reduces flexibility

[12] It is also possible to reach a similar invariance by fixing `numberOfBlocks`, but this still requires multiple hash-function intermediate states in transient memory.

for the applet developer, in particular when memory is constrained and rearranging input is costly, this allows underlying hardware to sequentially process each instance of the hash function without maintaining a variable-length intermediate state in addition to the caller-provided input and output buffers. This does not contradict the goal of achieving a speedup through internal parallelism, as the majority of the cost can be attributed to the Java stack on top of the underlying implementation (see Sects. 4.1 and 4.2). As a result, implementations that would support a parallel `update` method would still likely opt for a sequential underlying hashing primitive to reduce area cost.

5.2 Complete WOTS$^+$ Chains

Rather than providing a narrow API that allows a developer to efficiently make use of the underlying hash function primitive, the parallelism can be made transparent to the implementer through a more abstract API: computing WOTS$^+$ chains and authentication paths.

In the WOTS$^+$ chains, there is a lot of opportunity for shared execution. Besides the natural 'horizontal' parallelism across many chains (which would be the primary candidate for optimizations discussed in the previous Subsect. 5.1), there is potential gain in coupling the 'vertical' computations that take place during WOTS$^+$ public key and signature generation. On top of the benefits achieved from only passing through the Java stack once, rather than repeatedly for every application of the chaining function, the input to many of the underlying SHA-256 compression function calls overlaps significantly. In particular, for a single WOTS$^+$ key pair, the input to the first compression call is completely identical across all $16 \cdot 67 = 1072$ calls of the chaining function. Note that this is a consequence of the specific instantiation of the compression function in XMSS^{MT} as defined in [10], however, and does not immediately carry over to other function designs (in particular, the SPHINCS+ proposal [1] to the NIST standardization project [17] does not benefit from this).

Such an API goes beyond a straight-forward parameter for the hash function specifying the number of iterations, as the iterated function is not simply SHA-256, but rather the address- and key-aware chaining function. Furthermore, to make it effective for WOTS$^+$ signature generation, it would require specifying the length of each individual chain, as well as a variable number of chains (as an entire WOTS$^+$ signature will likely not fit in RAM on most Java Cards).

This middle ground between abstracting away the parallelism of the hash functions but still requiring (or, indeed, allowing) the developer to puzzle together the pieces has its upsides, but is clearly not without added complexities. We stress that the API of such a hybrid solution needs to be carefully thought through to be sufficiently fine-grained to provide a benefit over an all-in-one API (as described later, in Sect. 5.4), yet convenient to use so that it actually reduces boilerplate code and development overhead when compared to a more straight-forward parallel hashing API.

5.3 WOTS$^+$ Nodes and Hash Trees

Another unit of abstraction is a hash tree. In XMSS^{MT}, there are two specific instances of hash trees: the tree in XMSS, and the ℓ-tree in the WOTS$^+$ nodes.

The computation of WOTS$^+$ nodes can be hidden behind an API with relative ease. Given its position in the hypertree and the secret seed, the only relevant output is the root node of the ℓ-tree, easily fitting a single APDU (and thus appropriate as the result of a single function call). We note that in the SPHINCS+ proposal, ℓ-trees have been eliminated altogether. It is not inconceivable that future updates to XMSS will include the same change.

Abstracting the hash trees in the hypertree behind a single function is somewhat more complicated. The reason for this is twofold. First and foremost, preventing recomputation of such trees is crucial to make XMSS^{MT} practical, which implies carefully maintaining a state (either by storing leaf nodes, as is done in the current implementation, or through more involved tree traversal techniques [5]). This introduces a time/memory trade-off that strongly depends on the parameter choice – allowing more flexibility in terms of tree height and multi-tree depth significantly increases complexity of the underlying implementation. Secondly, as the relevant output comes in the form of an authentication path of multiple nodes, APDU size (and thus state machine management) becomes relevant as soon as $h > 8$.

Conversely, there is much to gain in terms of simplicity for the user if this is abstracted, as this prevents the users from having to re-implement the treehash algorithm and make complex state management decisions. We argue that this is a crucial requirement for non-expert usage.

5.4 Complete XMSS^{MT} Signatures

At the end of the spectrum, we consider an API that abstracts away as much of the scheme's internals as possible. Intuitively this matches the current approach of the Java Card API for public-key primitives; given a parameterized and keyed object and a message, there is a single API call that produces a signature. To allow for longer messages, an `update` mechanism is available similar to how message digests work (see Sect. 5.1). Crucially, this is made possible by the small size of signatures; for typical parameters, the resulting signature easily fits in RAM and even in a single output APDU.

When considering the multiple kilobytes of a typical XMSS^{MT} signature, such an API suggests writing the signature to persistent memory. This requires additional EEPROM/Flash and adds the extra cost of slow memory access. However, this is likely to compare favorably when considering the potential for performance improvement by implementing the entire scheme natively.

Alternatively, the API could be split up in a similar way as is done in this implementation; we refer to the states described in Fig. 2 – each state could represent an API call. This would still require the applet developer to implement the state machine, but makes conversion to output APDUs more natural.

Perhaps the most compelling argument for this high-level API is usability for applet developers. XMSS^{MT}, and tree traversal in general, is administratively notoriously tedious, and wrongly managing indices can easily lead to degraded security. In particular, a high-level API is required to properly abstract the state preparation step, as this would otherwise heavily depend on implementation choices (i.e. what part of the state is cached, and how it is iterated). Ease of use should not be underestimated as a critical factor towards adoption in real-world applications.

5.5 Side-Channel Countermeasures

Smart cards are a common target for physical attacks. To remedy this, manufacturers commonly implement a wide variety of platform-specific countermeasures. An API that abstracts away the usage of secret data is paramount for this to be effective. This requirement aligns well with the considerations of the rest of this section when considering the simplicity of the API exposed to the applet developer: a fine-grained API that requires the developer to implement the overarching scheme creates many potential pitfalls. To illustrate, the current lack of API required us to abuse the `AESKey` object to store sensitive key material in EEPROM, extracting it into RAM before use (although more recent versions of Java Card provide the `SensitiveArray` class for this purpose). Similarly, without API support, the expanded WOTS^{+} seeds live plainly in transient memory. While in general hash-based signatures have a history of robustness against side-channel attacks, it is precisely this usage of the PRF that has recently been under scrutiny [15].

References

1. Bernstein, D.J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.-L., Hülsing, A., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P.: SPHINCS+. Submission to NIST's post-quantum crypto standardization project (2017). https://sphincs.org
2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference, January 2011. http://keccak.noekeon.org/
3. Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token. Technical report TR-03110, German Federal Office for Information Security (BSI), Version 2.20 (2015)
4. Buchmann, J., Dahmen, E., Hülsing, A.: XMSS - a practical forward secure signature scheme based on minimal security assumptions. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 117–129. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25405-5_8. https://eprint.iacr.org/2011/484
5. Buchmann, J., Dahmen, E., Schneider, M.: Merkle tree traversal revisited. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 63–78. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88403-3_5. https://www.cdc.informatik.tu-darmstadt.de/reports/reports/AuthPath.pdf
6. Hülsing, A., Busold, C., Buchmann, J.: Forward secure signatures on smart cards. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 66–80. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35999-6_5. https://huelsing.files.wordpress.com/2013/05/xmss-smart.pdf

7. Eurosmart: Digital security industry to pass the 10 billion mark in 2018 for worldwide shipments of secure elements. Press Release (2017). http://www.eurosmart.com/news-publications/press-release/296
8. Java Card Forum: About the JCF (2018). https://javacardforum.com. Accessed 12 Mar 2018
9. Hülsing, A.: W-OTS+ – shorter signatures for hash-based signature schemes. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 2013. LNCS, vol. 7918, pp. 173–188. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38553-7_10. https://eprint.iacr.org/2017/965
10. Hülsing, A., Butin, D., Gazdag, S.-L., Rijneveld, J., Mohaisen, A.: XMSS: eXtended Merkle Signature Scheme. Request for Comments 8391 (2018). https://tools.ietf.org/html/rfc8391
11. Hülsing, A., Rausch, L., Buchmann, J.: Optimal parameters for XMSS^{MT}. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.) CD-ARES 2013. LNCS, vol. 8128, pp. 194–208. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40588-4_14. https://eprint.iacr.org/2017/966
12. Hülsing, A., Rijneveld, J., Schwabe, P.: ARMed SPHINCS – computing a 41 KB signature in 16 KB of RAM. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9614, pp. 446–470. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_17. https://eprint.iacr.org/2015/1042
13. Hülsing, A., Rijneveld, J., Song, F.: Mitigating multi-target attacks in hash-based signatures. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9614, pp. 387–416. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_15. https://eprint.iacr.org/2015/1256
14. Supplemental Access Control for Machine Readable Travel Documents. Technical report, International Civil Aviation Organization (ICAO), Version 1.1 (2014)
15. Kannwischer, M.J., Genêt, A., Butin, D., Krämer, J., Buchmann, J.: Differential power analysis of XMSS and SPHINCS. In: Fan, J., Gierlichs, B. (eds.) COSADE 2018. LNCS, vol. 10815, pp. 168–188. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89641-0_10. https://kannwischer.eu/papers/2018_hbs_sca.pdf
16. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_21. www.merkle.com/papers/Certified1979.pdf
17. NIST: Post-quantum cryptography: NIST's plan for the future (2016). http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/pqcrypto-2016-presentation.pdf
18. Rohde, S., Eisenbarth, T., Dahmen, E., Buchmann, J., Paar, C.: Fast hash-based signatures on constrained devices. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 104–117. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85893-5_8. https://www-old.cdc.informatik.tu-darmstadt.de/reports/reports/REDBP08.pdf
19. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing, pp. 387–394. ACM (1990). https://www.cs.princeton.edu/courses/archive/spr08/cos598D/Rompel.pdf
20. Safran Identity & Security: The impact of Java Card technology yesterday and tomorrow: Safran Identity & Security celebrates 20 years with the Java Card Forum. Press Release. https://www.morpho.com/en/media/impact-java-card-technology-yesterday-and-tomorrow-safran-identity-security-celebrates-20-years-java-card-forum-20170302. Accessed 12 Mar 2018

Detecting Privacy Information Abuse by Android Apps from API Call Logs

Katsutaka Ito[1(✉)], Hirokazu Hasegawa[2], Yukiko Yamaguchi[3], and Hajime Shimada[3]

[1] Graduate School of Informatics, Nagoya University, Nagoya, Japan
itokatu@net.itc.nagoya-u.ac.jp
[2] Information Strategy Office, Nagoya University, Nagoya, Japan
[3] Information Technology Center, Nagoya University, Nagoya, Japan

Abstract. In these years, the use of smartphones is spreading. Android is the most major smartphone OS in the world, and there are a lot of third-party application stores for Android. Such third-party stores make it easy to install third-party applications. However, these applications may access and obtain privacy information, in addition to their major functions. There is a survey showing that most users do not take good care of the settings about how their privacy information is handled by applications. Thus, privacy information abuse by authorized application is becoming a serious problem. In this paper, we propose a method to detect applications that access privacy information unrelated to their functionalities by analyzing API call logs, which can reveal the activities of the application. In order to record API call logs, we modified the Android source code, and run the rebuilt system on an emulator. We analyzed applications' API call logs with a statistical method, based on the frequency of privacy information accessing and network activities.

Keywords: Android · API call logs · Privacy information leakage

1 Introduction

As the use of smartphones becomes more general year by year, the users can improve their life with various types of applications. As reported in International Data Corporation's report about the smartphone OS share [1], Android occupies about 85% in the World. Different from iPhone, Android smartphones can use many third-party application stores such as 1mobile market[1] and stores served by cellular company easily. Even if the users have poor knowledge about application security, they can install and utilize various applications freely.

As reported in Information-technology Promotion Agency's report [2], about 90% smartphone users are fearing about the leakage of privacy information. However, only less than 20% users are taking care of the privacy information

[1] http://www.1mobile.com/.

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 143–157, 2018.
https://doi.org/10.1007/978-3-319-97916-8_10

handling manifests shown at installation or the first launch of applications. It shows that many people do not pay attention to handling of privacy information in spite of fear. Meanwhile, there are applications which aim to gather privacy information regardless of their functionality, especially in third-party application stores. Those applications usually obtain authentic permission to privacy information so that those applications can gather privacy information freely with using official Android APIs. Many people may give permissions to those types of applications with no care because they do not confirm manifests as written by above report. Protecting privacy information from such types of leakage is becoming important problem.

In this paper, we propose a method to detect those type of applications based on their behavior. By using API call logs, we can categorize the activity of application to several categories such as whether application access to privacy information, drawing something on display and so on. In our proposal, we firstly classify activities of applications into 4 types such as access to privacy information, network activity, providing information to users, and others. Then, we classify application into 3 classes (No-Access, Proper-Access, and Improper-Access) and calculated statistical value from the number of above activities. We evaluated 42 of Android applications which we collected at third-party stores.

2 Related Works

There are some researches which aim for detecting and protecting leakage of privacy information from Android applications. Kim et al. proposed a method for detecting leakage of privacy information by Dalvik bytecode of Android applications [3]. They defined analysis rules for Dalvik bytecode in order to decide whether applications are malicious or not. Their method successfully detect applications which aim to obtain privacy information in Google Play Store. Hosoya et al. revealed what permission is required by applications and what Android APIs are used generally through survey on manifest file [4]. They alerted that there are some applications which trace user through terminal ID. Mann et al. proposed a framework for static detection of privacy leaks [5]. They implemented a framework for tracking information flows based on arguments of API calls which can touch privacy information with static analysis on Dalvik bytecode. They firstly define privacy policies for API to generate privacy policies to enforce which includes what field of API return values include privacy information. Then, they evaluate bytecode of applications with privacy policies to enforce in the proposed system and obtain reports.

There are also some researches which aim to detect leakage of privacy information through static analysis. Han et al. proposed a method for detecting malicious applications from system-calls and API calls related with terminal information (e.g. IMEI number) and detecting malicious applications from K-means machine learning [6]. In spite of the existence of information which we have to protect from leakage (e.g. address book), the proposed method is only concerned about terminal information. Hatada et al. proposed a detection method of

PUA(Potentially Unwilling Applications) from DNS queries [7]. In their experiment, they extracted FQDN from DNS request of applications and calculated the similarity of FQDN lists between two applications for all applications. They revealed that PUAs request much more FQDN than normal applications and PUAs tend to access to particular FQDN as same as Android OS accesses. They also revealed that black list based detecting is less effective in detecting PUAs. Do et al. proposed a protection method of privacy information leakage by removing permission already accepted in application installation [8]. In the research, they removed permissions for an application which is unrelated for specific users by decompilation of apk file. They remove needless permissions by recompilation of manifest file and smali code in the apk file. Gibler et al. proposed statically automatic analysis method of Android applications for privacy leakages [9]. They proposed an analysis framework for finding potential leakages by mapping relationship between Android API and permissions in application analysis.

There is a research to protect privacy information leakage by modifying files in apk files and bytecode. Zhang et al. proposed a method to track sensitive information flows that are potentially involve in information leakage by inserting bytecode instructions for application program [10]. This work aims to achieve both flow-based security for privacy information and low run-time overhead. They implemented an application-wide static dataflow analysis prototype based on the Java bytecode optimization framework.

There are some researches which indicate vulnerabilities of Android framework. Davi et al. shows the privilege escalation attack on Android applications without permission [11]. They showed an application which does not have privilege can obtain privilege through other application by exploiting serialized transitive permission usage among three applications. Chen et al. implemented a method to detect system vulnerabilities with an architectural approach [12]. Their proposal is organized with online attack detector and offline vulnerability locator linked by a record and replay mechanism. Their system checks the change of data flow (source or destination address) to detect memory vulnerabilities. Chen et al. also tried to apply this method to inner Android frameworks and implemented a live patching system against kernel vulnerabilities of Android [13].

In our research, we propose a method to detect the leakage of privacy information through unmannered applications by calculating statistics of categorized API calls under application execution. Our method is base on statistical analysis whose calculation cost is lighter than that of machine learning in classification, so that it can achieve low run-time overhead.

3 Proposed Method

3.1 Threat Model of This Work

Android operating system has permission framework for applications. Applications which access to privacy information need to ask user whether they give

permission to applications or not. Applications which are aiming to gather privacy information also ask users whether applications may access to privacy information even if those permissions are mostly unrelated with their functionality. If the user gives authority to those types of applications, they abuse given permission to access to unrelated privacy information with no hesitation by using legitimate Android APIs. After those types of application gather privacy information, they may send the information to servers. However, a lot of people do not care for the manifest of applications shown at installing or the first launch as mentioned at Sect. 1 and gives permissions so that accessing to privacy information by those types of applications never become an illegal activity. As shown in Sect. 1, even for those less careful people, they has awareness about privacy information leakage. Therefore, we try to warn applications' improper activity to improve privacy for such types of people. There may be users who agrees with over privileged manifest against one's better judgement. Our proposal also can use for giving warning to those users. The users can get the other decision point whether they continue to use those applications or not.

3.2 Overview

As the number of third-party applications is large and new ones are made everyday, it is hard to evaluate all of applications' dangerousness for accessing to privacy information in advance. Thus, we propose a method to detect them when user execute them on Android terminals with a statistic-based algorithm. Firstly, we explore frequency and tendencies of API calls which is used in improper application's activities.

We show the outline of our works at Fig. 1. In our proposal, we focus on API call logs which give us information about applications' activities in order to estimate appropriateness of each application. We suppose that improper applications access to privacy information frequently and communicate with servers via network frequently. Based on the estimation, we focus on analyzing applications' activities such as accessing local data and networking. As a prototype system, we utilized Android emulator to run applications and record API call logs which is required for our proposal. Gathering activities of those types of privacy information may arise in API call so that we output them to text files for each application. After that, we analyze tendencies for API uses from API call logs by statistic-based method. Finally, we classify applications to three classes named No-Access, Proper-Access and Improper-Access, through API call logs according to their appropriateness.

We showed the steps of our proposed method as follows.

1. Install an application on Android on the emulator
2. Run application and record API call log
3. Analyze API call logs
4. Classify application according to appropriateness.

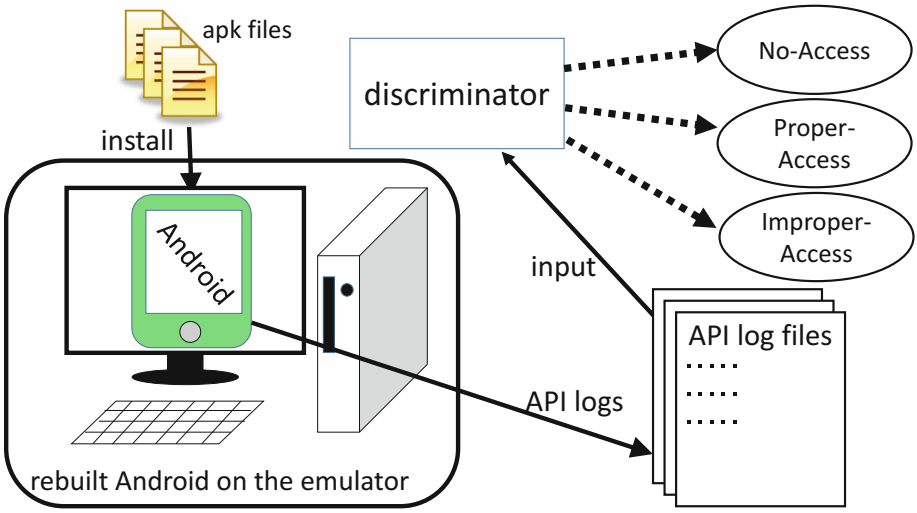


Fig. 1. Overview of this work

3.3 Classify Applications According to Appropriateness

We aim to detect the improper activities which access to privacy information. In order to estimate appropriateness of applications, we classified applications to following three types.

- No-Access
  Not access to privacy information
- Proper-Access
  Access to privacy information along with application's functionality
- Improper-Access
  Access to privacy information aiming to steal them, regardless of application's functionality

We classified applications which do not need permissions for accessing privacy information as No-Access. We classified applications which need only permissions for their functionality as Proper-Access. Note that legality for permissions depends on whether we can recognize or estimate the necessity from their functionality in current point. We classified applications which utilizes permissions not for their functionality as Improper-Access. Note that the Improper-Access includes access abuse of permissions (e.g. read all privacy information that is not connected to its functionality).

3.4 Classify Android APIs

In order to classify applications according to appropriateness of access, we classified APIs before analyzing. In this paper, we used Android 6.0 system which

is the most major version of Android OS[2]. We classify 1634 APIs belonging to "android" class into four types which are categorized as access to privacy information (ACCESS), send something to the network (SEND), display some information (RESPONSE), and the other (OTHER). Detailed relationship between categories and APIs are shown in Table 1.

We chose APIs which need permission from users and accesses to privacy information as ACCESS. They are written as "dangerous permissions" in Android reference. We also selected APIs which seems to have relation with the "dangerous" permission[3] (CALENDER, CAMERA, CONTACTS, LOCATION, MICROPHONE, PHONE, SENSORS, SMS, STORAGE permission groups) and put them to ACCESS. Based on in API logs which is obtained in preliminary evaluation, we chose APIs which are used to send something to the network as SEND. Based on in API logs which is obtained in preliminary evaluation, we chose APIs which are used when applications present information as RESPONSE.

Table 1. Proposition of API classification

Classification	Number	Example

ACCESS	24	`android.location.Location`
		`android.content.ContactContract`
		`android.hardware.Camera`
		...

SEND	3	`android.net.NetworkRequest`
		`android.net.LinkAddress`
		`android.net.LinkProperties`

RESPONSE	9	`android.view.View`
		`android.view.ViewGroup`
		`android.widget.TextView`
		...

OTHER	1598	`android.bluetooth.BluetoothDevice`
		`android.view.InputEvent`
		`android.widget.AnalogClock`
		...

3.5 Rebuilding Android to Output API Call Logs

To realize the proposed method, we have to record Android API call logs while running applications. However, we cannot know which API is called from original Android terminal. In order to record API call logs, we modified source

[2] https://developer.android.com/about/dashboards/index.html (at March 2018).
[3] https://developer.android.com/guide/topics/security/permissions.html.

code of Android which is obtained from Android Open Source Project (AOSP). Following items are procedures to add API call log recording function including rebuilding Android.

1. Download and Build
   We need Java classes of Android API body. In order to get them, we have to build Android once after download Android source code.
2. Modify
   We added code to output identifier when API has called to 1634 of Android class API body. Figure 2a shows source code of original. Figure 2b shows source code after we added code to get log into constructor as shown as red rectangle part.
3. Build again
   In order to get system image, we rebuilt Android after we modified all APIs source code.

```
public Location(String provider) {
    mProvider = provider;
}

/**
 * Construct a new Location object that is copied from an existing one.
 */
public Location(Location l) {
    set(l);
}

*-  Location.java    13% L120  Git:cac6285  (Java/l Abbrev)
```

(a) Original API code

```
 * @param provider the name of the provider that generated this location
 */
public Location(String provider) {
    mProvider = provider;
    Log.i("APILog","location.Location");
}

/**
 * Construct a new Location object that is copied from an existing one.
 */
public Location(Location l) {
    set(l);
    Log.i("APILog","location.Location");
}

--  Location.java    13% L122  Git:cac6285  (Java/l Abbrev)
```

(b) Modified API code

Fig. 2. Modification for recording API call logs

4. Run emulator with rebuilt Android
   We run Android emulator using rebuilt system image to obtain API call logs in following experiments.

4 Experimental Configuration

4.1 Environment for Collecting API Logs

We used Android 6.0 emulator which is most widely used version of Android OS. To reduce complexity comes from difference between host processor and emulator processor, we used x86 version of Android 6.0. We run emulator in Android Studio[4] on Windows7 and recorded API logs through "logcat" window. "logcat" is originally used for checking Android application status, and is displayed debugging information as shown in Fig. 3. By running applications on the modified Android system image which rebuilt as mentioned at Sect. 3.5, we can get API call logs in "logcat" window as shown in Fig. 4.

We recorded API call logs for each application from the moment of launch of applications to complete trial for confirm series of their functionality. Note that we have not tried whole functions of the application because it requires too much time. We terminated log recording after we tried representative function of the application. Here is examples when we terminate log recording. For a calculator application, we terminate log recording after performing some arithmetic. For a note-pad application, we terminate log recording after inputing some letters. For a searching store application, we terminate log recording after the application displayed store list according to input. For a camera application, we terminate log recording after taking a picture.

We modified Android 6.0 in order to record API call logs and recorded API call logs of 42 applications. In this paper, we modified all of APIs in a straightforward way. We hope that Android OS implement the function of output API call logs for debugging in order to follow update of OS if our detection method turned out to be effective.

4.2 Applying microG

We utilized Android Open Source Project (AOSP)[5] code to this experiment. However, most of applications which access location information crashed when we tried to install and run applications. One reason of that is the lack of Google Services and Playstore. Even though we added Google Services and Playstore into Android system image file, applications are still crashed. We guess that it was caused by Safety Net system which is included in Android system. Safety Net is the system to assess the integrity of the profile of combination between hardware and software. It is also works for the system to decide whether the device is tampered or not (e.g. detect Rooting) through signature key to application

[4] https://developer.android.com/studio/index.html.
[5] https://android.googlesource.com/.

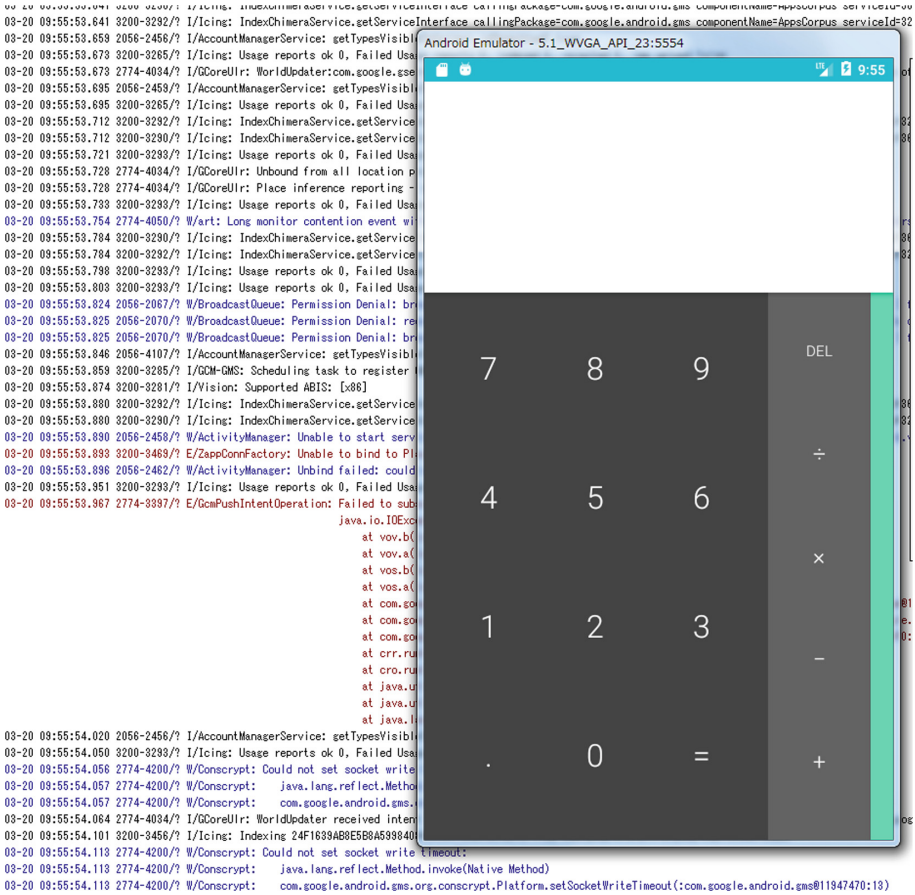


Fig. 3. Original Android on the Emulator and Logs

by author. In addition to tamper detection, Safety Net manages verification of the certification that the application verifies whether formal Google services and Playstore are installed or not. In order to avoid detection on Safety Net, we applied signature spoofing to Android source code and installed microG[6] frameworks instead of Google Services and Playstore. Signature spoofing enable us to spoof applications that verifies existence of formal Google Services and Playstore with signed signature. In order to enable microG, we modified four parts of source code related to package manager of Android and signature spoofing.

4.3 Collecting Applications for Evaluation

We collected 42 applications for evaluation which include sufficient samples for each of the category defined at Sect. 3.3. Those applications are collected through

[6] https://microg.org/.

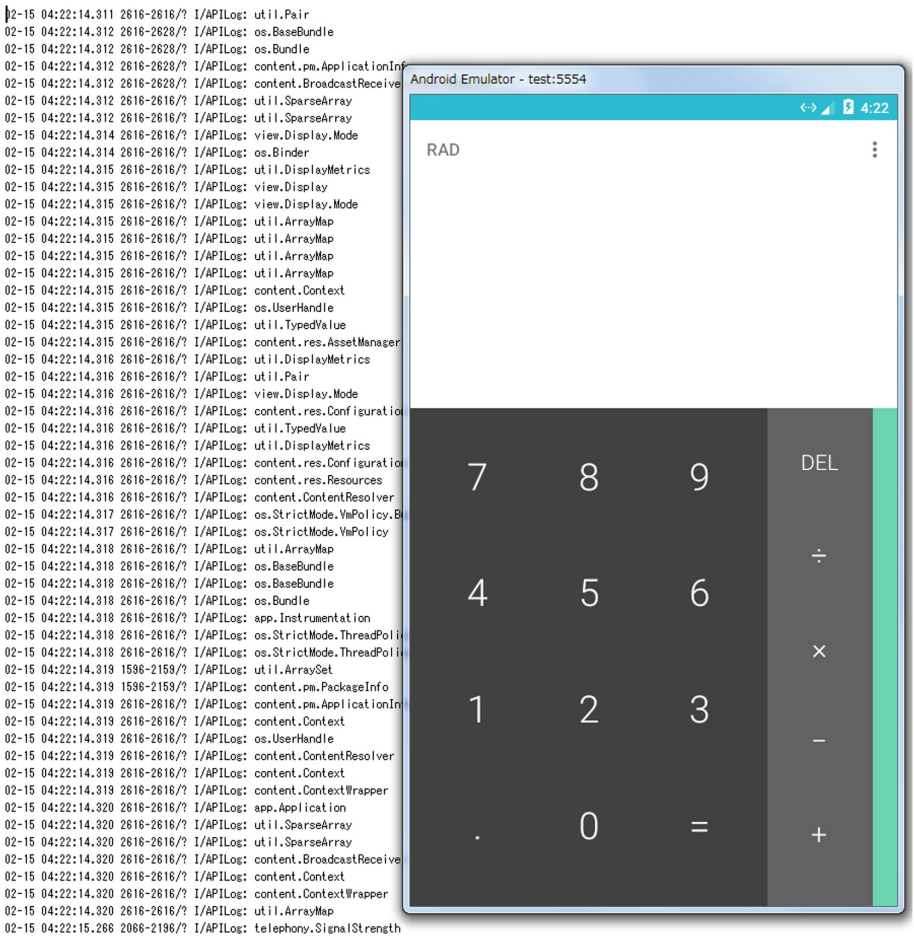


Fig. 4. Rebuilt Android on the Emulator and API call logs

APKPure[7] and 1mobile market. Detailed collection procedure for each category is as follows.

As No-Access application samples, we collected 5 applications which do not need permission to access to privacy information such as calculator and note-pad. To prove no permission requirement, we only confirmed that those applications' manifest do not include permission requirement after collection.

As Proper-Access application samples, we collected 30 applications which use only permission they obtained for its functional reason such as neighborhood searcher which uses location permission and camera which use camera and storage permission. To prove moderate permission requirement, we carefully confirmed relationship between functionality and required permissions.

[7] https://apkpure.com/jp/.

In order to find out Improper-Access application, we searched with keywords which we frequently hear fake or useless applications are existing category. Firstly, we choose applications which functions seem to convenient but it is impossible by physical reasons. For example, there are applications which appeal realizing solar charger in any terminal. Both of them are restricted by physical matter because solar charger requires solar panel hardware and flashlight does not brights with exceeding hardware limitation. We collected 10 suspicious applications which also includes permission request for privacy information in the manifest. Then, we confirmed whether those application really touch privacy. We found that 3 out of them do not touch privacy information so that we omit them from candidates. Finally, we used rest 7 applications as Improper-Access application samples.

To reduce false-positive rate, we did not used applications we cannot judge the appropriateness of permission obviously.

4.4 Evaluation Criteria

Our goal is to distinguish Improper-Access applications from Proper-Access ones by using API call logs. We used statistic based algorithm to extract features based on the classification mentioned described at Sect. 3.4.

We analyzed API call logs mentioned at Sect. 4.1 and calculated the frequency of ACCESS, SEND and RESPONSE for each application we collected. Firstly, we tried to find tendencies from the difference of averaged frequency of ACCESS, SEND, and RESPONSE rate between Proper-Access applications and Improper-Access ones. Secondly, we tried to find tendencies from the difference of averaged number of both ACCESS divided by RESPONSE and ACCESS divided by SEND. Thirdly, we tried to find tendencies from the length of API call logs between Proper-Access applications and Improper-Access ones.

4.5 Prediction Based on APIs Usage

The application categorized to No-Access should gives 0 for ACCESS count because they do not need permissions to access to privacy information as mentioned at Sect. 3.4. If not, we have to consider that there is possibility that the application has the way to avoid permission control.

We defined Proper-Access and Improper-Access based on whether they need unnecessary permissions to access to privacy information or abuse of privacy information or not. We estimated that the ACCESS frequency and the SEND frequency of Improper-Access applications would become higher value than those of Proper-Access ones because of collection of various privacy information unrelated with their functionality and send them to the other place. In addition, we predict that the value of both ACCESS divided by RESPONSE and ACCESS divided by SEND of Improper-Access applications will become higher value than those of Proper-Access ones because those applications gives comparatively small number of information reward for users (the less amount of RESPONSE) and much more amount of accessing and sending of privacy information.

5 Experimental Results and Discussion

5.1 Experimental Result

Appearance Ratio and Frequency of ACCESS. We calculated frequency of categorized API call named ACCESS, SEND, RESPONSE and their average for each No-Access, Proper-Access, and Improper-Access applications. Tables 2 and 3 shows calculated value. ACCESS, SEND, and RESPONSE show appearance ratio for all APIs and the reminders show calculated frequency. Table 3 shows number of API calls on No-Access, Proper-Access, and Improper-Access applications under this experiment with minimum, maximum, and average number.

Table 2. Result of frequency analysis (%)

	ACCESS	SEND	RESPONSE	ACCESS/RESPONSE	ACCESS/SEND
No-Access	0.00	0.00	1.07	0.00	0.00
Proper-Access	0.28	0.08	3.35	0.08	3.50
Improper-Access	0.09	0.22	1.59	0.06	0.41

Table 3. Length of API call logs (lines)

	Min	Max	Average
No-Access	2,052	8,424	3,640
Proper-Access	931	18,643	9,475
Improper-Access	6,338	13,005	7,868

Omitting Android.graphics Class. While we run applications and gathering API call logs, we found that most of applications use graphics class APIs so frequently especially in notification part. For example, while applications are loading data from the Internet and waiting for finishing communication, the application shows notification to the display using graphic class APIs. Due to these types of usage, graphics class API call occupies large part of API calls (29.5% on average) so that we considered that there is worth for data without graphic class API call. Tables 4 and 5 show result with omitting APIs which belong to android.graphics class on calculating frequency of the call.

Table 4. Result of frequency analysis without graphics class APIs (%)

	ACCESS	SEND	RESPONSE	ACCESS/RESPONSE	ACCESS/SEND
No-Access	0.00	0.00	1.31	0.00	0.00
Proper-Access	0.44	0.11	4.85	0.09	4.00
Improper-Access	0.11	0.28	1.98	0.06	0.39

Table 5. Length of API call logs without graphics class APIs (lines)

	Min	Max log	Average	Omitted ratio
No-Access	1,698	7,764	3,205	11.0%
Proper-Access	608	13,451	6,407	32.4%
Improper-Access	5054	10,613	6,360	19.2%

5.2 Factors to Affect Estimation

In the experimental results shown above, No-Access applications really do not use ACCESS APIs. As we defined No-Access as applications which do not access to privacy information, it is inevitable result excluding the application tries to obtain privacy information with exploiting some vulnerability.

The result about the frequency of ACCESS APIs usage is different from our estimation mentioned at Sect. 4.5. As shown in Tables 2 and 4, the frequency of ACCESS APIs used by Improper-Access applications is lower than that of Proper-Access applications. We guess that this result comes from that the Proper-Access applications access privacy information every time in required point, but the Improper-Access applications roughly read privacy information (e.g. gather all information at once) so that the Improper-Access application gives smaller count in API call count viewpoint. The result shows that the frequency of SEND APIs used by Improper-Access applications are more than twice than that of Proper-Access applications. The frequency of RESPONSE for Proper-Access applications are more than twice than that of Improper-Access application. We think that this characteristic comes from the Improper-Access applications will serve comparatively small amount of information to users because their first purpose is not for users convenience.

The difference about ACCESS between Proper-Access applications and Improper-Access applications is about 3 to 4 times. However, the number of ACCESS divided by SEND applications amplifies this difference so that the difference becomes about 9 to 10 times. Those results show that Improper-Access applications frequently send informations to the network so that dividing by SEND becomes good amplification method for detecting Improper-Access applications. The Improper-Access applications do not utilize graphic class API call compared to the Proper-Access application so that omitting graphics class APIs reduces difference between them. Maybe, the Improper-Access applications do not utilize notifications compared to the Proper-Access applications so that such a difference may occur.

6 Conclusion

We proposed a method to classify Android applications to 3 types based on accessing privacy information abuse and categorized them based on API call log by categorizing APIs into 4 types according to their role. We also proposed

statistic based application classification method based on API call logs and we showed the potential to detect applications which access to privacy information abuse. This research is currently under initial stage so that there are some differed points from our predictions in evaluation results. This should be analyzed with increasing number of application samples and consider the validity of our classification algorithm again.

As a future work, we have to continue to gather API call logs of much more applications to improve the detection method because current detection method is too preliminary one. We are planning to extract much more characteristic from ACCESS, SEND, and RESPONSE rate by calculating standard deviation and so on. Moreover, we have to try other method such as machine learning to detect privacy information leakage. After improving detection method, we will consider to evaluate with much more and various types of applications to proof effectiveness. In addition, we are planning to apply statistical analysis of API call logs to other than the detection of privacy information leakage such as detecting malicious activities for Android frameworks.

References

1. International Data Corporation. Smartphone OS marcket share. https://www.idc.com/promo/smartphone-market-share/os
2. Information technology Promotion Agency. A survey of awareness of the threat of information security in 2016. http://www.ipa.go.jp/files/000056568.pdf. (in Japanese)
3. Kim, J., Yoon, Y., Yi, K., Shin, J.: Scandal: static analyzer for detecting privacy leaks in android applications. In: Mobile Security Technologies, vol. 12 (2012)
4. Hosoya, R., Tsunoda, Y., Mori, T., Saito, T.: Measurement study of the privacy information collected by mobile apps. In: Computer Security Symposium, pp. 553–560, September 2017. (in Japanese)
5. Mann, C., Starostin, A.: A framework for static detection of privacy leaks in android applications. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, pp. 1457–1462 (2012)
6. Han, C., Matsumoto, S., Kawamoto, J., Sakurai, K.: Classified by the API call record and personal information leakage detection of android malware. Inf. Process. Soc. Jpn. **3A–4**, 1–8 (2016). (in Japanese)
7. Hatada, M., Mori, T.: Detecting android PUAs and classifying its variants with analysis of DNS queries. In: Computer Security Symposium, pp. 1068–1075, September 2017. (in Japanese)
8. Do, Q., Martini, B., Choo, K.-K.R.: Enhancing user privacy on android mobile devices via permissions removal. In: 2014 47th Hawaii International Conference on System Sciences (HICSS), pp. 5070–5079 (2014)
9. Gibler, C., Crussell, J., Erickson, J., Chen, H.: Androidleaks: automatically detecting potential privacy leaks in android applications on a large scale. In: International Conference on Trust and Trustworthy Computing, pp. 291–307 (2012)
10. Zhang, M., Yin, H.: Efficient, context-aware privacy leakage confinement for android applications without firmware modding. In: Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, pp. 259–270 (2014)

11. Davi, L., Dmitrienko, A., Sadeghi, A.-R., Winandy, M.: Privilege escalation attacks on android. In: International Conference on Information Security, pp. 346–360 (2010)
12. Chen, Y., Khandaker, M., Wang, Z.: Pinpointing vulnerabilities. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 334–345 (2017)
13. Chen, Y., Zhang, Y., Wang, Z., Xia, L., Bao, C., Wei, T.: Adaptive android kernel live patching. In: Proceedings of the 26th USENIX Security Symposium (2017)

Verification of LINE Encryption Version 1.0 Using ProVerif

Cheng Shi(✉) and Kazuki Yoneyama(✉)

Ibaraki University, Hitachi-shi Ibaraki, Japan
{18nm718l,kazuki.yoneyama.sec}@vc.ibaraki.ac.jp

Abstract. LINE is currently the most popular messaging service in Japan. Communications using LINE are protected by the original encryption scheme, called LINE Encryption, and specifications of the client-to-server transport encryption protocol and the client-to-client message end-to-end encryption protocol are published by the Technical Whitepaper. Though a spoofing attack (i.e., a malicious client makes another client misunderstand the identity of the peer) and a reply attack (i.e., a message in a session is sent again in another session by a man-in-the-middle adversary, and the receiver accepts these messages) to the end-to-end protocol have been shown, no formal security analysis of these protocols is known.

In this paper, we show a formal verification result of secrecy of application data and authenticity for protocols of LINE Encryption (Version 1.0) by using the automated security verification tool ProVerif. Especially, since it is claimed that the transport protocol satisfies forward secrecy (i.e., even if the static private key is leaked, security of application data is guaranteed), we verify forward secrecy for client's data and for server's data of the transport protocol, and we find an attack to break secrecy of client's application data. Moreover, we find the spoofing attack and the reply attack, which are reported in previous papers.

1 Introduction

1.1 Background

With the development of network communications technology, more and more people use messaging services to communicate. LINE is currently the most popular messaging service in Japan. Thus, if there is security vulnerability in LINE, widespread incidents may be caused due to its popularity. Hence, it is required that the security of LINE is rigorously analyzed.

In order to ensure the security of LINE, communications in LINE are protected by a dedicated encrypted communication scheme, called LINE Encryption, and specifications of the client-to-server transport encryption protocol (TEP) and the client-to-client message end-to-end encryption protocol (E2EEP) are published by the Technical Whitepaper [1]. In [1], some informal security analyses of the TEP and the E2EEP are shown, and it is claimed that the

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 158–173, 2018.
https://doi.org/10.1007/978-3-319-97916-8_11

TEP satisfies forward secrecy such that "In the event that a private key is leaked, messages that were encrypted before the leak are protected if the communication supports forward secrecy" [2]. Since there are two kinds of messages (client's application data encrypted by temporary key $\mathbf{key}_{\mathrm{temp}}$ and server's application data encrypted by forward secure key $\mathbf{key}_{\mathrm{FS}}$) in TEP, we need to consider two kinds of forward secrecy (i.e., forward secrecy for client's data and forward secrecy for server's data).

On the other hand, Espinoza et al. [3] showed a reply attack against the E2EEP. A man-in-the-middle (MTM) adversary can send an encrypted message in an old session as the message in the new session without changing the content of the message. Though the adversary cannot know the content of the message, the receiver client accepts these two messages as valid. Isobe and Minematsu [4] showed a spoofing attack against the E2EEP. A malicious client C_3 intercepts the E2EEP between clients C_1 and C_2, and impersonates C_1 to C_2. Hence, since another unknown attack may exist, security of LINE Encryption is still unclear.

On the other hand, since it is difficult to analyze all attacks by hands, such as the replay attack or the spoofing attack, automated security verification methods by using formal methods have been studied to formally verify the security of cryptographic protocols.

1.2 Contribution

In this paper, we give the first formal verification result of the security of LINE Encryption (Version 1.0) by using the automated security verification tool ProVerif [5]. Specifically, for TEP, we verify forward secrecy of both client's application data and server's application data, and server authenticity. For E2EEP, we verify secrecy of application data and authenticity. For verifications of forward secrecy, we use reachability to application data for the adversary. For verifications of authenticity, we use correspondence assertions of some events. We obtain the following verification results:

For TEP: We find an attack to break forward secrecy for client's data, but forward secrecy for server's data and server authenticity are not broken.

For E2EEP: We find the spoofing attack and the replay attack, but secrecy of application data is not broken.

Thus, our verification result captures all known attacks to LINE Encryption, and points out the attack to forward secrecy for client's data, which is not formally reported. Therefore, our result clarifies that the automated verification tool is useful to verify the security of messaging protocols.

2 Preliminaries

2.1 Client-to-Server Transport Encryption Protocol (TEP)

The client and server exchange the following messages in order to establish the transport key used to protect application data.

Static Keys. In order to guarantee that clients only connect to legitimate the LINE servers, TEP uses static ECC and ECDSA key pairs. The LINE servers securely stores the private part of each pair, while the corresponding public keys are embedded in LINE client applications.

- ECDH key pair for key exchange: ($\mathbf{static}_{\text{private}}$,$\mathbf{static}_{\text{public}}$)
- ECDSA key pair for server identity verification: ($\mathbf{sign}_{\text{private}}$,$\mathbf{sign}_{\text{public}}$)

Client Hello

1. Generate an initial ephemeral ECDH key ($\mathbf{c\_init}_{\text{private}}$,$\mathbf{c\_init}_{\text{public}}$) and a client nonce $\mathbf{c}_{\text{nonce}}$.
2. Derive a temporary transport key and initialization vector (IV) using the server's static key and the initial ephemeral key generated in Step 1 as follows.

   $\mathbf{len}_{\text{key}}$=16
   $\mathbf{len}_{\text{iv}}$=16
   $\mathbf{share}_{\text{temp}}$=ECDH($\mathbf{c\_init}_{\text{private}}$,$\mathbf{static}_{\text{public}}$)
   $\mathbf{MS}_{\text{temp}}$=HKDF$_{\text{ex}}$($\mathbf{c_init}_{\text{public}}||\mathbf{c}_{\text{nonce}}$,$\mathbf{share}_{\text{temp}}$)[1]
   $\mathbf{keyiv}_{\text{temp}}$=HKDF$_{\text{exp}}$($\mathbf{MS}_{\text{temp}}$,"legy temp key",$\mathbf{len}_{\text{key}}+\mathbf{len}_{\text{iv}}$)
   $\mathbf{key}_{\text{temp}}$=$\mathbf{keyiv}_{\text{temp}}$[0:15]
   $\mathbf{iv}_{\text{temp}}$=$\mathbf{keyiv}_{\text{temp}}$[16:31]

3. Generate an ephemeral ECDH client handshake key ($\mathbf{c}_{\text{private}}$,$\mathbf{c}_{\text{public}}$).
4. $\mathbf{c}_{\text{public}}$ and application data $\mathbf{appdata}_{\text{client}}$ are encrypted with $\mathbf{key}_{\text{temp}}$ and the client nonce $\mathbf{c}_{\text{nonce}}$ using the AES-GCM [6] AEAD cipher. The nonce is calculated by combining a client/server marker **marker**, a sequence number $\mathbf{num}_{\text{seq}}$, and $\mathbf{iv}_{\text{temp}}$ obtained in the handshake process.
5. Send the version $\mathbf{static}_{\text{keyversion}}$, client's initial ephemeral key $\mathbf{c\_init}_{\text{public}}$,[2] client nonce $\mathbf{c}_{\text{nonce}}$ and encrypted data $\mathbf{data}_{\text{enc}}$ to the server.

Server Hello

1. Calculate the temporary transport key $\mathbf{key}_{\text{temp}}$ and IV $\mathbf{iv}_{\text{temp}}$ using the server's static ECDH key $\mathbf{static}_{\text{private}}$ and the client's initial ephemeral key $\mathbf{c\_init}_{\text{public}}$ as follows.

   $\mathbf{share}_{\text{temp}}$=ECDH($\mathbf{static}_{\text{private}}$,$\mathbf{c\_init}_{\text{public}}$)
   $\mathbf{MS}_{\text{temp}}$=HKDF$_{\text{ex}}$($\mathbf{c_init}_{\text{public}}||\mathbf{c}_{\text{nonce}}$,$\mathbf{share}_{\text{temp}}$)
   $\mathbf{keyiv}_{\text{temp}}$=HKDF$_{\text{exp}}$($\mathbf{MS}_{\text{temp}}$,"legy temp key",$\mathbf{len}_{\text{key}}+\mathbf{len}_{\text{iv}}$)
   $\mathbf{key}_{\text{temp}}$=$\mathbf{keyiv}_{\text{temp}}$[0:15]
   $\mathbf{iv}_{\text{temp}}$=$\mathbf{keyiv}_{\text{temp}}$[16:31]

[1] In [1], it is described as $\mathbf{MS}_{\text{temp}}$ = HKDF$_{\text{ex}}$($\mathbf{c}_{\text{public}}||\mathbf{c}_{\text{nonce}}$, $\mathbf{share}_{\text{temp}}$). However, it is a typo. The authors confirmed the typo to the LINE Security Team.

[2] In [1], it is described as to send $\mathbf{c}_{\text{public}}$. However, it is a typo. The authors confirmed the typo to the LINE Security Team.

2. Decrypt application data $\mathbf{appdata}_{client}$ with $\mathbf{key}_{\text{temp}}$ and $\mathbf{iv}_{\text{temp}}$, and extract $\mathbf{c}_{\text{public}}$.
3. Generate an ephemeral key pair ($\mathbf{s}_{\text{private}}$,$\mathbf{s}_{\text{public}}$) and a server nonce $\mathbf{s}_{\text{nonce}}$.
4. Derive the forward-secure (FS) transport key $\mathbf{key}_{\text{FS}}$ and IV $\mathbf{iv}_{\text{FS}}$ as follows.

$$\mathbf{len}_{\text{key}}=16$$
$$\mathbf{len}_{\text{iv}}=16$$
$$\mathbf{share}_{\text{FS}}=\text{ECDH}(\mathbf{s}_{\text{private}},\mathbf{c}_{\text{public}})$$
$$\mathbf{MS}_{\text{FS}}=\text{HKDF}_{\text{ex}}(\mathbf{c}_{\text{nonce}}||\mathbf{s}_{\text{nonce}},\mathbf{share}_{\text{FS}})$$
$$\mathbf{keyiv}_{\text{FS}}=\text{HKDF}_{\text{exp}}(\mathbf{MS}_{\text{FS}},\text{“legy temp key”},\mathbf{len}_{\text{key}}+\mathbf{len}_{\text{iv}})$$
$$\mathbf{key}_{\text{FS}}=\mathbf{keyiv}_{\text{FS}}[0{:}15]$$
$$\mathbf{iv}_{\text{FS}}=\mathbf{keyiv}_{\text{FS}}[16{:}31]$$

5. Generate and sign the handshake state using the server's static signing key as follows.
$$\mathbf{state}=\text{SHA256}(\mathbf{c}_{\text{public}}||\mathbf{c}_{\text{nonce}}||\mathbf{s}_{\text{public}}||\mathbf{s}_{\text{nonce}})$$
$$\mathbf{state}_{\text{sign}}=\text{ECDSA}_{\text{sign}}(\mathbf{state},\mathbf{sign}_{\text{private}})$$
6. Application data $\mathbf{appdata}_{server}$ is encrypted with $\mathbf{key}_{\text{FS}}$ and the nonce $\mathbf{s}_{\text{nonce}}$ using the AES-GCM AEAD cipher. The nonce is calculated by combining a client/server marker **marker**, a sequence number $\mathbf{num}_{\text{seq}}$, and the $\mathbf{iv}_{\text{FS}}$ obtained in the handshake process.
7. Send the ephemeral key $\mathbf{s}_{\text{public}}$, server nonce $\mathbf{s}_{\text{nonce}}$ and encrypted data $\mathbf{data}'_{\text{enc}}$ to the client.

Client Finish

1. Verify the handshake signature. If the signature is valid, proceed to the next step. If not, abort the connection.
2. Derive $\mathbf{key}_{\text{FS}}$ and $\mathbf{iv}_{\text{FS}}$ as follows.

$$\mathbf{share}_{\text{FS}}=\text{ECDH}(\mathbf{c}_{\text{private}},\mathbf{s}_{\text{public}})$$
$$\mathbf{MS}_{\text{FS}}=\text{HKDF}_{\text{ex}}(\mathbf{c}_{\text{nonce}}||\mathbf{s}_{\text{nonce}},\mathbf{share}_{\text{FS}})$$
$$\mathbf{keyiv}_{\text{FS}}=\text{HKDF}_{\text{exp}}(\mathbf{MS}_{\text{FS}},\text{“legy temp key”},\mathbf{len}_{\text{key}}+\mathbf{len}_{\text{iv}})$$
$$\mathbf{key}_{\text{FS}}=\mathbf{keyiv}_{\text{FS}}[0{:}15]$$
$$\mathbf{iv}_{\text{FS}}=\mathbf{keyiv}_{\text{FS}}[16{:}31]$$

3. Encrypt all subsequent application data using $\mathbf{key}_{\text{FS}}$ and $\mathbf{iv}_{\text{FS}}$.

2.2 Message End-to-End Encryption (E2EEP)

Client-to-Client Key Exchange. In order to be able to exchange encrypted messages, clients must share a common cryptographic secret. When a LINE client wishes to send a message, it first retrieves the current public key of the recipient. Next, the client passes its own private key and the recipient's public key to the ECDH algorithm in order to generate a shared secret as follows.

$$\begin{aligned}\mathbf{SharedSecret}&=\text{ECDH}_{\text{curve25519}}(\mathbf{key}^{\mathbf{user1}}{}_{\text{private}},\mathbf{key}^{\mathbf{user2}}{}_{\text{public}})\\&=\text{ECDH}_{\text{curve25519}}(\mathbf{key}^{\mathbf{user2}}{}_{\text{private}},\mathbf{key}^{\mathbf{user1}}{}_{\text{public}})\end{aligned}$$

Message Encryption. The sender client encrypts a message with a unique encryption key and IV, and sends the encrypted message to the recipient client.

1. The encryption key $\mathbf{Key}_{\text{encrypt}}$ and IV $\mathbf{IV}_{\text{encrypt}}$ are derived from the shared secret calculated in the above process, and a randomly generate **salt** as follows.

   $\mathbf{Key}_{\text{encrypt}}$=SHA256(**Shared Secret**||**salt**||"Key")
   $\mathbf{IV}_{\text{pre}}$=SHA256(**Shared Secret**||**salt**||"IV")
   $\mathbf{IV}_{\text{encrypt}}$=$\mathbf{IV}_{\text{pre}}$[0:15]⊕$\mathbf{IV}_{\text{pre}}$[16:31]

2. The generated key and IV are used to encrypt the message payload **M** using AES in CBC block mode.
3. The sender calculates a message authentication code (MAC) of the ciphertext **C** as follows.

   $\mathbf{MAC}_{\text{plain}}$=SHA256(**C**)
   $\mathbf{MAC}_{\text{enc}}$=AESECB($\mathbf{Key}_{\text{encrypt}}$,$\mathbf{MAC}_{\text{plain}}$[0:15]⊕$\mathbf{MAC}_{\text{plain}}$[16:31])

4. **version**,**content type**,**salt**,**C**,**MAC_enc**,**sender key ID** and **recipient key ID** are included in the message sent to the recipient.
5. The recipient derives the symmetric encryption key $\mathbf{Key}_{\text{encrypt}}$, and IV $\mathbf{IV}_{\text{encrypt}}$ as described above.
6. The recipient calculates the MAC $\mathbf{MAC}'_{\text{enc}}$ of the received cipher text, and compares it with the MAC $\mathbf{MAC}_{\text{enc}}$ value included in the message. If they match, the contents of the message are decrypted and displayed. Otherwise, the message will be discarded.

2.3 ProVerif

ProVerif is a model checking tool that performs automated security verification. To verify a security requirement of a protocol by ProVerif, we must formalize the cyptographic primitives, the protocol specification and the security requirement as input to ProVerif. Here, we briefly explain how to formalize these by using an example of a symmetric key encryption. For the detail of ProVerif, please see [5].

1. Define communication paths and cryptographic primitives
   - Type designates types representing keys, random numbers, etc.

     ```
     type key (*types of private keys in symmetric key encryption
         *)
     type coins (*type of random numbers*)
     ```

   - Free name defines channel name.

     ```
     free c:channel (*Public communication channel*)
     free c:channel [private] (*Secret communication channel*)
     ```

   - Constructors defines cryptographic primitives such as encryption functions, etc.

```
fun senc(bitstring,key):bitstring (*function of symmetric
    key encryption*)
```

– Destructor specifies conditions of functions.

```
reduc forall m:bitstring,k:key;sdec(senc(m,k),k)=m. (*
    decryption condition of symmetric key encryption*)
```

2. Define participant behaviors within the cryptographic protocol

```
out(c,A) (*send the message A to channel c*)
in(c,B) (*receive the message B from channel c*)
```

3. Define public information, confidential information held in advance by each participant, and give it as input to participants.

```
new r:coins (*generate a random number*)
((!clientA)|(!serverB)) (*parallel execution of client and server*)
```

A variety of properties can be analyzed by ProVerif, such as the correspondence assertions (i.e., whether event B occurred before the event A occurred), the reachability (i.e., whether a specific event occurred), and the observation equivalence (i.e., whether is able to analyze two processes that perform different computations but have the same run result).

3 Formalization of LINE encryption

In this section, we give our formalization of LINE Encryption in ProVerif.

3.1 Formalization of TEP

Rules for Signature. Define types of signature key and verification key required for digital signature, furthermore, the function `spk` that creates a verification key from the signing key. When a plaintext (bitstring) and a signing key are inputted, the function `sign` generates a signature. When a signature and a verification key are inputted, if the verification result is correct, define the `checksign` outputs true.

```
type signpublickey. (*verification key *)
type signprivatekey. (*signing key *)
fun spk(signprivatekey):signpublickey. (*generate a verification
    key *)
fun sign(bitstring,signprivatekey):bitstring. (*generate a
    signature *)
reduc forall m:bitstring,sprikey:signprivatekey;
checksign(sign(m,sprikey),spk(sprikey),m)=true. (*verify signature
    *)
```

Rules for ECDH Key Exchange. Define types of the generator, exponents and the base point, and functions `Ggen` to convert the base point to the generator and `sca` to compute the scalar multiplication, and commutativity by `equation`.

```
type G. (*type of generator *)
type scalar. (*type of scalar *)
type basis. (*type of base points *)
fun Ggen(basis):G. (*convert base point to generator *)
fun sca(scalar,G):G. (*scalar multiplication *)
equation forall a:scalar, b:scalar, P:basis;
sca(a,sca(b,Ggen(P)))=sca(b,sca(a,Ggen(P))). (*commutativity*)
```

Ruless for XOR. Define the function `xor` to compute the XOR of two inputs, and the property of XOR by six `equation`.

```
fun xor(bitstring,bitstring):bitstring.
equation forall x:bitstring, y:bitstring; xor(xor(x,y),y)=x.
equation forall x:bitstring; xor(x,xor(x,x))=x.
equation forall x:bitstring; xor(xor(x,x),x)=x.
equation forall x:bitstring, y:bitstring; xor(y,xor(x,x))=y.
equation forall x:bitstring, y:bitstring; xor(xor(x,y),xor(x,x))=
    xor(x,y).
equation forall x:bitstring, y:bitstring; xor(xor(x,y),xor(y,y))=
    xor(x,y).
```

Parameter. Define key type, random number type, version type, and fixed values and word "legy temp key" as `const`.

```
type key.
type iv.
type coins.
type version.
const legy:bitstring[data].
const len:bitstring[data].
const num:bitstring[data].
const marker:bitstring[data].
```

Rules for AEAD. Define the function `senc` to encrypt a plaintext and the function `sdec` to decrypt a ciphertext, and the relationship between `senc` and `sdec` by `reduc`.

```
fun senc(key,coins,bitstring):bitstring. (*encryption function *)
reduc forall x:bitstring, k:key, r:coins;
sdec(k,r,senc(k,r,x))=x.(*decryption function *)
```

Declaration of Channel and Secret. Define the channel and secret information by `free`.

```
free c:channel. (*the channel *)
free appdata1:bitstring[private]. (*the client's secret information
     *)
free appdata2:bitstring[private]. (*the server's secret information
     *)
```

Type-Converting Functions. Define implicit functions to convert types for type adjustments of inputs of functions.

```
fun HKDFex(bitstring,G):bitstring. (*HKDFex*)
fun HKDFexp(bitstring,bitstring,bitstring):bitstring. (*HKDFexp*)
fun Ggen3(bitstring):key. (*convert bitstring type to key type *)
fun Ggen4(bitstring):iv. (*convert bitstring type to iv type *)
fun Hash(bitstring):bitstring. (*hash function *)
fun Ggen6(bitstring):coins. (*convert bitstring type to randam type
     *)
fun Ggen7(iv):bitstring. (*convert iv type to bitstring type *)
```

Verification of Forward Secrecy. Use reachability to verify forward secrecy. The attacker is passive, but he/she can obtain all static private keys of the server. Forward secrecy for client's (resp. server's) data require that the attacker cannot reach client's (resp. server's) application data. If the protocol has forward secrecy for client's (resp. server's) data, `attacker(appdata1)` (resp. `attacker(appdata2)`) will not happen. In other words, `appdata1` (resp. `appdata2`) cannot be obtained by the passive attacker.

```
set attacker = passive.
query attacker(appdata1).
query attacker(appdata2).
```

Verification of Server Authenticity. Using the correspondence assertions, we can verify authenticity for clients. We define event `Client1` corresponding to encrypting Client's secret message, and event `Server1` corresponding to decrypting the secret message. If `Server1` occurs, then `Client1` must occur before `Server1`. It corresponds to client authenticity. Also, we define event `Server2` corresponding to accepting digital signature verification, and event `Client2` corresponding to completion of the session. If `Client2` occurs, then `Server2` must occur before `Client2`. It corresponds to server authenticity.

```
event Client1(key,coins).
event Server1(key,coins).
event Client2(key,iv).
event Server2(key,iv).
query x:key, n:coins;
```

```
event(Server1(x,n))==>event(Client1(x,n)).
query x:key, n:iv;
event(Client2(x,n))==>event(Server2(x,n)).
```

Client Subprocess. Define client's actions.

```
let Client(ver:version,J:basis,stapu:G,spuk:signpublickey)=
new cinitpr:scalar;
new cnon:coins; (*Generate random number of client*)
let cinitpu=sca(cinitpr,Ggen(J))in
let sharedtemp=sca(cinitpr,stapu)in
let MStemp=HKDFex((cinitpu,cnon),sharedtemp)in (*MStemp*)
let keyivtemp=HKDFexp(MStemp,legy,len)in
let keytemp=Ggen3(breakf(keyivtemp))in (*keytemp*)
let ivtemp=Ggen4(breakb(keyivtemp))in (*ivtemp*)
let nonce=Ggen6(xor((marker,num),Ggen7(ivtemp)))in
new cpr:scalar;
let cpu=sca(cpr,Ggen(J))in
let dataenc=senc(keytemp,nonce,(cpu,appdata1))in (*dataenc*)
event Client1(keytemp,nonce);
out(c,(ver,cinitpu,cnon,dataenc));
in(c,(spu':G,srand:coins,statesign':bitstring,dataenc1':bitstring))
    ;
let statesign1=Hash((cpu,cnon,spu',srand))in
if checksign(statesign',spuk,statesign1)then (*Detect signature *)
let sharedFS'=sca(cpr,spu')in
let MSFS'=HKDFex((cnon,srand),sharedFS')in
let keyivFS'=HKDFexp(MSFS',legy,len)in
let keyFS'=Ggen3(breakf(keyivFS'))in
let ivFS'=Ggen4(breakb(keyivFS'))in
event Client2(keyFS',ivFS').
```

Server Subprocess. Define server's actions.

```
let Server(stapr:scalar,stapu:G,J:basis,sprk:signprivatekey)=
in(c,(sver:version,cinitpu':G,crand:coins,dataenc':bitstring));
let sharedtemp'=sca(stapr,cinitpu')in (*shardtemp*)
let MStemp'=HKDFex((cinitpu',crand),sharedtemp')in (*MStemp*)
let keyivtemp'=HKDFexp(MStemp',legy,len)in (*keyivtemp*)
let keytemp'=Ggen3(breakf(keyivtemp'))in (*keytemp*)
let ivtemp'=Ggen4(breakb(keyivtemp'))in (*ivtemp*)
let nonce1=Ggen6(xor((marker,num),Ggen7(ivtemp')))in
let (cpu':G,appdata1':bitstring)=sdec(keytemp',nonce1,dataenc')in
    (*cpu&app data*)
new snon:coins; (*Generate random number of server*)
new spr:scalar;
let sharedFS=sca(spr,cpu')in
let MSFS=HKDFex((crand,snon),sharedFS)in (*MSFS*)
```

```
let keyivFS=HKDFexp(MSFS,legy,len)in
let keyFS=Ggen3(breakf(keyivFS))in (*keyFS*)
let ivFS=Ggen4(breakb(keyivFS))in (*ivFS*)
let spu=sca(spr,Ggen(J))in
let state=Hash((cpu',crand,spu,snon))in (*state*)
let statesign=sign(state,sprk)in (*signature *)
let nonce2=Ggen6(xor((marker,num),Ggen7(ivFS)))in
let dataenc1=senc(keyFS,nonce2,appdata2)in (*dataenc*)
event Server1(keytemp',nonce1);
event Server2(keyFS,ivFS);
out(c,(spu,snon,statesign,dataenc1)).
```

Main Process for Verifying Forward Secrecy. Generate the version `ver`, base point `J`, the ECDH private key `stapr`, and the signing key `sprk`. The ECDH public key `stapu` and public key of signature `spuk` are exposed in channel `c`. The client subprocess and the server subprocess are executed in parallel in phase 0. In order to verify forward secrecy, the ECDH private key `stapr` and the signing key `sprk` will be exposed in phase 1.

```
process
new ver:version;
new J:basis;
new stapr:scalar;
new sprk:signprivatekey;
let stapu=sca(stapr,Ggen(J))in out(c,stapu);
let spuk=spk(sprk)in out(c,spuk);
(((!Client(ver,J,stapu,spuk))|(!Server(stapr,stapu,J,sprk)))|phase
    1;out(c,(stapr,sprk)))
```

3.2 Formalization of E2EEP

Rules for ECDH Key Exchange. The process is described in Sect. 3.1, and we omit it.

Rules for AES-CBC. Define the function `senccbc` to encrypt a plaintext and the function `sdeccbc` to decrypt a ciphertext, and the relationship between `senccbc` and `sdeccbc` by `reduc`.

```
type key.
type iv. (*type of keys *)
fun senccbc(bitstring,key,iv):bitstring. (*encryption function *)
reduc forall m:bitstring, k:key, i:iv ; sdeccbc(senccbc(m,k,i),k,i)
    =m. (*decryption function *)
```

Rules for AES-ECB. Define the function `sencecb` to encrypt a plaintext and the function `sdececb` to decrypt a ciphertext, and the relationship between `sencecb` and `sdececb` by `reduc`.

```
fun sencecb(bitstring,key):bitstring. (*encryption function *)
reduc forall m:bitstring, k:key; sdececb(sencecb(m,k),k)=m. (*
    decryption function *)
```

Parameter. Define random number type, version type and fixed words "Key" and "IV" by `const`.

```
type coins.
type version.
type ID.
const Key:bitstring[data].
const IV:bitstring[data].
```

Rules for XOR. The process is described in Sect. 3.1, and we omit it.

Declaration of Channel and Secret. Define the channel and secret information.

```
free c:channel. (*channel *)
free M:bitstring[private]. (*client's secret information *)
```

Hash Function and Type-Converting Functions. Define the hash function and implicit functions to convert types for type adjustments of inputs of functions.

```
fun Hash(bitstring):bitstring. (*hash function *)
fun Ggen1(G,coins,bitstring):bitstring. (*G, coins, bitstring to
    bitstring type *)
fun Ggen2(bitstring):key. (*convert bitstring type to key type *)
fun Ggen3(bitstring):iv. (*convert bitstring type to iv type *)
```

Verification of Secrecy and Authenticity for Replay Attack. Using the correspondence assertions, we can verify authenticity for clients. We define event `Client1` which Client1 encrypts Client1's secret message, and event `Client2` which Client2 decrypts Client 1's secret message. If `Client2` occurs, then `Client1` must occur only once before `Client2`. In order to verify replay attack, we use injective correspondence assertions `inj-event` to capture the one-to-one relationship. If it is a non-one-to-one relationship, it may happen that `Client2` is executed twice or more, but `Client1` is executed only once. In other words, it corresponds to a replay attack.

```
query attacker(M).
event Client1(key,iv).
event Client2(key,iv).
query x:key, i:iv;
inj-event(Client2(x,i))==>inj-event(Client1(x,i)).
```

Client1 Subprocess for Replay Attack. Define client1's actions.

```
let Client1(ver:version,c1cpr:scalar,c1cpu:G,c2cpu:G,J:basis,c1id:
    ID,c2id:ID)=
in(c,(c2cpu':G));
if c2cpu=c2cpu' then
new salt:coins;
let SharedSecret=sca(c1cpr,c2cpu')in
let Ken=Ggen2(Hash((SharedSecret,salt,Key)))in
let IVpre=Hash((SharedSecret,salt,IV))in
let IVen=Ggen3(xor(breakf(IVpre),breakb(IVpre)))in
event Client1(Ken,IVen);
let C=senccbc(M,Ken,IVen)in
let MACp=Hash(C)in
let MACe=sencecb(xor(breakf(MACp),breakb(MACp)),Ken)in
out(c,(ver,salt,C,MACe,c1id,c2id)).
```

Client2 Subprocess for Replay Attack. Define client2's actions.

```
let Client2(ver:version,c2cpr:scalar,c2cpu:G,c1cpu:G,J:basis,c2id:
    ID)=
out(c,(c2cpu));
in(c,(ver':version,ran:coins,C':bitstring,mac:bitstring,id1:ID,id2:
    ID));
if c2id=id2 then
let SharedSecret'=sca(c2cpr,c1cpu)in
let Ken'=Ggen2(Hash((SharedSecret',ran,Key)))in
let MACp'=Hash(C')in
let MACe'=sencecb(xor(breakf(MACp'),breakb(MACp')),Ken')in
if mac=MACe' then
let IVpre'=Hash((SharedSecret',ran,IV))in
let IVen'=Ggen3(xor(breakf(IVpre'),breakb(IVpre')))in
let M'=sdeccbc(C',Ken',IVen')in
event Client2(Ken',IVen').
```

Main Process for Replay Attack. Generate the version `ver`, base point `J`, Client1's ECDH private key `c1cpr` and client2's ECDH private key `c2cpr`. The Client1's ECDH public key `c1cpu` and client2's ECDH public key c2cpu are exposed in channel `c`. Client1 subprocess and Client2 subprocess are executed in parallel.

```
process
new ver:version;
new J:basis;
new c1cpr:scalar;
new c2cpr:scalar;
new c1id:ID;
new c2id:ID;
let c1cpu=sca(c1cpr,Ggen(J))in out(c,c1cpu);
let c2cpu=sca(c2cpr,Ggen(J))in out(c,c2cpu);
((!!Client1(ver,c1cpr,c1cpu,c2cpu,J,c1id,c2id))|(!Client2(ver,c2cpr
    ,c2cpu,c1cpu,J,c2id)))
```

Verification of Secrecy and Authenticity for Spoofing Attack. Using the correspondence assertions, we can verify authenticity for clients. Event `accept`, which the client1 believes that it has accepted to run the protocol with the client2 and the correct sender key ID. Event `term`, which the client2 believes that it has terminated a protocol run with the client1 with the correct sender key ID. When a spoofing attack occurs, a fake sender key ID is used by an attacker and the correspondence assertions cannot be correct execution.

```
query attacker(M).
event accept(ID).
event term(ID).
query i:ID;
event(term(i))==>event(accept(i)).
```

Client1 Subprocess for Spoofing Attack. Define client1's actions.

```
let Client1(ver:version,c1cpr:scalar,c1cpu:G,c2cpu:G,J:basis,c1id:
    ID)=
in(c,(c2cpu':G,c2id':ID));
if c2cpu=c2cpu' then
event accept(c1id);
new salt:coins;
let SharedSecret=sca(c1cpr,c2cpu')in
let Ken=Ggen2(Hash((SharedSecret,salt,Key)))in
let IVpre=Hash((SharedSecret,salt,IV))in
let IVen=Ggen3(xor(breakf(IVpre),breakb(IVpre)))in
let C=senccbc(M,Ken,IVen)in
let MACp=Hash(C)in
let MACe=sencecb(xor(breakf(MACp),breakb(MACp)),Ken)in
out(c,(ver,salt,C,MACe,c1id,c2id')).
```

Client2 Subprocess for Spoofing Attack. Define client2's actions.

```
let Client2(ver:version,c2cpr:scalar,c2cpu:G,c1cpu:G,J:basis,c2id:
    ID)=
```

Table 1. Verification results for TEP

Secrecy of app. data		Forward secrecy of app. data		Server
Client's data	Server's data	Client's data	Server's data	Authenticity
✓	✓	×	✓	✓

✓ means that no attack is found. × means that an attack is found.

Table 2. Verification results for E2EEP

Secrecy of message	Authenticity	
	Reply attack	Spoofing attack
✓	×	×

✓ means that no attack is found. × means that an attack is found.

```
out(c,(c2cpu,c2id));
in(c,(ver':version,ran:coins,C':bitstring,mac:bitstring,id1:ID,id2:
    ID));
if c2id=id2 then
let SharedSecret'=sca(c2cpr,c1cpu)in
let Ken'=Ggen2(Hash((SharedSecret',ran,Key)))in
let MACp'=Hash(C')in
let MACe'=sencecb(xor(breakf(MACp'),breakb(MACp')),Ken')in
if mac=MACe' then
let IVpre'=Hash((SharedSecret',ran,IV))in
let IVen'=Ggen3(xor(breakf(IVpre'),breakb(IVpre')))in
let M'=sdeccbc(C',Ken',IVen')in
event term(id1).
```

Main Process for Spoofing Attack. Generate the version `ver`, base point `J`, Client1's ECDH private key `c1cpr`, client2's ECDH private key `c2cpr`, the sender key ID `c1id` and the recipient key ID `c2id`. The Client1's ECDH public key `c1cpu`, client2's ECDH public key `c2cpu` , `c1id` and `c2id` are exposed in channel `c`. Client1 subprocess and Client2 subprocess are executed in parallel.

```
process
new ver:version;
new J:basis;
new c1cpr:scalar;
new c2cpr:scalar;
new c1id:ID;
new c2id:ID;
out(c,c2id);
out(c,c1id);
let c1cpu=sca(c1cpr,Ggen(J))in out(c,c1cpu);
let c2cpu=sca(c2cpr,Ggen(J))in out(c,c2cpu);
((!Client1(ver,c1cpr,c1cpu,c2cpu,J,c1id))|(!Client2(ver,c2cpr,c2cpu
    ,c1cpu,J,c2id)))
```

4 Verification Results

In this section, we show the verification results. To summarize, we obtain the following verification results as in Table 1 for TEP and Table 2 for E2EEP. Next, we describe the found attack against forward secrecy for client's data of TEP, and the found reply attack and spoofing attack for E2EEP.

4.1 Attack to Break Forward Secrecy for Client's Data of TEP

1. An attacker who monitors channel **c** can knows the **version**, initial ephemeral ECDH public key $\mathbf{c\_init}_{\text{public}}$, clint nonce $\mathbf{c}_{\text{nonce}}$ and ciphertext $\mathbf{data}_{\text{enc}}$, which the client sends to the server.
2. The attacker obtains the static private keys $\mathbf{static}_{\text{private}}$ and $\mathbf{sign}_{\text{private}}$.
3. By using $\mathbf{c\_init}_{\text{public}}$ and $\mathbf{static}_{\text{private}}$, the attacker generates temporary transport key $\mathbf{key}_{\text{temp}}$ and initialization vector (IV) $\mathbf{iv}_{\text{temp}}$.
4. By using $\mathbf{iv}_{temp}$ the attacker generates **nonce1**.
5. By using **nonce1** and $\mathbf{key}_{\text{temp}}$, the attacker decrypts $\mathbf{appdata}_{\text{client}}$.

Therefore, the attacker can obtain application data from the client by using the static private key of the server without interrupting the communication between the client and the server. It corresponds to breaking forward secrecy for client's data.

4.2 Replay Attack to E2EEP

1. An attacker can knows the static public information that the sender key ID **sender key ID**, ECDH public key $\mathbf{key}^{\mathbf{user1}}{}_{\text{public}}$, the recipient key ID **recipient key ID** and ECDH public key $\mathbf{key}^{\mathbf{user2}}{}_{\text{public}}$.
2. The attacker starts two sessions by initiating client2, and receives two ECDH public key $\mathbf{key}^{\mathbf{user2}}{}_{\text{public}}$.
3. The attacker sends the client2's ECDH public key $\mathbf{key}^{\mathbf{user2}}{}_{\text{public}}$ to client1.
4. Client1 returns **version**, **content type**, **salt**, **C**, **MAC**, **sender key ID** and **recipient key ID** to the attacker.
5. The attacker sends **version**, **content type**, **salt**, **C**, **MAC**, **sender key ID** and **recipient key ID** to client2 in the first session.
6. The attacker sends **version**, **content type**, **salt**, **C**, **MAC**, **sender key ID** and **recipient key ID** send to client2 in the second session.

Since **MAC** is valid both for the first session and the second session, client2 completes both sessions. It corresponds to the reply attack.

4.3 Spoofing Attack to E2EEP

1. An attacker can know the static information that the sender key ID **sender key ID**, ECDH public key $\mathbf{key}^{\mathbf{user1}}{}_{\text{public}}$, the recipient key ID **recipient key ID** and ECDH public key $\mathbf{key}^{\mathbf{user2}}{}_{\text{public}}$.

2. The attacker starts a session by initiating client2, and receives ECDH public key $\mathbf{key}^{\mathbf{user2}}{}_{\text{public}}$.
3. The attacker randomly generates a fake sender key ID **sender key ID'**.
4. The attacker sends client2's ECDH public key $\mathbf{key}^{\mathbf{user2}}{}_{\text{public}}$ and the recipient key ID **recipient key ID** to client1.
5. Client1 returns **version**, **content type**, **salt**, **C**, **MAC**, sender key ID **sender key ID** and recipient key ID **recipient key ID** to the attacker.
6. The attacker sends **version**, **content type**, **salt**, **C**, **MAC**, the fake sender key ID **sender key ID'** and recipient key ID **recipient key ID** to client2.

Since **MAC** does not depend on the sender key ID, client2 accepts **MAC** if the sender key ID is replaced. Hence, the attacker can impersonate the fake sender to client2. It corresponds to the spoofing attack.

5 Summary

We verified the security of LINE Encryption (Version 1.0), (i.e., the TEP and the E2EEP) by ProVerif. In LINE documents [1,2], it is claimed that the TEP satisfies forward secrecy (i.e., even if the static private key is leaked, security of encrypted application data before leakage is guaranteed). However, it is not clear if both client's and server's data must be protected. We clarify actual forward secrecy of the TEP by showing an attack to break forward secrecy for client's data. In addition, we found a replay attack and a spoofing attack of E2EEP.

Since all known attacks (the reply attack and the spoofing attack) and a new attack (breaking forward secrecy for client's data) are found, our result shows usefulness of ProVerif to verify security of messaging protocols like LINE. Finally, we note that the found attack for forward secrecy for client's data is not serious because the LINE security team says that client's data does not contain any sensitive information in the current implementation of LINE encryption. However, it may be potential vulnerability if an engineer use the TEP for another implementation.

References

1. LINE Encryption Overview (Ver. 1.0). https://scdn.line-apps.com/stf/linecorp/en/csr/line-encryption-whitepaper-ver1.0.pdf
2. LINE Encryption Status Report, 24 August 2018. https://linecorp.com/en/security/encryption_report
3. Espinoza, A.M., Tolley, W.J., Crandall, J.R., Crete-Nishihata, M., Hilts, A.: Alice and Bob, who the FOCI are they?: analysis of end-to-end encryption in the LINE messaging application. In: FOCI @ USENIX Security Symposium (2017)
4. Isobe, T., Minematsu, K.: Spoofing attack and forgery attack against LINE's end-to-end encryption. In: SCIS 2018 (2018). (in Japanese)
5. Blanchet, B., Smyth, B., Cheval, V., Sylvestre, M.: ProVerif 1.98. http://prosecco.gforge.inria.fr/personal/bblanche/proverif
6. McGrew, D., Viega, J.: The Galois/Counter Mode of Operation (GCM). Manuscript, May 2005. NIST website

The Anatomy of the HIPAA Privacy Rule: A Risk-Based Approach as a Remedy for Privacy-Preserving Data Sharing

Makoto Iguchi(✉), Taro Uematsu, and Tatsuro Fujii

Kii Corporation, Tokyo 107-0052, Japan
{makoto.iguchi,taro.uematsu,tatsuro.fujii}@kii.com

Abstract. This paper explores the effectiveness of a risk-based approach methodology in constructing systematic standards for privacy-conscious data sharing and disclosure. We consider the HIPAA (Health Insurance Portability and Accountability Act of 1996) Privacy Rule as an example and show that the data disclosure methods defined in the HIPAA Privacy Rule are well-constituted, by assessing the privacy risks of each disclosure method. We further explore factors that contribute to the success of the HIPAA Privacy Rule and discuss how we can leverage these factors as a reference for constructing privacy-conscious and systematic data disclosure rules and regulations in other domains.

1 Introduction

With the growth of the digital economy, the sharing of data among multiple parties has become common practice. Data sharing is traditionally performed among parties within the same organization, but now we are seeing data sharing activities being performed among different organizations, and even with the public.

Although such a data sharing model enables us to utilize data to its fullest extent, it introduces new risks. This is particularly true when the shared target consists of personal data. Appropriate safety measures, such as data de-identification and contract signing, need to be applied in order to prevent possible privacy infringements. One problem is that appropriate safety measures are often not apparent. For example, some existing regulations state that data must be de-identified before being shared with the public, but an actual method for de-identifying the data is not explicitly defined in the regulations. To enjoy the benefits of data sharing, it is essential to introduce some straightforward standards with concrete definitions of the required safety measures.

The goal of this paper is to demonstrate that a risk-based approach methodology, which is sometimes regarded as a case-by-case approach, is useful for constructing straightforward rules and standards for realizing privacy-conscious

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 174–189, 2018.
https://doi.org/10.1007/978-3-319-97916-8_12

data sharing. We will briefly introduce existing risk-based approaches in the context of data sharing in Sect. 2. In Sect. 3, we will present a simplified risk-based approach framework. We will employ this framework in Sect. 4 to analyze the HIPAA (Health Insurance Portability and Accountability Act of 1996) Privacy Rule and clarify how the risk-based approach contributes to the construction of easy-to-follow data sharing methodologies. Section 5 explores what can be learnt from the HIPAA Privacy Rule analysis, and finally Sect. 6 concludes the paper.

2 Related Work

Applying the concept of a risk-based approach in the domain of data security and privacy protection assessment is a well-known technique. Various risk-based approach methodologies for assessing security risks to data's confidentiality, integrity, and availability have been proposed by many parties, including standard-setting bodies like the International Organization for Standardization (ISO) [1], non-regulatory agencies like the National Institute of Standards and Technology (NIST) [2], and online communities such as the Open Web Application Security Project (OWASP) [3].

As the sharing of data among different organizations and with the public has become more popular, the risk-based approach has been extended for assessing the privacy risks of data that are intentionally disclosed externally [4]. The proposed methodologies include a framework issued by the Commission Nationale de l'Informatique et des Libertés (CNIL) [5], a code of practice issued by the Information Commissioner's Office (ICO) [6], guidelines published by the ISO [7] and the Information and Privacy Commissioner of Ontario [8], and a study conducted by El Emam [9].

The main contribution of this paper is not to introduce a new risk-based approach. We do propose a risk-based approach framework for assessing data privacy risks in this paper, but rather the framework is merely presented as a tool for analyzing the HIPAA Privacy Rule. The originality of this paper lies in the re-evaluation of the HIPAA Privacy Rule in the context of a risk-based approach and in the extraction of factors that contribute to the construction of easy-to-follow safety rules and standards.

3 Risk-Based Approach Framework

In this section, we will formulate a risk-based approach framework for quantifying privacy risks of shared data. The framework is later employed in Sect. 4 to analyze the HIPAA Privacy Rule.

As previously mentioned in Sect. 2, applying a risk-based approach to assessing data privacy risks is a well-known technique. Various risk-based methods have been proposed for evaluating data security and privacy risks. Although these all aim towards the same goal, their assessment approaches are slightly different.

We study existing methodologies and formulate a simplified risk-based approach framework based on this study. In this framework, data privacy risks are assessed through the following steps:

Step 1: Clarify a privacy risk.
Step 2: Assess the magnitude of the risk as the severity level.
Step 3: Assess the feasibility of the risk occurring as the likelihood level.
Step 4: Assess the overall data privacy risk based on its severity and likelihood levels.

3.1 Step 1: Clarifying a Privacy Risk

The first step is to clarify a data privacy risk that needs to be assessed. In this paper, we define a data privacy risk as the probability of the shared data being compromised and resulting in harm (i.e., physical, material, or moral damages) to the data subject. The clarification step involves the analysis of "how" the shared data could be compromised and "who" could compromise the shared data.

The analysis of "how" is performed by predicting how malicious data recipients will compromise the data and infringe on the privacy of the data subject. As explored in various studies [10–13], malicious data recipients compromise data by (1) linking the data with other data, (2) re-identifying an original data subject, and (3) misusing the data to harm the data subject. The result of the analysis is used to score the data's severity level in Step 2.

The analysis of "who" is performed by evaluating the data recipients and how the data sharing is performed. Factors such as the attributes of the data recipients and method used for sharing the data should be taken into consideration. The result of this analysis is used to score the data's likelihood level in Step 3.

3.2 Step 2: Assessing Severity Level

The severity level indicates the extent of damage that will be triggered if the shared data is compromised. This is determined by the data's identification and prejudicial levels.

Identification Level. The identification level evaluates how easy it is to re-identify the data subject. If the re-identification is easy, we consider the data to be high risk and set a high score on the data's identification level.

Some of the factors that need to be addressed when assessing the identification level are as follows:

– The amount of personally identifiable information (PII) in the shared data: Identifying a data subject from data containing many PII items (e.g., first

name, last name, and birthday) is usually considered to be easier than identifying a subject from data with little PII (e.g., first name only). The identification level of the former type of data should thus be scored higher than that of the latter type.
- The availability of auxiliary data: If there are auxiliary data that are easily obtainable and can be linked to the shared data, the risk of the data subject being re-identified could be higher than when such data are not available.
- De-identification techniques applied to shared data: Various techniques such as randomization, generalization, and pseudonymization have been proposed for reducing the re-identification risk of data [14,15]. If any of these techniques are applied to shared data, this should be incorporated in assessing the identification level.

In this paper, we use the following simple three-grade evaluation scale for the identification level.

- HIGH: Identifying the data subject is extremely easy.
- MEDIUM: Identifying the data subject is not easy, but not impossible.
- LOW: Identifying the data subject is virtually impossible.

Prejudicial Level. The prejudicial level evaluates how much damage would be caused if the shared data is compromised. For example, the risk for data that contains a customer's credit card number together with the security code should be assessed as higher than that of data containing only the customer's email address. The prejudicial level of the former type of data should thus be scored higher than that of the latter type.

When assessing the prejudicial level, the following damage type should be addressed [5]:

- Physical damage, such as loss of amenity, disfigurement, or economic loss related to physical integrity.
- Material damage, such as loss incurred or lost revenue concerning a data subject's assets.
- Moral damage, such as physical or emotional suffering.

As for the severity level, we use the following simple three-grade evaluation scale for the prejudicial level.

- HIGH: The data subject may encounter significant consequences, which they may not overcome (e.g., loss of life).
- MEDIUM: The data subject may encounter some inconveniences, which they should be able to overcome with some difficulties (e.g., financial loss).
- LOW: The data subject will not be affected or may encounter a few negligible inconveniences, which they will overcome with no problem (e.g., receiving SPAM messages).

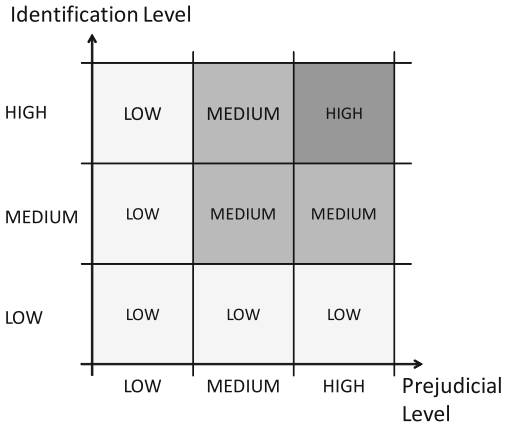


Fig. 1. Severity level matrix

Determining the Severity Level. The severity level of shared data is determined by the data's identification and prejudicial levels using the matrix shown in Fig. 1.

As shown in Fig. 1, a HIGH identification or prejudicial level score alone does not yield a HIGH severity level. For example, the data "Mr. John Smith (age xx) who lives in YYY drinks a glass of water every day" has a HIGH identification level score, but its severity level is LOW because the prejudicial level score is LOW. Likewise, the data "A woman who lives somewhere in Japan will pass away if she takes a drug X after drinking some alcohol" has a HIGH prejudicial level score, but its severity level is LOW because the identification level score is LOW.

The severity level assessed here will be used in Step 4 when we evaluate the overall data privacy risk.

3.3 Step 3: Assessing Likelihood Level

The likelihood level indicates the probability of shared data being compromised. The likelihood level is determined by the data's threat and vulnerability levels.

Threat Level. The threat level evaluates the probability of data recipients attempting to compromise shared data.

To assess the threat level, the following attributes of the data recipients should be taken into account:

– Size: If the data is disclosed to a large number of people, then the probability of the shared data being compromised is higher than when the data is disclosed to a handful of people.

- Motive: If the data recipients have some particular motivation to compromise the shared data, then this fact must be considered when assessing the threat level.
- Trust: If the trustworthiness of the data recipient is questionable for certain reasons, then the threat level should be set as high.

Again, we use the following simple three-grade evaluation scale for the threat level.

- HIGH: The chance of data recipients attempting to compromise the shared data is high.
- MEDIUM: The chance of data recipients attempting to compromise the shared data is remote.
- LOW: The chance of data recipients attempting to compromise the shared data is low.

Vulnerability Level. The vulnerability level scores how weak the shared data is. In other words, the vulnerability level quantifies the levels of safeguards implemented for preventing the occurrence of data compromise.

The vulnerability level can be evaluated by checking how the following safeguards are implemented.

- Administrative safeguards: The policies and procedures implemented to prevent data privacy infringement (e.g., documentation, training, and contracting).
- Physical safeguards: The control to ensure that the shared data is physically protected (e.g., locating a secure zone).
- Technical safeguards: The technology and policies implemented to protect data privacy (e.g., encrypting and auditing).

The vulnerability level can also be assessed by checking the model employed to disclose the data, as described by Elliot et al. [16]:

- Open access (unrestricted access): Highly vulnerable data disclosure model, in which anyone can access the data. No restrictions are applied on what the recipients can do with the data.
- Delivered access: A restricted form of data disclosure, in which access to the data is restricted to those recipients who have agreed to specific conditions concerning access to the data.
- On-site safe settings: More restricted form of data disclosure, in which the data is only accessible from a particular location.
- Virtual access: A date disclosure model supporting limited "remote" access. For example, analysis servers allow outsiders to analyze the data, while not allowing them to access the original data directly.

As for the other levels, we use the following simple three-grade evaluation scale for the vulnerability level.

– HIGH: The current safeguards are insufficient.
– MEDIUM: The current safeguards are working, but there is some room for improvement.
– LOW: The current safeguards are sufficient.

Determining the Likelihood Level. The likelihood level of the shared data is determined by the data's threat and vulnerability levels using the matrix shown in Fig. 2.

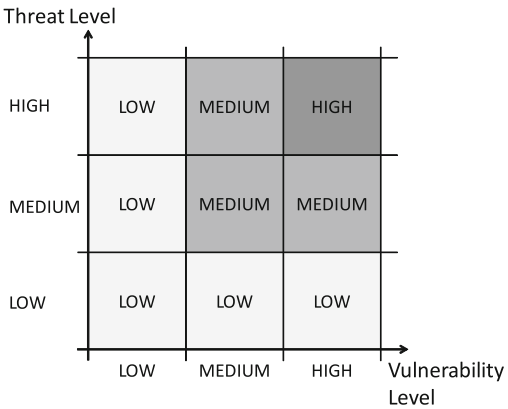


Fig. 2. Likelihood level matrix

As shown in Fig. 2, a HIGH threat or vulnerability level score alone does not yield a HIGH likelihood level. Even if the data recipients are not trustworthy (a HIGH threat level score), we can keep the likelihood level LOW by implementing strong safeguards (a LOW vulnerability level score). Likewise, implementing few safeguards (a HIGH vulnerability level score) might be acceptable if the data recipients are trustworthy (a LOW threat level score).

The likelihood level assessed here will be used in Step 4 when we evaluate the overall data privacy risk.

3.4 Step 4: Assessing Overall Data Privacy Risk

Finally, we assess the privacy risk of the data using a combination of the severity and likelihood levels. Figure 3 presents a matrix for combining the severity and likelihood levels.

The overall data privacy risk is to be interpreted as follows:

– If the data privacy risk is LOW, this represents an ideal situation.
– If the data privacy risk is MEDIUM, the situation is acceptable. Should we need to reduce the privacy risk to LOW, we can do so by improving the severity and/or likelihood levels if scored as MEDIUM.

– If the data privacy risk is HIGH or CRITICAL, this is not acceptable. We must improve the severity and/or likelihood levels if scored as HIGH.

To improve the severity level, we need to reduce the data's identification level. For instance, we should explore whether we can apply some de-identification techniques and lower the risk of re-identification.

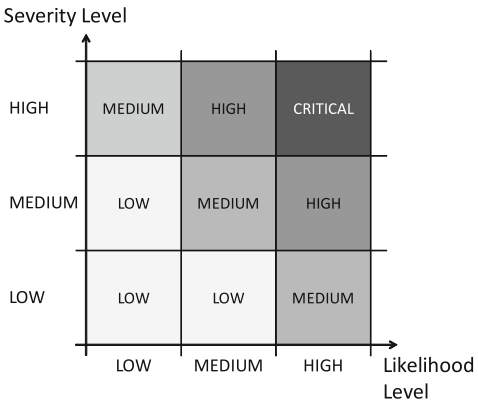


Fig. 3. Overall risk matrix

To improve the likelihood level, we need to reduce the data's vulnerability level. For example, the applied safeguards should be revisited to bring the likelihood level down to the acceptable level.

4 Analysis of the HIPAA Privacy Rule Using the Risk-Based Approach

In this section, we analyze the HIPAA Privacy Rule using our risk-based approach framework in order to clarify the effectiveness of the risk-based approach in constructing a straightforward data sharing method. We will briefly overview the HIPAA Privacy Rule and three data sharing methods that are defined in the Privacy Rule. Then, we will analyze the HIPAA Privacy Rule using our risk-based approach framework and describe how the three data sharing methods balance the data severity and likelihood levels to keep the data privacy risks at an acceptable level.

4.1 HIPAA Overview

The Health Insurance Portability and Accountability Act of 1996 (HIPAA) [17] is a set of regulations in the United States for handling protected health information securely and efficiently. The HIPAA, together with its expansion in 2009 by

the Health Information Technology for Economic and Clinical Health (HITECH) Act, establishes a set of standards for protecting the security and privacy of protected health information. There are pros and cons on the HIPAA, but some studies report that its primary effectiveness is acknowledged even by those who are against the regulations [18].

The HIPAA standards apply to covered entities and business associates (45 CFR §160.103).

- Covered Entity: Health care provider (e.g., doctor or pharmacy), health plan (e.g., health insurance company), or health care clearinghouse (e.g., billing house or community health information system).
- Business Associate: Organization or person that provides data transmission services concerning protected health information to a covered entity. For example, a cloud service provider that creates, receives, maintains, or transmits protected health information on behalf of a covered entity is explicitly defined as a business associate [19].

There are two primary rules covered in HIPAA: The Security Rule and Privacy Rule. The Security Rule sets national standards for protecting the confidentiality, integrity, and availability of protected health data, while the Privacy Rule sets national standards for protecting data subjects' privacy. The methodologies for data sharing are covered in the Privacy Rule, and so we will mainly focus on this.

4.2 Data Disclosure Method Covered in the HIPAA Privacy Rule

The HIPAA Privacy Rule defines the following three methods of protected health information disclosure. For each method, the HIPAA Privacy Rule defines the specification of the shared data set and the required safeguards that need to be applied to the data recipients [20].

Disclosure of Protected Health Data as is (Raw Data). The first data disclosure method consists of sharing the protected health information as is. This pattern is applied when a covered entity delegates some of their health care activities and functions to a business associate. For instance, a covered entity will disclose protected health information as is when they delegate the management of the information to a cloud service provider.

When a covered entity is to disclose protected health information in this manner, the HIPAA Privacy Rule requires that the covered entity supervises their business associate to ensure the protected health information is treated securely (i.e., HIPAA Security Rule compliance). In particular, the covered entity must sign a business associate agreement with the business associate to obtain satisfactory assurance and regulate the usage of the disclosed data.

Disclosure of De-identified Data. The second data disclosure method comprises the sharing of de-identified health information. Once protected health information is de-identified, the information can no longer be linked with its original data subject. Such information is no longer considered as protected health information, and a covered entity can freely disclose such information to anyone in any manner.

The HIPAA Privacy Rule defines two methods of de-identifying protected health data: (1) Expert Determination method (45 CFP §164.514(b)(1)) and (2) Safe Harbor method (45 CFR §164.514(b)(2)).

The Expert Determination method allows an expert to analyze the data and design a method of de-identifying protected health information. The HIPAA Privacy Rule requires that the chosen method must minimize the risk of re-identification to be "very small."

The Safe Harbor method provides an easy-to-follow rule. This method allows a covered entity to de-identify protected health information by removing 18 specific identifiers. These identifiers are summarized in Table 1. Once these identifiers are removed, the data is regarded as de-identified.

Disclosure of Limited Data Set. The third data disclosure method comprises the sharing of a limited data set (45 CFR §164.514(e)). A limited data set contains partially de-identified data. The HIPAA Privacy Rule allows a covered entity to create a limited data set by removing 16 specific identifiers. These identifiers are summarized in Table 1. As shown in Table 1, a couple of fields that must be removed in the Safe Harbor method are allowed in a limited data set.

A limited data set may only be disclosed for the purposes of research, public health, or health care operations. Before sharing a limited data set, a data use agreement covering the following provisions must be signed by the data recipients (45 CFR §164.514(e)(4)):

- Acceptable uses and disclosures of the limited data set.
- Deployment of the appropriate safeguards to prevent use or disclosure of information that is inconsistent with the agreement.
- Assurance that any agents to whom the limited data set is provided agree to the same restrictions and conditions that apply to the data set recipient.
- Re-identification of information or contacting data subjects is prohibited.

In general, a data use agreement provides a weaker legal binding compared to a business associate agreement.

4.3 Analysis of Data Disclosure Methods with a Risk-Based Approach

The HIPAA Privacy Rule does not explicitly mention the risk-based approach[1]. The only mention in connection with the HIPAA Privacy Rule is found in a

[1] In the HIPAA Security Rule, assessing potential risks concerning the confidentiality, integrity, and availability of health information is defined as a required obligation (45 CFR §164.308(a)(1)(ii)(A)).

Table 1. De-identified data set and limited data set

Identifier	De-identified data (Safe Harbor method)	Limited Data Set
Names	Remove	Remove
Geographic information	Remove information smaller than state. 3-digit zip code allowed for an area where >20,000 people live	Remove postal address information other than city, town, state, or zip code
Dates	Remove all elements of dates (except year); age and any date (including years) if age is >89	OK
Phone Numbers	Remove	Remove
Fax Numbers	Remove	Remove
Email Addresses	Remove	Remove
Social Security Numbers	Remove	Remove
Medical Record Numbers	Remove	Remove
Health Plan Beneficiary Numbers	Remove	Remove
Account Numbers	Remove	Remove
Certificate/License Numbers	Remove	Remove
Vehicle Identifiers and Serial Numbers	Remove	Remove
Device Identifiers and Serial Numbers	Remove	Remove
URLs	Remove	Remove
IP addresses	Remove	Remove
Biometric Identifiers	Remove	Remove
Full-face Photographs	Remove	Remove
Any Other Unique Identifying Numbers, Characters, or Code	Remove	OK

guidance document published by the United States Department of Health and Human Services [20]. The guidance states that an expert should assess the data risk when designing de-identification procedures in the Expert Determination method.

Although not explicitly claimed, we can see that the three data disclosure methods defined in the HIPAA Privacy Rule are carefully designed using the risk-based approach. The following summarizes the results of analyzing the data disclosure methods. In the analysis, we leverage the risk-based approach framework we explained in Sect. 3.

Disclosure of Protected Health Data (as is)

- The identification level is assessed as HIGH, because the data is not de-identified at all.
- The prejudicial level is fixed to HIGH, because the data is protected health data.

- The threat level is assessed as LOW, because the HIPAA Privacy Rule obliges a covered entity to select a trustworthy business associate.
- The vulnerability level is also assessed as LOW, because the HIPAA Privacy Rule obliges a covered entity to implement an adequate level of safeguards, including the signing of a business associate agreement.

Disclosure of De-identified Data

- The identification level is assessed as LOW, because the data must be fully de-identified, regardless of whether the Expert Determination or Safe Harbor method is used. For example, a study by Benitez and Malin [21] showed that data that had been de-identified using the Safe Harbor method had a re-identification risk ranging from 0.01% to 0.25%.
- The prejudicial level is fixed to HIGH, because the data is protected health data.
- The threat level is assessed as HIGH, because the de-identified data can be disclosed to anyone.
- The vulnerability level is assessed as HIGH, because the de-identified data can be disclosed freely with no safeguards.

Disclosure of a Limited Data Set.

- The identification level is assessed as MEDIUM, because the data is partially de-identified. The study by Benitez and Malin [21] showed that the re-identification risk for a limited data set ranges from 10% to 60%.
- The prejudicial level is fixed to HIGH, because the data is protected health data.
- The threat level is assessed as MEDIUM. The purpose of the data disclosure is restricted to research, public health, or health care operations, and so a covered entity is indirectly obliged to select somewhat trustable recipients. In addition, a limited data set is still regarded as protected health information, and so the covered entity is required to choose data recipients who are capable of complying with the HIPAA Security Rule.
- The vulnerability level is assessed as MEDIUM. The data recipients are not fully regulated in the way that a business associate is controlled by a business associate agreement. Instead, the recipients are bound by a data use agreement, which provides a more relaxed level of control over the data recipient.

Table 2 summarizes how the data privacy risk is assessed for each data disclosure method.

As shown in Table 2, we can see that the HIPAA Privacy Rule, although not explicitly stated, balances the severity and likelihood levels effectively in all three data disclosure methods. For all methods, the overall data privacy risk is controlled to an acceptable level (MEDIUM) by requiring an appropriate level of safeguards to be implemented on the basis of the shared data.

Table 2. Analysis of data sharing methods using the risk-based approach

Method of Disclosure	Identification Level	Prejudicial Level	Severity Level	Threat Level	Vulnerability Level	Likelihood Level	Overall Privacy Risk
Disclosure of Raw Data	HIGH	HIGH	HIGH	LOW	LOW	LOW	MEDIUM
Disclosure of De-identified Data	LOW		LOW	HIGH	HIGH	HIGH	MEDIUM
Disclosure of Limited Data Set	MEDIUM		MEDIUM	MEDIUM	MEDIUM	MEDIUM	MEDIUM

5 Leveraging the Success of HIPAA Privacy Rule

Following the HIPAA Privacy Rule analysis in Sect. 4, we will explore how we can make good use of the analysis result.

We will consider the following two findings and explore how we can leverage them when to construct new rules and regulations for realizing privacy-conscious data sharing in other data domains.

- Properly scoping the target data contributes to an effective assessment of the severity level and the definition of straightforward safety measures.
- Covering data sets with multiple identification levels contributes to flexibility in supporting various data sharing requirements.

5.1 Properly Scoping the Target Data

One of the main factors in why the HIPAA Privacy Rule could formulate systematic data sharing methods is because it is clear in its target. The target of the HIPAA Privacy Rule is focused on protecting the data privacy of "protected health information" only.

The fact that the target data is strictly limited to health information allows the HIPAA Privacy Rule to enjoy the following benefits:

- The content of the target data is fixed. This fact makes the magnitude of the potential damage evident, making the assessment of the target data's prejudicial level definitive (i.e., the prejudicial level is fixed to HIGH).
- The personally identifiable information included in the target data is fixed. This fact supports the construction of straightforward rules for de-identifying the data. Both the Safe Harbor method and limited data set specification provide precise rules, by stating identifiers that are to be removed. A covered entity could just follow the instructions to convert protected health information to either a de-identified or limited data set.
- The above two benefits make the formulated de-identification methods verifiable. Because the target data is explicitly fixed and the de-identification methods are systematically defined, Benitez and Malin [21] could later re-evaluate the de-identification methods by reassessing the re-identification risk.

When we want to formulate new secure data sharing rules and regulations, we should attempt to follow the same practices. It is essential to scope the target data correctly. If we try to construct data sharing rules without setting a clear data scope, then we will most likely arrive at vague and unsystematic rules for the following reasons:

- The content of the target data will be unfocused, making the assessment of the prejudicial level difficult and meaningless.
- The personally identifiable information included in the target data will be unorganized, making the formulation of systematic de-identifying rules difficult.
- Because the target data is unorganized, it will be difficult to later re-access the validity of the formulated de-identification methods.

5.2 Covering Data Sets with Multiple Identification Levels

The HIPAA Privacy Rule defines three types of data sets (i.e., raw data, de-identified data, and limited data set) with different identification levels. The HIPAA Privacy Rule also defines distinct data disclosure methods for each of these.

Because the HIPAA Privacy Rule provides a selection of data set types for disclosure, a data provider can select a method of sharing their health data in accordance to their situation and follow the rules to minimize the data privacy risks to an acceptable level.

Covering multiple data sets with different identification levels is a good practice to follow when we want to formulate new privacy-conscious data sharing rules and regulations. The handling of a data set with the MEDIUM identification level will play an especially crucial role in constructing systematic rules. This is because creating fully de-identified data is not trivial in some domains. An opinion issued by the EU's Article 29 Data Protection Working Party [14] states that a perfectly de-identified data set is not easy to generate in some situations. The opinion also states that a de-identified data set can still contain residual privacy risks to data subjects. By incorporating data set with the MEDIUM identification level into a rule, we will be able to support such cases.

As an illustrative example, consider how we can adapt some of recent data disclosure techniques that rely on data encryption (e.g., the homomorphic encryption privacy-preserving protocol) [15] into new regulations. We can map these algorithms for the "MEDIUM" strategy as follows:

- In most countries, encrypted data is not accepted as fully de-identified as long as its decrypting key exists. The identification level of the encrypted data should thus be assessed as MEDIUM. This means that the severity level will be assessed as MEDIUM.
- Because the severity level is MEDIUM, some relaxed controls must be enforced on the data recipients to control the likelihood level to MEDIUM. In the case of the homomorphic encryption protocol, the protocol only allows

data recipients to compute certain functions on the data, while not allowing them to access the original content. In a sense, the protocol technically regulates how the data recipients can process the data. This is analogous to the HIPAA Privacy Rule contractually regulating the data recipients through a data use agreement. Therefore, one possible rule we could formulate is to define the application of the homomorphic encryption protocol as an adequate technical safeguard that provides a MEDIUM vulnerability level.

6 Conclusion

This paper explored the effectiveness of a risk-based approach to constructing systematic rules and regulations for privacy-preserving data disclosure. We first defined a simplified risk-based framework for evaluating data privacy risks. The proposed framework was then applied to analyze the HIPAA Privacy Rule. We identified that the three data disclosure models defined in the HIPAA Privacy Rule control the data privacy risks by balancing the severity and likelihood levels. On the basis of the analysis, we explored how to learn from the success of the HIPAA Privacy Rule for constructing new privacy-preserving data sharing rules. We identified two key factors that should be considered to deliver effective and systematic data sharing rules and regulations.

Naturally, other factors should be addressed when constructing such rules. The regulations enforced in each country are different, and these need to be taken into consideration when formulating data sharing rules. However, we believe that our findings will contribute to the realization of some straightforward standards for privacy-preserving data sharing.

References

1. ISO/IEC 27005:2011: Information technology - Security techniques - Information security risk management. Standard, International Organization for Standardization (ISO) (2011)
2. Ross, R.S.: Nist sp 800–30 rev. 1: Guide for conducting risk assessments. Technical report (2012). https://dx.doi.org/10.6028/NIST.SP.800-30r1
3. Open Web Application Security Project (OWASP): Owasp risk rating methodology (2016). https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology
4. Garfinkel, S.L.: Nistir 8053: De-identification of personal information. Technical report, October 2015. https://doi.org/10.6028/nist.ir.8053
5. Commission Nationale de l'Informatique et des Libertés (CNIL): Methodology for Privacy Risk Management, Translation of June 2012 edition (2012)
6. Information Commissioner's Office (ICO): Conducting Privacy Impact Assessments Code of Practice (2014)
7. ISO/IEC 29134:2017: Information technology - Security techniques - Guidelines for privacy impact assessment. Standard, International Organization for Standardization (ISO) (2017)
8. Information and Privacy Commissioner of Ontario: De-identification Guidelines for Structured Data (2016)

9. El Emam, K., Arbuckle, L.: Anonymizing Health Data: Case Studies and Methods to Get You Started, 1st edn. O'Reilly Media, Inc. (2013)
10. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: 2008 IEEE Symposium on Security and Privacy (sp 2008), pp. 111–125, May 2008
11. Narayanan, A., Shmatikov, V.: De-anonymizing social networks. In: 2009 30th IEEE Symposium on Security and Privacy, pp. 173–187, May 2009
12. de Montjoye, Y.A., Radaelli, L., Singh, V., Pentland, A.: Unique in the shopping mall: on the reidentifiability of credit card metadata. Science **347**(6221), 536–539 (2015)
13. Douriez, M., Doraiswamy, H., Freire, J., Silva, C.T.: Anonymizing nyc taxi data: Does it matter? In: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 140–148, October 2016
14. European Commission Article 29 Data Protection Working Party: Opinion 05/2014 on Anonymisation Techniques (2014)
15. Mendes, R., Vilela, J.P.: Privacy-preserving data mining: methods, metrics, and applications. IEEE Access **5**, 10562–10582 (2017)
16. Elliot, M., Mackey, E., O'Hara, K., Tudor, C.: The Anonymisation Decision Making Framework. UKAN, United Kingdom (2016)
17. U.S. Department of Health & Human Services, Office for Civil Rights: Health information privacy (2015). https://www.hhs.gov/hipaa/index.html
18. Chesanow, N.: Is hipaa creating more problems than it's preventing? (2013). https://www.medscape.com/viewarticle/810648
19. U.S. Department of Health & Human Services, Office for Civil Rights: Guidance on hipaa & cloud computing (2017). https://www.hhs.gov/hipaa/for-professionals/special-topics/cloud-computing/index.html
20. U.S. Department of Health & Human Services, Office for Civil Rights: Guidance regarding methods for de-identification of protected health information in accordance with the health insurance portability and accountability act (hipaa) privacy rule (2015). https://www.hhs.gov/hipaa/for-professionals/privacy/special-topics/de-identification/index.html
21. Benitez, K., Malin, B.: Evaluating re-identification risks with respect to the hipaa privacy rule. J. Am. Med. Inf. Assoc. **17**(2), 169–177 (2010)

Secret Sharing

Improvements to Almost Optimum Secret Sharing with Cheating Detection

Louis Cianciullo(✉) and Hossein Ghodosi

James Cook University, Townsville 4811, Australia
{louis.cianciullo,hossein.ghodosi}@jcu.edu.au

Abstract. Secret sharing allows a secret s to be distributed amongst n participants in the form of shares. An authorised set of these participants is then able to reconstruct s at a latter date by pooling their shares. Secret sharing with cheating detection capability (SSCD) allows participants to detect the submission of faulty or modified shares. Within this field researchers consider two different models of security, the OKS model and the CDV model.

In SPACE 2015 Jhanwar and Safavi-Naini (JS) presented two SSCD schemes, one developed under each of the security models. We prove that both of these schemes fail to detect cheating. We then show that with some modifications both schemes can be made secure. The resulting schemes have near optimal share size, support operations from an arbitrary finite field and provide a high level of security even if the secret domain is small. The first of these schemes is devised under the OKS model and is the most efficient of its kind, whilst the second is devised under the CDV model and is as efficient as the current best solution.

1 Introduction

Secret sharing is a cryptographic primitive independently discovered by Blakley [4] and Shamir [15] in 1979. In a secret sharing scheme, a secret s is distributed amongst n participants, $P_1, \cdots, P_n$, in the form of shares, $V_1, \cdots, V_n$, such that P_i obtains V_i for $1 \leq i \leq n$. At a latter date an authorised subset of participants is able to reconstruct s by combining their shares. We can classify authorised and unauthorised subsets of participants by means of an access structure Γ, such that the subset $\mathcal{A} \in \Gamma$ is an authorised subset that can reconstruct s, whilst $A' \notin \Gamma$ is an unauthorised subset that cannot. In a perfect secret sharing scheme if an unauthorised subset attempt to reconstruct s then they should obtain no additional information regarding s, other than what their individual shares give them.

A secret sharing scheme is more formally defined as a pair of algorithms `SHARE` and `REC`. The `SHARE` algorithm is executed by a trusted entity $\mathcal{D}$, known as the dealer and `REC` is executed by a second, separate entity $\mathcal{C}$, the combiner. We define these algorithms in the following manner:

L. Cianciullo—This research is supported by an Australian Government Research Training Program (RTP) Scholarship.

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 193–205, 2018.
https://doi.org/10.1007/978-3-319-97916-8_13

`SHARE`$(s) \rightarrow (V_1, \ldots, V_n)$: A probabilistic algorithm that takes a secret s and produces n random shares.

`REC`$(\gamma) \rightarrow s'$: A deterministic algorithm that takes a subset of shares γ and outputs a secret s'.

Given a secret $s \in \mathcal{S}$ that is used to compute shares $V_1, \cdots, V_n$ and a subset of these shares, denoted by γ, used to compute a secret s' then the following properties will hold (assuming that all participants follow the protocol exactly):

$$\Pr[s' = s \mid \gamma \in \Gamma] = 1$$

$$Pr[s' = s \mid \gamma \notin \Gamma] = Pr[s' = s]$$

These properties state that in a perfect secret sharing scheme a set of unauthorised participants cannot reduce their uncertainty of the secret. It is a well known fact that in a perfect secret sharing scheme the share size cannot be smaller than the secret itself. That is, $|V_i| \geq |S|$ where $|V_i|$ denotes the maximum size of a given participant's share i.e. $\max_{1 \leq i \leq n}(|V_i|)$.

In a (t, n) threshold secret sharing scheme we define an unauthorised subset of participants as any subset γ' in which $|\gamma'| < t$ where $t \leq n$. It is this particular type of secret sharing scheme that we consider exclusively throughout the rest of this article.

In 1988 Tompa and Woll [16] demonstrated that Shamir's (t, n) threshold scheme [15] is vulnerable to share forgery. This is when one or more participants submit modified shares to $\mathcal{C}$, resulting in the reconstruction of a secret $s' \neq s$. Because of this, various stronger notions of secret sharing have been considered throughout the literature (see [2,7] for two examples). One such notion is secret sharing with cheating detection capability (SSCD).

SSCD considers the scenario in which corrupt participants, known as cheaters, modify their shares in order to trick other (honest) participants into reconstructing a false secret. An effective SSCD scheme can alert the honest participants to the submission of false shares. This type of protocol has applications in such things as robust secret sharing [8] and secure message transmission [11].

More formally a (t, n, δ) SSCD scheme is a threshold secret sharing scheme in which the probability of successful cheating occurring is less than or equal to δ [1]. To put this another way a (t, n, δ) SSCD scheme is a secret sharing scheme with a modified `REC` algorithm. In the definition that we consider, there are n participants, $P_1, \cdots, P_n$ with shares, $V_1, \cdots, V_n$ where t of these same participants will attempt to reconstruct the secret s. We assume that up to $t - 1$ of these participants are dishonest (cheaters) and wish to force the honest participants into accepting a secret s', where $s' \neq s$.

The goal of SSCD is to detect when such cheating occurs in order to prevent reconstruction of an invalid secret. Thus, the `REC` algorithm given for secret sharing is modified so that it either outputs s' (which may or may not be equal to the original secret) or a special symbol, $\perp$, which indicates that cheating has been detected.

For any such scheme with the above reconstruction protocol in which up to $t-1$ faulty shares, $V'_{i_1}, \cdots, V'_{i_{t-1}}$ and one unmodified share, V_{i_t} are submitted, the probability of failure, δ, is defined as:

$$\Pr[\texttt{REC}(V'_1, \cdots, V'_{t-1}, V_t) = \perp] \geq 1 - \delta$$

Conversely let s be a valid secret distributed to a set of n participants by `SHARE` and define s' as the secret computed by `REC` using shares (which may or may not be modified) from t of these same participants, then:

$$\Pr[s \neq s'] \leq \delta$$

An extensive amount of work has been produced on SSCD with the main goal being the reduction of share size. A key factor that greatly influences share size is the model of security a SSCD scheme is devised under. Typically schemes are devised under one of two models, the CDV model and the OKS model.

1.1 CDV Model

The CDV model was given by Carpentieri, De Santis and Vaccaro [6] in 1994. Schemes devised under this model consider the scenario in which the cheaters actually know the secret. Thus, such a scheme is only secure if the honest participants can detect cheating (with probability $1-\delta$) given that the cheaters already know the secret.

Ogata, Kurosawa and Stinson [14] give a bound for the share size of schemes devised under this model. As before let $|V_i|$ denote the maximum share size of a given participant. Let $|\mathcal{S}|$ denote the size of the secret domain and δ the probability of successful cheating occurring:

$$|V_i| \geq \frac{|\mathcal{S}| - 1}{\delta^2} + 1$$

1.2 OKS Model

In [14] Ogata, Kurosawa and Stinson introduced the OKS model. This model of security assumes that the cheaters do not know the secret. As a result of this the bound on share size for schemes developed under this model varies from the bound given by the CDV model:

$$|V_i| \geq \frac{|\mathcal{S}| - 1}{\delta} + 1$$

Remark 1. SSCD schemes devised under the OKS model typically have smaller share sizes than those devised under the CDV model. However, a common trait for schemes developed under the OKS model is that δ is dependent on $|\mathcal{S}|$ e.g. $\delta = \frac{1}{\sqrt{|\mathcal{S}|}}$ [13]. This is not the case for schemes devised under the CDV model as in these type of schemes the probability of successful cheating does not necessarily depend on the secret domain.

1.3 Our Contribution

First we present some flaws that exist in two SSCD schemes given in [10]. One of these schemes is developed under the OKS model and the other is developed under the CDV model. The flaws in question allow just one cheating participant to fool the other participants into accepting an incorrect secret. Secondly we show how to modify the schemes in [10], such that they are secure and satisfy the following desirable properties originally given in [12] by Obana and Tsuchida.

- **Capable of supporting an arbitrary finite field:** Computations can be done in a field of any characteristic.
- **Near optimal share size:** The share size is only slightly larger than the bound given for the particular model the scheme is developed under, where an optimal scheme is one that meets the bound.
- **Adequate level of security even if the secret domain is relatively small:** The probability of successful cheating occurring is no larger than $\frac{1}{|\mathcal{S}|}$ or can be set arbitrarily (as is the case for most schemes developed under the CDV model).

Whilst most schemes developed under the CDV model have these attributes there is only one other scheme developed under the OKS model that currently achieves this, and it is given by Obana and Tsuchida in [12]. The share size for this scheme is 2 bits larger than optimal whilst our scheme developed under the OKS model has a share size only 1 bit larger than optimal. Making our scheme is the most efficient yet.

The other scheme presented in this paper is developed under the CDV model and achieves the same share size as the scheme presented by Cabello et al. [5]. To the best of our knowledge this scheme has the smallest share size of any such scheme secure in the CDV model.

2 Preliminaries

2.1 Shamir's Secret Sharing Scheme

In Shamir's seminal paper [15] he introduced a (t, n) threshold secret sharing scheme in which each participant is given a point on a polynomial of degree $t-1$ for their share. Suppose that there are n participants, $P_1, \cdots P_n$ and that all computations are done in the field $\mathbb{F}_q$, where q is a prime number such that $q > n$. The protocol is as follows:

$\texttt{SHARE}(s) \rightarrow (V_1, \ldots, V_n)$: A probabilistic algorithm in which $\mathcal{D}$ picks a random polynomial, $f(x)$ of degree at most $t-1$, where $f(0) = s$. Participants are assigned the share $V_i = f(i)$ for $1 \leq i \leq n$.

$\texttt{REC}(\gamma) \rightarrow s'$: A deterministic algorithm that takes a subset of shares γ, where $|\gamma| \geq t$ and computes and outputs s' using Lagrange interpolation. Without

loss of generality assume γ is composed of shares from the first set of $|\gamma|$ participants i.e. $P_1, \cdots, P_{|\gamma|}$, then:

$$s' = \sum_{i=1}^{|\gamma|} V_i \prod_{\substack{1 \le i \le |\gamma| \\ i \ne j}} \frac{j}{j-i}$$

In Shamir's scheme it is assumed that all participants follow the protocol exactly, i.e. they always submit the correct shares for `REC`.

2.2 The Tompa and Woll Attack

Tompa and Woll [16] showed that Shamir's scheme is vulnerable to an attack in which malicious participants are able to obtain the secret s and force the honest participants into reconstructing $s' \neq s$. In fact, it is possible for just one cheater in a group of t participants to accomplish this. To carry out this attack successfully cheaters modify their shares in the following fashion.

Given a set of t participants $P_1, \cdots, P_t$ attempting to reconstruct the secret, assume that there is subset of dishonest participants within this set denoted by θ where $1 \le |\theta| \le t-1$. This set of cheaters compute a polynomial $\Delta(x)$ of degree at most $t-1$. They set $\Delta(i) = 0$ for $P_i \notin \theta$ (all of the honest participants) and $\Delta(0) = \beta$, which is an arbitrary value picked by the cheaters. $P_j \in \theta$ then submits the share $V_j + \Delta(j)$ where V_j is the share originally assigned to P_j. All dishonest participants submit these modified shares whilst $P_i \notin \theta$ submits his unaltered share V_i (which can be viewed as $V_i + \Delta(i)$).

This results in the reconstruction of $f'(x) = f(x) + \Delta(x)$ where $f(x)$ is the original polynomial used to compute and distribute shares. It is now easy for the cheaters to compute s as $f'(0) = s + \beta$.

2.3 JS SSCD Scheme Devised in OKS Model

In this section the scheme developed under the OKS model presented in [10] is reviewed. All computations are done in the field $\mathbb{F}_q$ where $q > 2n$.

`SHARE`$(s) \rightarrow (V_1, \ldots, V_n)$: $\mathcal{D}$ picks a random polynomial $f(x)$, of degree[1] at most $2t-2$, where $f(0) = s$. $\mathcal{D}$ also chooses $2n$ distinct public elements $\alpha_1, \cdots, \alpha_{2n}$. Each participant, P_i is given the share $V_i = (f(\alpha_i), f(\alpha_{i+n}))$ for $1 \le i \le n$.

`REC`$(\gamma) \rightarrow s'$: A subset, γ, of t participants (of which up to $t-1$ can be cheaters) submit shares to $\mathcal{C}$ who reconstructs $f'(x)$ using Lagrange interpolation. If the degree of $f'(x)$ is $2t-2$ or less, output $s' = f'(0)$ as the secret, otherwise output $\perp$.

[1] In [10] the degree of the polynomial is $2t$. This is because they set $t+1$ as the threshold value, this is not common practice in SSCD schemes which usually employ a threshold value of t.

2.4 JS SSCD Scheme Devised in CDV Model

Here we present the SSCD scheme devised under the CDV model presented in [10]. All computations are done in the field $\mathbb{F}_q$ where $q > 3n$.

$\texttt{SHARE}(s) \rightarrow (V_1, \ldots, V_n)$: $\mathcal{D}$ picks a random polynomial $f(x)$, of degree at most $3t-3$, where $f(0) = s$. $\mathcal{D}$ also chooses $3n$ distinct public points $\alpha_1, \cdots, \alpha_{3n}$. Each participant, P_i is given the share $V_i = (f(\alpha_i), f(\alpha_{i+n}), f(\alpha_{i+2n}))$ for $1 \leq i \leq n$.

$\texttt{REC}(\gamma) \rightarrow s'$: A subset, γ, of t participants (of which up to $t-1$ can be cheaters) submit shares to $\mathcal{C}$ who reconstructs $f'(x)$ using Lagrange interpolation. If the degree of $f'(x)$ is $3t-3$ or less, output $s' = f'(0)$ as the secret, otherwise output $\perp$.

3 Flaws in JS SSCD

This section is concerned with the security flaw inherent in the two schemes presented by Jhanwar and Safavi-Naini in [10]. It is shown that neither of these two schemes is secure. We begin by investigating their first scheme which was developed under the OKS model.

The authors state that this scheme will detect cheating with probability $\delta = 1 - \frac{1}{q}$ even if up to $t-1$ of the t participants are cheaters. The mechanism used for cheating detection relies on the fact that a degree $2t-2$ polynomial is uniquely defined by $2t-1$ points. So if t participants submit their shares then $\mathcal{C}$ obtains $2t$ points. If these shares are unaltered then they will all describe the same degree $2t-2$ polynomial $f(x)$. However, if one or more of these shares are changed in a random fashion then, with probability $\frac{1}{q}$, the collection of shares will describe a degree $2t-1$ polynomial.

For the above scenario in which shares are changed randomly the scheme is indeed secure. However, the scheme is not secure against the Tompa and Woll attack. Using this attack just one participant can cheat the other honest participants with probability 1, as demonstrated below.

Let θ denote the set of dishonest participants where $1 \leq |\theta| \leq t-1$. In order to cheat they must submit shares that are consistent with the honest participants' shares, i.e. all shares describe a polynomial of degree $2t-2$. In order to accomplish this they can use the Tompa and Woll attack to force reconstruction of $f'(x) \neq f(x)$, where $f'(x)$ is a polynomial of degree at most $2t-2$. To do so θ compute $\Delta(x)$ of degree at most $2t-2$ such that for all honest participants $P_i \notin \theta$, $\Delta(\alpha_i) = \Delta(\alpha_{i+n}) = 0$ and $\Delta(0) = \beta$. All cheaters $P_j \in \theta$ submit the modified shares $V'_j = (f(\alpha_j) + \Delta(\alpha_j), f(\alpha_{j+n}) + \Delta(\alpha_{j+n}))$. This will result in the reconstruction of a polynomial $f'(x) = f(x) + \Delta(x)$ that is of degree at most $2t-2$ with $f'(0) = \beta + s$. This occurs because of the well known homomorphic property of Shamir's secret sharing scheme [3]. This property states that given two sets of shares, $r_1, \cdots, r_z$ and $k_1, \cdots, k_z$, associated with the polynomials $R(x)$ and $K(x)$ respectively, the individual summation of these shares: $r_1 + k_1, \cdots, r_z + k_z$, will describe the polynomial $R(x) + K(x)$.

The second scheme devised by JS is devised under the CDV model. However, the cheating detection mechanism is the same as the first scheme described. This time however, $\mathcal{D}$ uses a polynomial of degree at most $3t-3$ to distribute shares and each participant is given three points on this polynomial as his share. Thus the exact same attack can be used against this scheme, with the cheaters computing a polynomial $\Delta(x)$ of degree $3t-3$.

4 Improvement to JS SSCD

In this section we show how to modify JS's SSCD schemes in order to add security. Both modified schemes are unconditionally secure with the same share size, secret space and probability of succesful cheating as claimed in the original protocols presented in [10]. The basic idea behind these corrected schemes is to use a slightly modified version of what Hoshino and Obana [9] call a check digit function.

In a cheater detection scheme that employs a check digit function, two types of shares are distributed: one directly related to the secret s, and the other related to a check digit $l = \mathcal{L}(s)$. At reconstruction of a secret s' and a check digit l', the combiner, $\mathcal{C}$ checks whether $l' = \mathcal{L}(s')$. If this is not the case then $\mathcal{C}$ concludes that cheating has occurred and outputs $\perp$, otherwise $\mathcal{C}$ assumes that s' is valid and broadcasts this value.

Our proposed scheme follows this basic definition. However, instead of constructing shares related to the check digit l and the secret s we instead distribute two or three shares (using the OKS or CDV models, respectively) to each participant corresponding to either two or three polynomials of degree at most $t-1$, respectively. These polynomials can then be used to reconstruct a larger degree polynomial $A(x)$ (of degree $2t-2$ or $3t-3$) as well as the check digit. Note that the definition of a check digit described here is slightly different to [9] in that for our schemes $l = s$. Thus we compare the reconstructed secret to the check digit, if the two are not equal then we reject s' and conclude that cheating has occurred. This is explained in greater detail in the following sections.

4.1 Proposed OKS-Secure Scheme

In this scheme a polynomial, $A(x)$ of degree at most $2t-2$, where $A(0) = s$, is used to construct two polynomials, $f(x)$ and $g(x)$ of degree at most $t-1$, which are then used to distribute shares to participants. Upon reconstruction both $f(x)$ and $g(x)$ are reconstructed and used to reconstruct $A(x)$. In order to check that no cheating has occurred, $\mathcal{C}$ checks that $A(0) = f(0) \cdot g(0)$. As with the scheme given by JS all computations are performed in $\mathbb{F}_q$ where q is a prime number and $q > 2n$. The `SHARE` and `REC` algorithms are as follows:

`SHARE`$(s) \rightarrow (V_1, \ldots, V_n)$: $\mathcal{D}$ picks a random polynomial $A(x) = s + Q_1 x + \cdots + Q_{2t-2}x^{2t-2}$ where $Q_j \in \mathbb{F}_q$ for $1 \leq j \leq 2t-2$. He then uses the first $2t-1$ points on $A(x)$ to construct two more polynomials in the following fashion:

$$f(x) = A(1) + A(2)x + \cdots + A(t)x^{t-1}$$

$$g(x) = \alpha + A(t+1)x + \cdots + A(2t-1)x^{t-1}$$

Where α is picked such that $s = A(1) \cdot \alpha$ and the other coefficients of both $g(x)$ and $f(x)$ are points on $A(x)$. Participant P_i is assigned the share $V_i = (f(i), g(i))$ for $1 \leq i \leq n$.

$\texttt{REC}(\gamma) \rightarrow s'$: A subset, γ, of t participants (of which up to $t-1$ can be cheaters) submit shares to $\mathcal{C}$ who reconstructs $f'(x)$ and $g'(x)$ using Lagrange interpolation. He then uses the coefficients of $f'(x)$ and $g'(x)$ (specifically the points $A'(1), \cdots, A'(2t-1)$) to reconstruct $A'(x)$. If $A'(0) = f'(0) \cdot g'(0)$ then $\mathcal{C}$ outputs $A'(0)$ as the secret, otherwise output $\perp$.

Security. To prove the security of the proposed **OKS-secure scheme** we begin by showing that it is perfect and then prove that except with probability δ a set of dishonest participants cannot cheat the honest participants into accepting a false secret.

Theorem 1. *The proposed **OKS-secure scheme** is perfect, i.e. fewer than t participants cannot reduce their uncertainty of the secret s.*

Proof. Say that the first $t-1$ participants with shares $V_i = (f(i), g(i))$ for $1 \leq i \leq t-1$ wish to compute s. By pooling their shares and forming a coalition they can construct the following system:

$$\begin{aligned}
&f(1) = A(1) + A(2) + \cdots + A(t)\\
&g(1) = \alpha + A(t+1) + \cdots + A(2t-1)\\
&f(2) = A(1) + 2 \cdot A(2) + \cdots + 2^{t-1} \cdot A(t)\\
&g(2) = \alpha + 2 \cdot A(t+1) + \cdots + 2^{t-1} \cdot A(2t-1)\\
&\vdots\\
&f(t-1) = A(1) + (t-1) \cdot A(2) + \cdots + (t-1)^{t-1} \cdot A(t)\\
&g(t-1) = \alpha + (t-1) \cdot A(t+1) + \cdots + (t-1)^{t-1} \cdot A(2t-1)
\end{aligned}$$

For any value i it is known that $A(i) = s + i \cdot Q_1 + \cdots + i^{2t-2} \cdot Q_{2t-2}$. So the above system of equations can therefore be rewritten purely in terms of the coefficients of $A(x)$ (including s). This results in an unsolvable system composed of $2t-2$ independent equations and $2t-2$ unknowns. To clarify this, let $s = Q_0$ then, consequently, α becomes:

$$Q_0 \cdot (A(1))^{-1} = Q_0 (\sum_{i=0}^{2t-2} Q_i)^{-1}$$

Therefore, the shares of a given participant in the coalition, denoted as P_w for $1 \leq w \leq t-1$, can be rewritten as:

$$f(w) = \sum_{j=1}^{t} \left(w^{j-1} \cdot \sum_{i=0}^{2t-2} j^i Q_i \right)$$

$$g(w) = Q_0(\sum_{z=i}^{2t-2} Q_i)^{-1} + \sum_{j=1}^{t} \left(w^{j-1} \cdot \sum_{i=0}^{2t-2} (j+t)^i Q_i\right)$$

Although both of these equations share the common factors of w, $Q_0, Q_1, \cdots, Q_{2t-2}$ it is evident that they are linearly independent of each other due to the different set of evaluation points used (the j values). This same principle applies to the shares held by each of the other participants. That is, the set of shares held by each of the participants in the coalition are linearly independent of each other due to the evaluation points used. In this case these evaluation points are the w values.

From this we can state that each of the $t-1$ participant is essentially given 2 shares of a degree $2t-2$ polynomial, albeit in roundabout fashion. Therefore, the coalition has a set of $2t-2$ shares relating to a polynomial of degree $2t-2$. Due to the perfectness of Shamir's secret sharing scheme [15] they cannot compute any information relating to s as (from the point of view of the $t-1$ participants) all potential q amount of values of s are equally likely.

Theorem 2. *The proposed* ***OKS-secure scheme*** *is a* (t, n, δ) *SSCD under the OKS model with share size* $|V_i| = q^2$*, secret domain* $\mathcal{S} = \mathbb{F}_q$ *and probability of cheating* $\delta = \frac{1}{q}$*.*

Proof. In order to successfully cheat, the set of $t-1$ dishonest participants, denoted by θ, will need to not only force the reconstruction of a secret $s' = s+\beta_1$ but also ensure that $s' = f'(0) \cdot g'(0)$ where $f'(0) = f(0)+\beta_2$ and $g'(0) = g(0)+\beta_3$ such that β_1 , β_2 , and β_3 are some constants. Using the Tompa and Woll attack it is possible for θ to actually pick the values of β_1, β_2 and β_3 and in order to cheat they need to pick these values such that:

$$s + \beta_1 = (f(0) + \beta_2)(g(0) + \beta_3)$$

Knowing that $s = f(0) \cdot g(0)$ this equation then becomes:

$$\beta_1 = \beta_3 \cdot f(0) + \beta_2 \cdot g(0) + \beta_2 \cdot \beta_3$$

In Shamir's scheme unless $t+1$ points are known no information about the secret can be obtained. Therefore both $f(0)$ and $g(0)$ are essentially random (from θ's point of view) as $f(0) = A(1)$ and $g(0) = \frac{s}{f(0)}$. Thus the probability of cheating is $\frac{1}{q}$ as the multiplication of two numbers in the field $\mathbb{F}_q$ can result in any of the q possibilities. Since each participant gets for their share two points each drawn from $\mathbb{F}_q$ and the secret is also drawn from $\mathbb{F}_q$ then the proposed scheme is a (t, n, δ) SSCD under the OKS model with $|V_i| = q^2$, $|\mathcal{S}| = q$ and $\delta = \frac{1}{q}$.

4.2 Proposed CDV-Secure Scheme

In our **OKS-secure scheme** it is easy to see that if the cheaters know the value of s then it is entirely possible for them to compute $\beta_1 = \beta_3 \cdot f(0) + \beta_2 \cdot g(0) + \beta_2 \cdot \beta_3$

and therefore break the security of the system. In fact if the cheaters know S then they can actually compute the share of the honest participant, as they will be able to solve the system given in the proof of Theorem 1.

So, while this scheme is certainly secure in the OKS setting (where the cheaters do not know the secret) it is not at all secure in the CDV setting (where they do know the secret). However, it is possible to rectify this with a simple modification.

The construction and cheating detection mechanism for the **CDV-secure scheme** is much the same as for the **OKS-secure scheme**. However, here $A(x)$ is of degree at most $3t-3$ and we use the first $3t-2$ points on $A(x)$ to construct three polynomials $f(x)$, $g(x)$ and $h(x)$ of degree at most $t-1$. This means that each participant obtains three shares, one relating to each of the degree $t-1$ polynomials. As per the second scheme described in [10] this prevents malicious participants (who know the secret) from computing the honest participant's share. We define the check digit as $l = f(0) \cdot g(0) \cdot h(0)$. All computations are performed in $\mathbb{F}_q$ where q is a prime number such that $q > 3n$ and $|\mathcal{S}| \leq q$.

$\texttt{SHARE}(s) \rightarrow (V_1, \ldots, V_n)$: $\mathcal{D}$ picks a random polynomial $A(x) = s + Q_1 x + \cdots + Q_{3t-3}x^{3t-3}$. He then uses the first $3t-2$ points on $A(x)$ to construct three more polynomials:

$$f(x) = A(1) + A(2)x + \cdots + A(t)x^{t-1}$$

$$g(x) = \alpha + A(t+1)x + \cdots + A(2t-1)x^{t-1}$$

$$h(x) = \omega + A(2t)x + \cdots + A(3t-2)x^{3t-2}$$

Where α and ω are picked such that $s = A(1) \cdot \alpha \cdot \omega$ and the other coefficients of $f(x)$, $g(x)$ and $h(x)$ are all points on $A(x)$. Participant P_i is assigned the share $V_i = (f(i), g(i), h(i))$ for $1 \leq i \leq n$.

$\texttt{REC}(\gamma) \rightarrow s'$: A subset, γ, of t participants (of which up to $t-1$ can be cheaters) submit shares to $\mathcal{C}$ who reconstructs $f'(x)$, $g'(x)$ and $h'(x)$ using Lagrange interpolation. $\mathcal{C}$ then uses the coefficients of $f'(x)$, $g'(x)$ and $h'(x)$ (specifically the points $A'(1), \cdots, A'(3t-2)$) to reconstruct $A'(x)$, again using Lagrange interpolation. If $A'(0) = f'(0) \cdot g'(0) \cdot h'(0)$ then $\mathcal{C}$ outputs $A'(0)$ as the secret, otherwise output $\perp$.

Security. As before, in this section the security of the given scheme is proved.

Theorem 3. *The proposed **CDV-secure scheme** is perfectly secure.*

Proof. Analogous to the proof of Theorem 1.

Theorem 4. *The proposed **CDV-secure scheme** is a (t, n, δ) SSCD under the CDV model with share size $|V_i| = q^3$, secret domain size $|\mathcal{S}| \leq q$ and probability of cheating $\delta = \frac{1}{q}$.*

Proof. As with the proof for Theorem 2 assume that we have t participants who wish to reconstruct s and that $t-1$ of these participants make up the set of dishonest participants denoted by θ. All computations are done in the field $\mathbb{F}_q$.

In order to successfully cheat, θ need to force reconstruction of $s' = f'(0) \cdot g'(0) \cdot h'(0)$. Where we define $s' = s + \beta_1$, $f'(0) = f(0) + \beta_2$, $g'(0) = g(0) + \beta_3$ and $h'(0) = h(0) + \beta_4$. Since θ know the value of s they can easily compute s'. So to successfully cheat they must solve the following equation:

$$s' = (f(0) + \beta_2)(g(0) + \beta_3)(h(0) + \beta_4)$$

In order to solve this equation they must first compute $f(0)$, $g(0)$ and $h(0)$. We show that this is not possible except with probability $\frac{1}{q}$ which is analogous to guessing. Without loss of generality assume that θ is composed of participants $P_1, \cdots, P_{t-1}$ and P_t is the honest participant. By pooling their shares, θ can construct the following system:

$$\begin{aligned}
f(1) &= A(1) + A(2) + \cdots + A(t) \\
g(1) &= \alpha + A(t+1) + \cdots + A(2t-1) \\
h(1) &= \omega + A(2t) + \cdots + A(3t-2) \\
f(2) &= A(1) + 2 \cdot A(2) + \cdots + 2^{t-1} \cdot A(t) \\
g(2) &= \alpha + 2 \cdot A(t+1) + \cdots + 2^{t-1} \cdot A(2t-1) \\
h(2) &= \omega + 2 \cdot A(2t) + \cdots + 2^{t-1} \cdot A(3t-2) \\
&\vdots \\
f(t-1) &= A(1) + (t-1) \cdot A(2) + \cdots + (t-1)^{t-1} \cdot A(t) \\
g(t-1) &= \alpha + (t-1) \cdot A(t+1) + \cdots + (t-1)^{t-1} \cdot A(2t-1) \\
h(t-1) &= \omega + (t-1) \cdot A(2t) + \cdots + (t-1)^{t-1} \cdot A(3t-2)
\end{aligned}$$

As with the proof of Theorem 1, the above system can be represented in terms of the coefficients of $A(x) = s + Q_1 x + \cdots + Q_{3t-3} x^{3t-3}$ and ω (as $\alpha = \frac{s}{\omega \cdot A(1)}$). Since s is known, the system is composed of $3t-3$ equations and $3t-2$ unknowns, meaning that θ cannot compute the values of $f(0)$, $g(0)$ or $h(0)$ except with probability $\frac{1}{q}$, i.e., they attempt to guess a value for either of the three variables. Therefore we can define the probability of successful cheating as $\delta = \frac{1}{q}$.

Since each participant receives three values as their share and all values (except for $s \in \mathcal{S}$) are drawn from $\mathbb{F}_q$ then the size of the share size is $|V_i| = q^3$. Thus the proposed scheme is a (t, n, δ) SSCD under the CDV model.

5 Conclusion

The results presented in this paper are threefold:

1. It was shown that the two SSCD schemes presented in [10] are vulnerable to the classic attack described by Tompa and Woll [16].

2. The construction of a near optimal SSCD scheme secure under the OKS model that supports arbitrary finite fields and a small secret domain was given. This is the most efficient of such schemes to date, being the same size as the original scheme devised in [10], which they state is 2 bits greater than optimal. Thus it is 1 bit more efficient than the scheme given in [12] which is the only other secure scheme that achieves these extended capabilities.
3. A near optimal SSCD scheme secure in the CDV model that has the same share size as the insecure scheme described by JS [10] was given. This is an extremely efficient scheme as it achieves a share size equal to that of the scheme presented by Cabello et al. [5] which is the most efficient yet.

An open problem within this field is the construction of a SSCD scheme secure under the CDV model that achieves optimum share size. Another interesting topic recently introduced in [12] is the construction of a SSCD scheme secure in the OKS model that is not only optimal but also supports an arbitrary field and a small secret domain.

References

1. Araki, T., Obana, S.: Flaws in some secret sharing schemes against cheating. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 122–132. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73458-1_10
2. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC 1988, pp. 1–10. ACM, New York (1988)
3. Benaloh, J.C.: Secret sharing homomorphisms: keeping shares of a secret secret (extended abstract). In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 251–260. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_19
4. Blakley, G.: Safeguarding cryptographic keys. In: Proceedings of the 1979 AFIPS National Computer Conference, pp. 313–317. AFIPS Press, Monval (1979)
5. Cabello, S., Padró, C., Sáez, G.: Secret sharing schemes with detection of cheaters for a general access structure. Des. Codes Crypt. **25**(2), 175–188 (2002)
6. Carpentieri, M., De Santis, A., Vaccaro, U.: Size of shares and probability of cheating in threshold schemes. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 118–125. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_10
7. Cevallos, A., Fehr, S., Ostrovsky, R., Rabani, Y.: Unconditionally-secure robust secret sharing with compact shares. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 195–208. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_13
8. 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. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 471–488. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_27

9. Hoshino, H., Obana, S.: Almost optimum secret sharing schemes with cheating detection for random bit strings. In: Tanaka, K., Suga, Y. (eds.) IWSEC 2015. LNCS, vol. 9241, pp. 213–222. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22425-1_13
10. Jhanwar, M.P., Safavi-Naini, R.: Almost optimum secret sharing with cheating detection. In: Chakraborty, R.S., Schwabe, P., Solworth, J. (eds.) SPACE 2015. LNCS, vol. 9354, pp. 359–372. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24126-5_21
11. 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). https://doi.org/10.1007/978-3-642-10230-1_8
12. Obana, S., Tsuchida, K.: Cheating detectable secret sharing schemes supporting an arbitrary finite field. In: Yoshida, M., Mouri, K. (eds.) IWSEC 2014. LNCS, vol. 8639, pp. 88–97. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09843-2_7
13. Ogata, W., Kurosawa, K.: Optimum secret sharing scheme secure against cheating. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 200–211. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_18
14. Ogata, W., Kurosawa, K., Stinson, D.R.: Optimum secret sharing scheme secure against cheating. SIAM J. Discret. Math. **20**(1), 79–95 (2006)
15. Shamir, A.: How to Share a Secret. Commun. ACM **22**(11), 612–613 (1979)
16. Tompa, M., Woll, H.: How to share a secret with cheaters. J. Crypt. **1**(2), 133–138 (1988)

XOR-Based Hierarchical Secret Sharing Scheme

Koji Shima(✉) and Hiroshi Doi

Institute of Information Security, Yokohama, Kanagawa, Japan
{dgs164101,doi}@iisec.ac.jp

Abstract. Hierarchical secret sharing schemes are known for how they share a secret among a group of participants partitioned into levels. In this study, we apply the general concept of hierarchy to Kurihara et al.'s XOR-based secret sharing scheme and propose a hierarchical secret sharing scheme applicable at any level. In our scheme, a minimal number of higher-level participants are required for recovery of the secret. Such hierarchical schemes applicable at any level over finite fields have been proposed, but no XOR-based hierarchical schemes applicable at any level have been reported. We realize the first such scheme. Moreover, considering practical use, we present an evaluation of our software implementation.

Keywords: Secret sharing scheme · Hierarchical access structure
XOR-based scheme · Ideal scheme · Software implementation

1 Introduction

In the modern information society, there is a strong need to securely store large amounts of secret information to prevent information theft or leakage and avoid information loss. Secret sharing schemes are known to simultaneously satisfy the need to distribute and manage secret information to prevent such information theft and loss. [1] and [2] independently introduced the basic idea of a (k, n) threshold secret sharing scheme almost four decades ago in 1979. In Shamir's (k, n) threshold scheme, n shares are generated from the secret, and each of these shares is distributed to a participant. Next, the secret can be recovered using any subset k of the n shares, but it cannot be recovered with fewer than k shares. Furthermore, every subset comprising less than k participants cannot obtain any information regarding the secret. Therefore, the original secret is secure even if some of the shares are leaked or exposed. Conversely, the secret can be recovered even if a few of the shares are missing.

Several hierarchical secret sharing schemes are known for how they share the given secret among a group of participants who are partitioned into levels. In such schemes, often, a minimal number of higher-level participants are required to recover the secret. For example, opening a bank vault may require, say, three employees, at least one of whom must be a department manager. In this scenario,

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 206–223, 2018.
https://doi.org/10.1007/978-3-319-97916-8_14

we have what is called a $(\{1,3\},n)$ hierarchical secret sharing scheme. In [3,4], Tassa introduced polynomial derivatives to generate shares and focused on the question related to Birkhoff interpolation problems.

Given that the hierarchical scheme described above requires indispensable participants to recover a given secret, from another perspective, this scheme can be used to delete the secret. In (k,n) threshold schemes, the reliability of data deletion depends on the deletion of more than $n-k$ shares. However, while using such a hierarchical scheme, the reliability of data deletion depends on the deletion of the specific shares held by the indispensable participants, that is, the deletion of more than $|\mathcal{U}_0|-k_0$ shares at the highest level when we use Definition 1. Considering practical and strategic use, for example, in cases where data deletion must be done with urgency, deletion of the secret is guaranteed by the deletion of the shares possessed by the indispensable participants. In other words, this scheme satisfies the need to prevent both information theft and information loss. Moreover, it offers the advantage of facilitating quick and straightforward deletion of the secret after it has been distributed.

1.1 Secret Sharing Schemes and Hierarchical Schemes

Shamir's (k,n) threshold secret sharing scheme can be implemented using arbitrary values of k and n, with $k \leq n$. However, the scheme requires extensive calculations for generating the n shares and recovering the secret from k shares because in doing so, a polynomial of degree $k-1$ must be processed. Fujii et al. [5] proposed a fast $(2,n)$ threshold scheme that uses only XOR operations to distribute and recover the secret. Kurihara et al. [6–8] proposed $(3,n)$ and (k,n) threshold schemes that use only XOR operations. In [9], Kurihara et al. presented a faster technique for realizing field operations over GF(q) by using the construction mechanisms of Feng et al. [10] and Blömer et al. [11] for the matrix representation of finite fields. Chen et al. [12] proposed a (k,n) threshold scheme that constructs shares based on a systematic information dispersal algorithm (IDA). All abovementioned schemes are *ideal*.

Tassa [3,4] proposed a $(\mathbf{k},n)$ hierarchical secret sharing scheme in which a minimal number of higher-level participants are required for recovering the secret. Tassa's scheme is *ideal*. Tassa used the derivative of a polynomial to achieve hierarchy and recover the secret via Birkhoff interpolation. In [13], Selçuk et al. proposed a function called the truncated version to achieve the described hierarchy. This truncated version truncates the polynomial from to the lowest-order term depending on the level. In [21,22], Shima et al. proposed fast hierarchical secret sharing schemes.

In addition, Tassa [3,4] showed other hierarchical settings studied by other authors. Shamir [2] suggested accomplishing a hierarchical scheme by assigning capable participants a large number of shares. Here, levels are represented as subsets of participants, but the necessary number of participants for recovery is determined based on a weighted average of the thresholds associated with each of these levels. In other words, when any subset of lower-level participants is sufficiently large, only the lower-level participants are needed to recover the secret. Simmons [14] and Brickell [15] considered other hierarchical settings. However,

the necessary number of participants is the highest of the thresholds associated with the various levels. Therefore, their hierarchical settings are unsuitable for the scenario in which a minimal number of higher-level participants must be involved in recovery of the secret.

Tassa then defined a $(\mathbf{k}, n)$ hierarchical secret sharing scheme as follows.

Definition 1. *Let* $\mathbf{k} = \{k_i\}_{i=0}^{m}$, $0 < k_0 < \cdots < k_m$, *and let* $k = k_m (\geq 2)$ *be the maximal threshold. A* $(\mathbf{k}, n)$ *hierarchical secret sharing scheme where a minimal number of higher-level participants is required for any recovery of the secret is defined as the following access structure* Γ*:*

$$\Gamma = \left\{ \mathcal{V} \subset \mathcal{U} : \left| \mathcal{V} \cap \left(\bigcup_{j=0}^{i} \mathcal{U}_j \right) \right| \geq k_i, \forall i \in \{0, 1, \cdots, m\} \right\}.$$

Here, let $\mathcal{U}$ *be a set of* n *participants and assume that* $\mathcal{U}$ *is composed of levels, that is,* $\mathcal{U} = \bigcup_{i=0}^{m} \mathcal{U}_i$, *where* $\mathcal{U}_i \bigcap \mathcal{U}_j = \emptyset$ *for all* $0 \leq i < j \leq m$. *The scheme then generates each share of the participants* $u \in \mathcal{U}$ *to satisfy the access structure.*

Given $\mathbf{k} = \{1, 3\}$ as an example, we have a $(\{1, 3\}, n)$ hierarchical scheme that consists of two levels and requires at least one indispensable participant from $\mathcal{U}_0$ and three or more participants from $\mathcal{U}_0 \bigcup \mathcal{U}_1$ to recover the secret.

1.2 Example Scenarios of Hierarchical Schemes

Tassa presented the opening of a bank vault as an example scenario. In this scenario, a fast $(\{1, 3\}, n)$ hierarchical secret sharing scheme is required. Castiglione et al. [16] presented other scenarios; the project manager and team members can access a project workspace according to their levels of authority; nurses may access a subset of patients' clinical data, while a doctor can access all data.

Here, we present a file management system as an example scenario. We store the indispensable participant's share in local storage such as smartphones, and we store the remaining two shares in external storage such as USB mass storage and cloud storage. Only the owner of the smartphone can recover this data by using either of the two external storage devices, and data cannot be recovered using only the two external storage devices. Therefore, there is a need for a fast $(\{1, 2\}, 3)$ hierarchical secret sharing scheme.

1.3 Our Contribution

In this paper, we propose an XOR-based *ideal* hierarchical secret sharing scheme, which can be applied to any level. Our scheme then requires a minimal number of higher-level participants to recover the secret. In addition, when k participants at the highest level $\mathcal{U}_0$ cooperate to recover the secret, the proposed scheme can yield the same result as Kurihara et al.'s (k, n) threshold scheme [7].

Tassa [3,4] proposed a hierarchical scheme over a finite field of sufficiently large prime order q. Kurihara et al. proposed an XOR-based non-hierarchical scheme. Shima et al. [22] proposed a hierarchical scheme over finite fields of characteristic two. No XOR-based hierarchical schemes applicable at any level have been reported. Our overall contributions are summarized as follows:

– We apply the general concept of hierarchy to Kurihara et al.'s XOR-based secret sharing scheme and realize a simple XOR-based hierarchical secret sharing scheme applicable to any level. We achieve hierarchy by using our function $F_W(i, j)$, shown later in Table 3, and providing mathematical proof.
– Our scheme eliminates an issue typically associated with such hierarchical schemes [3,4,22]; depending on the allocation of participant identities, there is a specific case in which the secret cannot be recovered in spite of an authorized subset.

2 Preliminaries

2.1 Notations and Definitions

– $\oplus$ denotes a bitwise XOR operation.
– $||$ denotes a concatenation of bit sequences.
– $n \in \mathbb{N}$ denotes number of participants.
– n_p is a prime number such that $n_p \geq n$.
– The following index values are elements of $\mathrm{GF}(n_p)$, that is, $X_{c(a\pm b)}$ denotes $X_{c(a\pm b) \bmod n_p}$. (1) random numbers, (2) divided pieces of the secret and the shares, (3) XOR-ed terms of (1) and (2), (4) participants, and (5) matrices.
– $H(X)$ denotes the Shannon entropy of a random variable X.

2.2 Definitions and Properties of Matrix

– $\mathbf{I}_n$ denotes the $n \times n$ identity matrix. $\boldsymbol{O}$ denotes a zero matrix.
– $\mathbf{1}$ denotes a matrix in which all elements are ones.
– Elementary row operations are as follows: (1) Each element in a row is multiplied by a non-zero constant, (2) A row is replaced by the sum of that row and a multiple of another row, (3) A row within the matrix is switched with another row.
– The leading coefficient of a row is the first nonzero element in that row.
– A matrix in row echelon form is one in which the leading coefficient of a nonzero row is strictly to the right of the leading coefficient of the row above, and all nonzero rows are above any rows of all zeros.
– Any matrix can be transformed to the row echelon form by using elementary row operations. The echelon form is not unique, but all row echelon forms have the same number of zero rows.
– The rank of matrix $\mathbf{A}$, $\mathrm{rank}(\mathbf{A})$, equals the number of nonzero rows of a matrix transformed to the row echelon form.

2.3 Perfect and Ideal Secret Sharing Schemes

In this subsection, we refer to Beimel [17] for a *perfect* secret sharing scheme and refer to Blundo et al. [18,19] and Kurihara et al. [6–8] for an *ideal* secret sharing scheme. Let S be a random variable in a given probability distribution on the secret, S_B be a random variable in a given probability distribution on the shares

in each authorized set B, and S_T be a random variable in a given probability distribution on the shares of each unauthorized set T. A *perfect* secret sharing scheme would satisfy the following conditions:

Correctness, Accessibility. $H(S|S_B) = 0$.
Perfect privacy, Perfect security. $H(S|S_T) = H(S)$.

A secret sharing scheme is called *ideal* if it is *perfect* and the information rate equals one. In other words, if the size of every bit of the shares equals the bit size of the secret, the scheme is *ideal*. As Tassa [4] mentioned in Definition 1.1, we may apply the information rate to a hierarchical secret sharing scheme.

3 Related Work

From Kurihara et al.'s scheme [7], we equally divide the secret $s \in \{0,1\}^{d(n_p-1)}$ into $n_p - 1$ blocks $s_1, \cdots, s_{n_p-1} \in \{0,1\}^d$. d is, for example, 8 and 64. The dealer securely distributes each share w_i for $i = 0, \cdots, n-1$ of a (k, n_p) threshold scheme to participant P_i. Therefore, we can construct a (k, n) threshold scheme by using a (k, n_p) threshold scheme even if n is a composite number.

Table 1 shows the distribution algorithm. Step 1 divides the secret equally into $n_p - 1$ pieces of the d-bit sequence. Step 2 generates $(k-1)n_p - 1$ pieces of the d-bit random number. Here, we define the vectors $\mathbf{s}, \mathbf{r}$ and $\mathbf{e}$, as given in Table 1. Step 3 generates shares w_i by using the function $F_W(i,j)$. We can view Step F1 as $w_{(i,j)} = \mathbf{v}_{(i,j)} \cdot \mathbf{e}$ with a uniquely given vector $\mathbf{v}_{(i,j)}$ corresponding to the (k, n_p) threshold scheme. If $k = 3$ and $n_p = 5$, $\mathbf{v}_{(2,1)} = (0100\ 00010\ 0001)$.

Table 1. Kurihara et al.'s distribution algorithm

Input: $s \in \{0,1\}^{d(n_p-1)}$
Output: $(w_0, \cdots w_{n-1})$

1: $s_0 \leftarrow \{0\}^d, s_1 || \cdots || s_{n_p-1} \leftarrow s$
2: for $i \leftarrow 0$ to $k-2$:
  for $j \leftarrow 0$ to $n_p - 1$:
    $r_j^i \xleftarrow{\$} \{0,1\}^d$ (discard $r_{n_p-1}^0$)
3: for $i \leftarrow 0$ to $n-1$:
  for $j \leftarrow 0$ to $n_p - 2$:
    $w_{(i,j)} \leftarrow F_W(i,j)$
  $w_i \leftarrow w_{(i,0)} || \cdots || w_{(i,n_p-2)}$
4: return $(w_0, \cdots w_{n-1})$

$F_W(i,j)$
F1: $w_{(i,j)} \leftarrow \left(\bigoplus_{h=0}^{k-2} r_{h \cdot i+j}^h \right) \oplus s_{j-i}$
F2: return $w_{(i,j)}$

– $\mathbf{s}$ denotes the vector $(s_1, \cdots, s_{n_p-1})^{\mathrm{T}}$.
– $\mathbf{r}$ denotes the vector $(r_0^0, \cdots, r_{n_p-2}^0, r_0^1, \cdots, r_{n_p-1}^1, \cdots, r_0^{k-2}, \cdots, r_{n_p-1}^{k-2})^{\mathrm{T}}$.
– $\mathbf{e}$ denotes the $(kn_p - 2)$-dimensional vector $\begin{bmatrix} \mathbf{r} \\ \mathbf{s} \end{bmatrix}$.

– $\mathbf{v}_{(i,j)}$ denotes a (kn_p-2)-dimensional binary vector such that $w_{(i,j)} = \mathbf{v}_{(i,j)} \cdot \mathbf{e}$.

Table 2 shows the recovery algorithm. Participants P_i for $i = t_0, \cdots, t_{k-1}$ cooperate to recover the secret. Step 1 equally divides each share into d-bit pieces. Step 2 generates the $k(n_p-1)$-dimensional vector $\mathbf{w}$. In Step 3, we obtain the matrix $\mathbf{M}$ by using the function $F_{MAT}()$. Step 4 recovers $s_1, \cdots, s_{n_p-1}$ by calculating $\mathbf{M} \cdot \mathbf{w}$. In Step 5, we obtain the secret s by concatenating $s_1, \cdots, s_{n_p-1}$. In Step F1, we obtain the vectors $\mathbf{v}_{(t_i,j)}$ such that $w_{(t_i,j)} = \mathbf{v}_{(t_i,j)} \cdot \mathbf{e}$. In Step F2, we obtain the matrix $\mathbf{G}$. In Step F3, the matrix $\left[\mathbf{G}\ \mathbf{I}_{k(n_p-1)}\right]$ is transformed to the row echelon form $\left[\bar{\mathbf{G}}\ \mathbf{J}\right]$. Matrices $\bar{\mathbf{G}}$ and $\mathbf{J}$ correspond to the matrices transformed from $\mathbf{G}$ and $\mathbf{I}_{k(n_p-1)}$, respectively. We then view the matrix $\left[\bar{\mathbf{G}}\ \mathbf{J}\right]$ as a matrix divided into block matrices $\boldsymbol{O}, \mathbf{G}_0, \mathbf{G}_1, \mathbf{G}_2, \mathbf{J}_0$, and $\mathbf{J}_1$, as given in Table 2. Step F4 transforms the matrix $[\mathbf{G}_0\ \mathbf{J}_0]$ to the matrix $\left[\mathbf{I}_{n_p-1}\ \mathbf{M}\right]$. Step 5 outputs the matrix $\mathbf{M}$. Intuitively, the secret can be recovered in an authorized set because $\mathbf{G}$ has full rank and can be transformed to $\bar{\bar{\mathbf{G}}}$ to obtain $s_1, \cdots, s_{n_p-1}$.

Table 2. Kurihara et al.'s recovery algorithm

Input: $(w_{t_0}, \cdots, w_{t_{k-1}})$
Output: s

1: for $i \leftarrow 0$ to $k-1$:
  $w_{(t_i,0)} || \cdots || w_{(t_i,n_p-2)} \leftarrow w_{t_i}$
2: $\mathbf{w} \leftarrow (w_{(t_0,0)}, \cdots, w_{(t_0,n_p-2)}, \cdots, w_{(t_{k-1},0)}, \cdots, w_{(t_{k-1},n_p-2)})^{\mathrm{T}}$
3: $\mathbf{M} \leftarrow F_{MAT}(t_0, \cdots, t_{k-1})$
4: $(s_1, \cdots, s_{n_p-1})^{\mathrm{T}} \leftarrow \mathbf{M} \cdot \mathbf{w}$
5: $s \leftarrow s_1 || \cdots || s_{n_p-1}$
6: return s

$F_{MAT}(t_0, \cdots, t_{k-1})$

F1: for $i \leftarrow 0$ to $k-1$:
  for $j \leftarrow 0$ to n_p-2:
    $\mathbf{v}_{(t_i,j)} \leftarrow w_{(t_i,j)} = \mathbf{v}_{(t_i,j)} \cdot \mathbf{e}$
F2: $\mathbf{G} \leftarrow (\mathbf{v}_{(t_0,0)}, \cdots, \mathbf{v}_{(t_{k-1},n_p-2)})^{\mathrm{T}}$
F3: $\left[\begin{array}{cc:c} \mathbf{G}_2 & \mathbf{G}_1 & \mathbf{J}_1 \\ \boldsymbol{O} & \mathbf{G}_0 & \mathbf{J}_0 \end{array}\right] = \left[\bar{\mathbf{G}} \,\vdots\, \mathbf{J}\right] \leftarrow \left[\mathbf{G}\ \mathbf{I}_{k(n_p-1)}\right]$
F4: $\left[\begin{array}{cc:c} \mathbf{G}_2 & \mathbf{G}_1 & \mathbf{J}_1 \\ \boldsymbol{O} & \mathbf{I}_{n_p-1} & \mathbf{M} \end{array}\right] = \left[\bar{\bar{\mathbf{G}}} \,\vdots\, \bar{\bar{\mathbf{J}}}\right] \leftarrow \left[\bar{\mathbf{G}} \,\vdots\, \mathbf{J}\right]$
F5: return $\mathbf{M}$

– $\mathbf{w}$ denotes a $k(n_p-1)$-dimensional vector.
– $\mathbf{M}$ denotes a $(n_p-1) \times k(n_p-1)$ binary matrix.
– $\mathbf{G}$ denotes a $k(n_p-1) \times (kn_p-2)$ binary matrix.

We analyze the proof techniques [7] to describe the correctness and perfect privacy of our proposed scheme and to actively use their lemmas.

Definition 2. *Let $t_0, \cdots, t_{L-1}$ denote the indexes of L shares, which are arbitrary numbers such that $0 \le t_i, t_j \le n-1$ and $t_i \ne t_j$ if $i \ne j$. Let $s_0 = \{0\}^d$ denote a singular point. Furthermore, let $\mathbf{r}$ be the same as shown in Table 1 and let $\mathbf{s}' = (s_0, \cdots, s_{n_p-1})^{\mathrm{T}}$. Then, we define the $L(n_p-1) \times (kn_p-1)$ binary generator matrix $\mathbf{G}' = [\mathbf{U}\ \mathbf{V}]$ such that*

$$\mathbf{w} = \mathbf{G}' \cdot \begin{bmatrix} \mathbf{r} \\ \mathbf{s}' \end{bmatrix} = \mathbf{U} \cdot \mathbf{r} \oplus \mathbf{V} \cdot \mathbf{s}'$$
$$= (w_{(t_0,0)}, \cdots, w_{(t_0,n_p-2)}, \cdots, w_{(t_{L-1},0)}, \cdots, w_{(t_{L-1},n_p-2)})^{\mathrm{T}}.$$

We may discuss correctness and perfect privacy with $\mathbf{s}'$ by using this singular point because $\mathbf{w}$ does not change. Furthermore, we may set as zero or one each

element of the column of $\mathbf{G}'$ corresponding to $s_0 = \{0\}^d$. Therefore, we can represent the matrix $\mathbf{V}$ with $(n_p - 1) \times n_p$ matrices $\mathbf{E}_{(j)}$.

Lemma 1 ([7] Lemma 1). *Under Definition 2, the conditions*

$$\begin{aligned} &rank(\mathbf{G}') = rank(\mathbf{U}) = L(n_p - 1) && \text{if } 1 \le L \le k-1, \\ &rank(\mathbf{G}') = k(n_p - 1),\ rank(\mathbf{U}) = (k-1)(n_p - 1) && \text{if } L \ge k \end{aligned}$$

are satisfied when matrices $\mathbf{G}'$ and $\mathbf{E}_{(j)}$ are determined, i.e.,

$$\mathbf{G}' = \left[\begin{array}{c|c} \mathbf{U} & \mathbf{V} \end{array}\right] = \left[\begin{array}{cccc|c} \mathbf{I}_{n_p-1} & \mathbf{E}_{(t_0)} & \cdots & \mathbf{E}_{((k-2)t_0)} & \mathbf{E}_{((n_p-1)t_0)} \\ \mathbf{I}_{n_p-1} & \mathbf{E}_{(t_1)} & \cdots & \mathbf{E}_{((k-2)t_1)} & \mathbf{E}_{((n_p-1)t_1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{I}_{n_p-1} & \mathbf{E}_{(t_{L-1})} & \cdots & \mathbf{E}_{((k-2)t_{L-1})} & \mathbf{E}_{((n_p-1)t_{L-1})} \end{array}\right], \mathbf{E}_{(j)} = \left[\begin{array}{c|c|c} & 0 & \\ \boldsymbol{O} & \vdots & \mathbf{I}_{n_p-j} \\ & 0 & \\ \hline & 0 & \\ \mathbf{I}_{j-1} & \vdots & \boldsymbol{O} \\ & 0 & \end{array}\right].$$

Here, we define matrix $\mathbf{E}^{(2)}_{(i,j)} = \mathbf{E}_{(i)} \oplus \mathbf{E}_{(j)}$. We then perform elementary row operations on matrix $\mathbf{G}'$ and obtain matrix $[\mathbf{U}'\ \mathbf{V}']$ represented by matrices

$$\mathbf{U}' = \left[\begin{array}{c|ccc} \mathbf{I}_{n_p-1} & \mathbf{E}_{(t_0)} & \cdots & \mathbf{E}_{((k-2)t_0)} \\ \hline \boldsymbol{O} & \mathbf{E}^{(2)}_{(t_0,t_1)} & \cdots & \mathbf{E}^{(2)}_{((k-2)t_0,(k-2)t_1)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \mathbf{E}^{(2)}_{(t_0,t_{L-1})} & \cdots & \mathbf{E}^{(2)}_{((k-2)t_0,(k-2)t_{L-1})} \end{array}\right] = \left[\begin{array}{c|c} \mathbf{I}_{n_p-1} & * \\ \hline \boldsymbol{O} & \\ \vdots & \mathbf{U}'' \\ \boldsymbol{O} & \end{array}\right],$$

$$\mathbf{V}' = \left[\begin{array}{c} \mathbf{E}_{((n_p-1)t_0)} \\ \hline \mathbf{E}^{(2)}_{((n_p-1)t_0,(n_p-1)t_1)} \\ \vdots \\ \mathbf{E}^{(2)}_{((n_p-1)t_0,(n_p-1)t_{L-1})} \end{array}\right] = \left[\begin{array}{c} \mathbf{E}_{((n_p-1)t_0)} \\ \hline \mathbf{V}'' \end{array}\right].$$

We also define matrix $\mathbf{D} = [\mathbf{U}''\ \mathbf{V}'']$, and Lemma 1 satisfies the conditions

$$\begin{aligned} &rank(\mathbf{D}) = rank(\mathbf{U}'') = (L-1)(n_p - 1) && \text{if } 1 \le L \le k-1, \\ &rank(\mathbf{D}) = (k-1)(n_p - 1),\ rank(\mathbf{U}'') = (k-2)(n_p - 1) && \text{if } L \ge k. \end{aligned}$$

Kurihara et al. introduced several other techniques in their proof of Lemma 1. Specifically, they introduced Lemmas 2 and 3, and defined matrix $\mathbf{H}_{(i,j)}$ by adding one row to $\mathbf{E}^{(2)}_{(i,j)}$ with all rows of $\mathbf{E}^{(2)}_{(i,j)}$.

Lemma 2 ([7] Lemma 4). *Suppose p is a prime number. Let $i, j, l \in \mathrm{GF}(p)$, $i \neq j$, and let $\mathbf{i}_l$ be a p-dimensional unit vector such that*

$$\mathbf{i}_l = (\overset{0}{\check{0}}, \cdots, 0, \overset{l}{\check{1}}, 0, \cdots, \overset{p-1}{\check{0}}).$$

Furthermore, let $\mathcal{X}$ be a set of p-dimensional binary vectors defined by

$$\mathcal{X} = \{\mathbf{i}_{i+m} \oplus \mathbf{i}_{j+m} \mid 0 \leq m \leq p-2\}.$$

Then, all vectors in $\mathcal{X}$ are linearly independent.

Lemma 3 ([7] Lemma 6). *Suppose $x_0, \cdots, x_{i-1}$ for $i \geq 2$ are random numbers selected from the finite set $\{0,1\}^h$ for $h > 0$ independently from each other with uniform probability $1/2^h$. Let $\mathcal{X}$ be the set defined by $\mathcal{X} = \{x_0, \cdots, x_{i-1}\}$. Then, $x_0, \cdots, x_{i-1}$ and all XOR-ed combinations of the elements in $\mathcal{X}$ are random numbers that are pairwise independent and distributed uniformly over $\{0,1\}^h$.*

Definition 3. *Let $i, j \in \mathrm{GF}(n_p), i \neq j$. We define the matrix $n_p \times n_p$ as follows:*

$$\mathbf{H}_{(i,j)} = \begin{bmatrix} \mathbf{E}^{(2)}_{(i,j)} \\ \bigoplus_{l=0}^{n_p-2} (\mathbf{i}_{i+l} \oplus \mathbf{i}_{j+l}) \end{bmatrix}.$$

Then, using Lemma 2 and $\mathbf{L}_i$, which denotes a circulant matrix based on the identity matrix, $\mathbf{H}_{(i,j)}$ is represented as

$$\mathbf{H}_{(i,j)} = \begin{bmatrix} \mathbf{E}^{(2)}_{(i,j)} \\ \mathbf{i}_{i-1} \oplus \mathbf{i}_{j-1} \end{bmatrix} = \mathbf{L}_i \oplus \mathbf{L}_j, \quad \mathbf{L}_i = \begin{bmatrix} \mathbf{E}_{(i)} \\ \mathbf{i}_{i-1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{O} & \mathbf{I}_{n_p-i} \\ \mathbf{I}_i & \boldsymbol{O} \end{bmatrix}.$$

Since $\mathrm{rank}(\mathbf{E}^{(2)}_{(i,j)}) = n_p - 1$ from Lemma 2, $\mathrm{rank}(\mathbf{H}_{(i,j)}) = n_p - 1$. Furthermore, we can represent matrix $\mathbf{D} = [\mathbf{U}'' \ \mathbf{V}'']$ as matrix $\mathbf{M} = [\mathbf{P} \ \mathbf{Q}]$, where

$$\mathbf{P} = \begin{bmatrix} \mathbf{H}_{(t_0,t_1)} & \cdots & \mathbf{H}_{((k-2)t_0,(k-2)t_1)} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{(t_0,t_{L-1})} & \cdots & \mathbf{H}_{((k-2)t_0,(k-2)t_{L-1})} \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} \mathbf{H}_{((n_p-1)t_0,(n_p-1)t_1)} \\ \vdots \\ \mathbf{H}_{((n_p-1)t_0,(n_p-1)t_{L-1})} \end{bmatrix}.$$

As a result, $\mathrm{rank}(\mathbf{D}) = rank(\mathbf{M})$, $\mathrm{rank}(\mathbf{U}'') = rank(\mathbf{P})$, and $\mathrm{rank}(\mathbf{V}'') = rank(\mathbf{Q})$ are satisfied. Then, we can simply multiply $\mathbf{H}_{(i,j)}$ required in elementary row operations because $\mathbf{L}_i \cdot \mathbf{L}_j = \mathbf{L}_j \cdot \mathbf{L}_i = \mathbf{L}_{i+j}$ is satisfied.

4 Proposed Scheme

In this section, we describe the proposed $(\mathbf{k}, n)$ hierarchical secret sharing scheme that satisfies Definition 1. Our scheme requires a minimal number of higher-level participants to recover the secret. From another perspective, not only k_0 participants at the highest level but also k_0+1 or more participants at the highest level can participate in the recovery. Therefore, we construct the scheme such that it can lead to the same result as Kurihara et al.'s (k, n) threshold scheme [7] when k participants at the highest level cooperate to recover the secret.

4.1 Distribution Algorithm

We equally divide secret $s \in \{0,1\}^{d(n_p-1)}$ into $n_p - 1$ blocks $s_1, \cdots, s_{n_p-1} \in \{0,1\}^d$. d is, for example, 8 and 64. The dealer securely distributes each share w_i for $i = 0, \cdots, n-1$ of a $(\mathbf{k}, n_p)$ hierarchical scheme to participant P_i. Therefore, we can construct a $(\mathbf{k}, n)$ hierarchical scheme with a $(\mathbf{k}, n_p)$ hierarchical scheme even if n is a composite number. Table 3 shows the difference between the proposed and Kurihara et al.'s algorithms.

Table 3. $F_W(i,j)$ of proposed scheme

$R(x) := \left(\bigoplus_{h=0}^{x} r_{h\cdot i+j}^h\right) \oplus \begin{cases} \text{if } x = k-2 : 0 \\ \text{else :} \quad \bigoplus_{h=x+1}^{k-2} \bigoplus_{a=0}^{n_p-1} r_a^h \end{cases}$
F1: $w_{(i,j)} \leftarrow \begin{cases} \text{if } l = 0 : R(k-2) \oplus s_{j-i} \\ \text{else :} \quad R(k-1-k_{l-1}) \oplus \left(\bigoplus_{a=1}^{n_p-1} s_a\right) \end{cases}$
F2: return $w_{(i,j)}$

– Participant $P_i \in \mathcal{U}_l$ receives $w_{(i,j)}$ and k_l is defined by Definition 1.

4.2 Recovery Algorithm

The difference of $F_W(i,j)$ leads to corresponding changes in $\mathbf{G}$, $\mathbf{G}'$, and $\mathbf{v}_{(i,j)}$, but our recovery algorithm is identical to that of Kurihara et al., as shown in Table 2. We can then view Step 3 of Table 2 as a precomputation.

4.3 Achieving Hierarchy

This creative solution of $F_W(i,j)$ in the proposed algorithm achieves hierarchy and is nontrivial, while the difference appears to be minor. Tassa's scheme [3,4] achieves hierarchy by using the properties of polynomial derivatives; shares of lower-level participants are not generated from the secret and some random values. Because Step F1 in Table 3 means $w_{(i,j)} = \mathbf{v}_{(i,j)} \cdot \mathbf{e}$, we have attempted to set all elements corresponding to the secret and some random values of $\mathbf{v}_{(i,j)}$ to zero for the lower-level participants, but this approach did not work. Through repeated consideration, we can achieve hierarchy when those elements are all ones and prove the correctness and perfect privacy of our scheme by actively using Kurihara et al.'s proof techniques. Intuitively, $\mathbf{G}$ used in recovery has full rank in an authorized set and does not have full rank in an unauthorized set, also including k or more participants. Section 4.5 shows an example of our scheme.

Achieving hierarchy with the proposed XOR-based scheme eliminates a traditional issue. As Tassa stated in [3,4], depending on the allocation of participant identities, there is a specific case in which the secret cannot be recovered in spite of the presence of an authorized subset because the determinant of a matrix used in recovery, related to the generator matrix, is zero under a certain probability. Our scheme does not require values of participant identities in the generator matrix. Therefore, this allocation issue does not occur, and every authorized set always receives the secret information.

4.4 Security Analysis

Based on Kurihara et al.'s proof approach [7,20], we present the effects on our $F_W(i,j)$ in Table 3. We introduce Lemmas 4 and 5.

Without loss of generality, we may view matrix $\mathbf{G}'$, corresponding to participants who cooperate to recover the secret, as a matrix listed from to the highest-level participants in the recovery set. In other words, the right block matrices of the first $\mathbf{1}$ in each row of $\mathbf{G}'$ are all $\mathbf{1}$'s and the block matrices below those $\mathbf{1}$'s in that row are also all $\mathbf{1}$'s. We then define matrix $\hat{\mathbf{E}}^{(2)}_{(i)} = \mathbf{E}_{(i)} \oplus \mathbf{1}$ and the $n_p \times n_p$ matrix $\hat{\mathbf{H}}_{(i)}$ by adding one row to $\hat{\mathbf{E}}^{(2)}_{(i)}$ along with all rows of $\hat{\mathbf{E}}^{(2)}_{(i)}$. $\hat{\mathbf{H}}_{(i)}$ can be represented as $\hat{\mathbf{H}}_{(i)} = \bigoplus_{j=0, j\neq i}^{n_p-1} \mathbf{L}_j$.

Lemma 4. *Under Definition 2, Let* $2 \leq L \leq k$. $\mathbf{M} = [\mathbf{P}\ \mathbf{Q}]$, *corresponding to* $\mathbf{D} = [\mathbf{U}''\ \mathbf{V}'']$, *is represented as* $\mathbf{M}^{(L)}_k = [\mathbf{M}_{(i,j)}]^{L-1\ k-1}_{i=1\ j=1}$. *We define*

$$\mathbf{H}^a_{(i,j)} = \begin{cases} \mathbf{H}_{(a\cdot t_i, a\cdot t_j)} & \text{if } 1 \leq a \leq k-2 \\ \mathbf{H}_{((n_p-1)t_i,(n_p-1)t_j)} & \text{if } a = k-1 \end{cases},$$

$$\hat{\mathbf{H}}^a_{(i)} = \begin{cases} \hat{\mathbf{H}}_{(a\cdot t_i)} & \text{if } 1 \leq a \leq k-2 \\ \hat{\mathbf{H}}_{((n_p-1)t_i)} & \text{if } a = k-1 \end{cases}.$$

Furthermore, we define q_i *for the* i*-th block row of* $\mathbf{M}^{(L)}_k$ *and view* q_0 *as* q_{L-1}. *Let* $a + q_a \leq b + q_b, a < b$ *for all* a, b *since we may view* $\mathbf{M}^{(L)}_k$ *as a matrix listed from to the highest-level participants, and let*

$$\mathbf{M}_{(i,j)} = \begin{cases} \mathbf{H}^j_{(0,i)} & (i + j + q_i < L) \\ \hat{\mathbf{H}}^j_{(0)} & (i + j - q_i \geq L) \end{cases}.$$

If $q_i \leq 0$ *for all* i, *then* $\mathrm{rank}(\mathbf{M}^{(L)}_k) = \mathrm{rank}(\mathbf{D}) = (L-1)(n_p-1)$, *else* $\mathrm{rank}(\mathbf{M}^{(L)}_k) = \mathrm{rank}(\mathbf{D}) < (L-1)(n_p-1)$.

Proof. For example, if $L = 3 < k, q_1 = 0$, and $q_2 = 0$, $\mathbf{M}^{(L)}_k$ is represented as

$$\mathbf{M}^{(L)}_k = \left.\begin{bmatrix} \mathbf{H}^1_{(0,k-2)} & \hat{\mathbf{H}}^2_{(0)} & \cdots & \hat{\mathbf{H}}^{k-2}_{(0)} & \hat{\mathbf{H}}^{k-1}_{(0)} \\ \hat{\mathbf{H}}^1_{(0)} & \hat{\mathbf{H}}^2_{(0)} & \cdots & \hat{\mathbf{H}}^{k-2}_{(0)} & \hat{\mathbf{H}}^{k-1}_{(0)} \end{bmatrix}\right\} L-1 \text{ blocks}$$

and this lemma proves $\mathrm{rank}(\mathbf{M}^{(L)}_k) = (L-1)(n_p-1)$. If $L = k$ and $q_i = 0$ for all i, $\mathbf{M}^{(L)}_k$ is represented as

$$\mathbf{M}^{(L)}_k = \left.\begin{bmatrix} \mathbf{H}^1_{(0,1)} & \mathbf{H}^2_{(0,1)} & \cdots & \mathbf{H}^{k-2}_{(0,1)} & \hat{\mathbf{H}}^{k-1}_{(0)} \\ \mathbf{H}^1_{(0,2)} & \mathbf{H}^2_{(0,2)} & \cdots & \hat{\mathbf{H}}^{k-2}_{(0)} & \hat{\mathbf{H}}^{k-1}_{(0)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{H}^1_{(0,k-2)} & \hat{\mathbf{H}}^2_{(0)} & \cdots & \hat{\mathbf{H}}^{k-2}_{(0)} & \hat{\mathbf{H}}^{k-1}_{(0)} \\ \hat{\mathbf{H}}^1_{(0)} & \hat{\mathbf{H}}^2_{(0)} & \cdots & \hat{\mathbf{H}}^{k-2}_{(0)} & \hat{\mathbf{H}}^{k-1}_{(0)} \end{bmatrix}\right\} L-1 \text{ blocks}$$

and this lemma also proves $\mathrm{rank}(\mathbf{M}_k^{(L)}) = (L-1)(n_p-1)$.

Let $i_0 = 0, \cdots, L-2$ and let $\bar{\mathbf{M}}_k^{(L)}$ be the matrix such that the j-th block column of $\mathbf{M}_k^{(L)}$ for $j = 1, \cdots, k-1$ is switched with the $(k-j)$-th block column and the i-th block row for $i = 1, \cdots, L-2$ is switched with the $(L-1)$-th block row. Since elementary matrix operations do not change the rank of a matrix, we may discuss $\mathrm{rank}(\bar{\mathbf{M}}_k^{(L)})$ instead of $\mathrm{rank}(\mathbf{M}_k^{(L)})$. If $q_{i_0} = 0$ for all i_0, $\bar{\mathbf{M}}_k^{(L)}$ is represented as

$$\bar{\mathbf{M}}_k^{(L)} = \left.\begin{bmatrix} \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(0)}^{2} & \hat{\mathbf{H}}_{(0)}^{1} \\ \hat{\mathbf{H}}_{(0)}^{k-1} & \mathbf{H}_{(0,1)}^{k-2} & \cdots & \mathbf{H}_{(0,1)}^{2} & \mathbf{H}_{(0,1)}^{1} \\ \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \cdots & \mathbf{H}_{(0,2)}^{2} & \mathbf{H}_{(0,2)}^{1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(0)}^{2} & \mathbf{H}_{(0,k-2)}^{1} \end{bmatrix} \right\} L-1 \text{ blocks} .$$

Since $\hat{\mathbf{H}}_{(i)}^{x} \oplus \mathbf{H}_{(i,j)}^{x} = \hat{\mathbf{H}}_{(j)}^{x}$, $\bar{\mathbf{M}}_k^{(L)}$ can be transformed to

$$\bar{\mathbf{M}}_k^{(L)} \to \begin{bmatrix} \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \hat{\mathbf{H}}_{(0)}^{k-3} & \cdots & \hat{\mathbf{H}}_{(0)}^{1} \\ \boldsymbol{O} & \hat{\mathbf{H}}_{(1)}^{k-2} & \hat{\mathbf{H}}_{(1)}^{k-3} & \cdots & \hat{\mathbf{H}}_{(1)}^{1} \\ \boldsymbol{O} & \boldsymbol{O} & \hat{\mathbf{H}}_{(2)}^{k-3} & \cdots & \hat{\mathbf{H}}_{(2)}^{1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \cdots & \hat{\mathbf{H}}_{(k-2)}^{1} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_0 & * & * \\ \boldsymbol{O} & & \\ \vdots & * & \mathbf{M}' \\ \boldsymbol{O} & & \end{bmatrix}$$

by using elementary row operations. Here, we define matrices $\mathbf{M}_0$ and $\mathbf{M}'$. Since $\hat{\mathbf{H}}_{(i_0)}^{k-1-i_0} \neq \boldsymbol{O}$ for all i_0, $\mathrm{rank}(\bar{\mathbf{M}}_k^{(L)}) = (L-1)(n_p-1)$. If $q_{i_0} \geq 1$ for at least any one of i_0, $\mathrm{rank}(\bar{\mathbf{M}}_k^{(L)}) < (L-1)(n_p-1)$ because at least one of the diagonal block matrices of $\bar{\mathbf{M}}_k^{(L)}$, corresponding to $\hat{\mathbf{H}}_{(i_0)}^{k-1-i_0}$ for all i_0, is $\boldsymbol{O}$. Note that if $L < k$, we can discuss the diagonal block matrices of $\mathbf{M}'$.

Next, we consider the remaining condition $q_{i_0} < 0$ for all i_0. For example, if $q_1 = -1, q_2 = -2$, and the others $q_{i_0} = 0$, $\bar{\mathbf{M}}_k^{(L)}$ can be transformed to

$$\bar{\mathbf{M}}_k^{(L)} \to \begin{bmatrix} \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \hat{\mathbf{H}}_{(0)}^{k-3} & \hat{\mathbf{H}}_{(0)}^{k-4} & \cdots & \hat{\mathbf{H}}_{(0)}^{1} \\ \hat{\mathbf{H}}_{(1)}^{k-1} & \hat{\mathbf{H}}_{(1)}^{k-2} & \hat{\mathbf{H}}_{(1)}^{k-3} & \hat{\mathbf{H}}_{(1)}^{k-4} & \cdots & \hat{\mathbf{H}}_{(1)}^{1} \\ \hat{\mathbf{H}}_{(2)}^{k-1} & \hat{\mathbf{H}}_{(2)}^{k-2} & \hat{\mathbf{H}}_{(2)}^{k-3} & \hat{\mathbf{H}}_{(2)}^{k-4} & \cdots & \hat{\mathbf{H}}_{(2)}^{1} \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \hat{\mathbf{H}}_{(3)}^{k-4} & \cdots & \hat{\mathbf{H}}_{(3)}^{1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \cdots & \hat{\mathbf{H}}_{(k-2)}^{1} \end{bmatrix} .$$

In general, $\bar{\mathbf{M}}_k^{(L)}$ can be transformed to

$$\bar{\mathbf{M}}_k^{(L)} \rightarrow \begin{bmatrix} \mathbf{M}_0^{(m_0)} & & & * \\ & \mathbf{M}_1^{(m_1)} & & \\ & & \ddots & \\ \boldsymbol{O} & & & \mathbf{M}_\alpha^{(m_\alpha)} \end{bmatrix},$$

where $\mathbf{M}_i^{(m_i)}$ is an $m_i \times m_i$ block matrix. If $\mathrm{rank}(\mathbf{M}_i^{(m_i)}) = m_i(n_p - 1)$ for all i, $\mathrm{rank}(\bar{\mathbf{M}}_k^{(L)}) = (L-1)(n_p-1)$. Without loss of generality, we can discuss $\mathbf{M}_0^{(m_0)}$ instead of all matrices $\mathbf{M}_i^{(m_i)}$. $\mathbf{M}_0^{(m_0)}$ can be transformed to

$$\begin{aligned} \mathbf{M}_0^{(m_0)} &= \begin{bmatrix} \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(0)}^{(k-m_0)} \\ \hat{\mathbf{H}}_{(1)}^{k-1} & \hat{\mathbf{H}}_{(1)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(1)}^{(k-m_0)} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{H}}_{(m_0-1)}^{k-1} & \hat{\mathbf{H}}_{(m_0-1)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(m_0-1)}^{(k-m_0)} \end{bmatrix} \\ &\rightarrow \left[\begin{array}{cccc} \hat{\mathbf{H}}_{(0)}^{k-1} & \hat{\mathbf{H}}_{(0)}^{k-2} & \cdots & \hat{\mathbf{H}}_{(0)}^{(k-m_0)} \\ \hline \mathbf{H}_{(0,1)}^{k-1} & \mathbf{H}_{(0,1)}^{k-2} & \cdots & \mathbf{H}_{(0,1)}^{(k-m_0)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{(0,m_0-1)}^{k-1} & \mathbf{H}_{(0,m_0-1)}^{k-2} & \cdots & \mathbf{H}_{(0,m_0-1)}^{(k-m_0)} \end{array}\right] = \begin{bmatrix} * \\ \hline \mathbf{H}' \end{bmatrix}. \end{aligned}$$

Here, let $\mathbf{M}$ be the matrix used in Kurihara et al.'s scheme and let $\bar{\mathbf{M}}$ be the matrix such that the j-th block column of $\mathbf{M}$ for $j = 1, \cdots, k-1$ is switched with the $(k-j)$-th block column. $\bar{\mathbf{M}}$ can be also transformed to the same matrix transformed from $\mathbf{M}$. Then, we can view $\mathbf{H}'$ as a submatrix of $\bar{\mathbf{M}}$. $\mathbf{H}'$ can be transformed to the row echelon form as with $\bar{\mathbf{M}}$ or $\mathbf{M}$. In other words, $\mathrm{rank}(\mathbf{H}') = (m_0-1)(n_p-1)$. Furthermore, $\mathrm{rank}(\mathbf{M}_0^{(m_0)}) = m_0(n_p-1)$ because each block row of $\mathbf{H}'$ is XOR-ed with the first block row of $\mathbf{M}_0^{(m_0)}$. Therefore, $\mathrm{rank}(\bar{\mathbf{M}}_k^{(L)}) = (L-1)(n_p-1)$ and the proof is complete. □

Lemma 5. *Under Definition 2, Let $2 \leq L$. If the block matrices in $\mathbf{V}$ are all $\mathbf{1}$'s,* $\mathrm{rank}(\mathbf{G}') \leq (L-1)(n_p-1)$. *Furthermore, if the block matrices in $\mathbf{V}$ and the $k-2, k-1, \cdots, j$-th block columns of $\mathbf{U}$ are all $\mathbf{1}$'s,* $\mathrm{rank}(\mathbf{G}') \leq (L-1)(n_p-1)$.

Proof. Matrices $\mathbf{V}$ and $\mathbf{V}'$ are represented as

$$\mathbf{V} = \left.\begin{bmatrix} \mathbf{1} \\ \vdots \\ \mathbf{1} \end{bmatrix}\right\} L \text{ blocks}, \quad \mathbf{V}' = \begin{bmatrix} \mathbf{1} \\ \boldsymbol{O} \\ \vdots \\ \boldsymbol{O} \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \hline \mathbf{V}'' \end{bmatrix}.$$

Here, we refer to Lemma 4. $\mathbf{M}$, corresponding to $\mathbf{D} = [\mathbf{U}''\ \mathbf{V}'']$, can be represented as $\mathbf{M}_k^{(L)}$ in which $\mathbf{M}_0$ is always $\boldsymbol{O}$ if $L \leq k$. It is apparent that

$\mathrm{rank}(\mathbf{M}) \leq (L-1)(n_p-1)$. Furthermore, even if $L > k$, we can retain $\mathrm{rank}(\mathbf{M}) \leq (L-1)(n_p-1)$. The proof is therefore complete. □

We introduce Theorems 1, 2, and 3 under the access structure of Definition 1. Theorem 1 shows correctness $H(S|S_B) = 0$. Theorems 2 and 3 show perfect privacy $H(S|S_T) = H(S)$.

Theorem 1. *Let $L = k$ in Definition 2. Assume that any set of k participants $B = \{P_{t_0}, \cdots, P_{t_{k-1}}\}$ agrees to recover the secret. Then, Lemma 1 holds even if matrix $\mathbf{1}$'s are included and the secret can be recovered.*

Proof. Consider the set of participants that can recover the secret. Because the corresponding $\mathbf{G}'$ has $\mathrm{rank}(\mathbf{G}') = k(n_p-1)$ and $\mathrm{rank}(\mathbf{U}) = (k-1)(n_p-1)$ from Lemma 4, the secret is therefore recovered. □

Theorem 2. *Let $1 \leq L \leq k-1$ in Definition 2. Assume that any set of L participants $T = \{P_{t_0}, \cdots, P_{t_{L-1}}\}$ agrees to recover the secret. Then, we receive no information regarding the secret.*

Proof. Consider $\mathbf{G}'$ corresponding to the set of participants. Let $\alpha = L'(n_p-1)$ be the rank of $\mathbf{G}'$. If $L = 1$, $\mathrm{rank}(\mathbf{G}') = L(n_p-1) = \alpha$. From Lemma 4, if $L = 2, \cdots, k-1$, $\mathrm{rank}(\mathbf{G}') = \alpha \leq L(n_p-1)$. In other words, we can find $\bar{\mathbf{G}}' = \begin{bmatrix}\bar{\mathbf{U}} & \bar{\mathbf{V}}\end{bmatrix}$ such that $\mathrm{rank}(\bar{\mathbf{G}}') = \alpha$ and $\mathrm{rank}(\bar{\mathbf{U}}) = \alpha$. If we receive no information regarding the secret from the $\bar{\mathbf{G}}'$, we do not receive information more than $\bar{\mathbf{G}}'$.

Suppose that elements of $\mathbf{s}$, as given in Table 1, and elements of $\mathbf{r}$ are mutually independent and that elements of $\mathbf{r}$ are selected from the finite set $\{0,1\}^d$ with uniform probability $1/2^d$. $\bar{\mathbf{U}}$ and $\bar{\mathbf{V}}$ are $L'(n_p-1) \times ((k-1)n_p-1)$ and $L'(n_p-1) \times n_p$ matrices, respectively. Because all rows of $\bar{\mathbf{U}}$ are linearly independent for $1 \leq L' \leq k-1$, we can use Lemma 3. In other words, all elements of the $L'(n_p-1)$-dimensional vector $\bar{\mathbf{U}} \cdot \mathbf{r}$ are d-bit random numbers that are mutually independent and distributed uniformly over $\{0,1\}^d$. Therefore, the vector $\bar{\mathbf{U}} \cdot \mathbf{r}$ is uniformly distributed over $\{0,1\}^{dL'(n_p-1)}$.

Next, we suppose that $\mathbf{w}'$ denotes a fixed value of $\mathbf{w}$. $\mathbf{w} = \mathbf{w}'$ can be obtained with uniform probability $(1/2)^{dL'(n_p-1)}$ from any selected $\mathbf{s}$ or $\bar{\mathbf{V}} \cdot \mathbf{s}'$. Because $\mathbf{s}$ is independent of $\mathbf{w}$, we have $H(S|S_T) = H(S)$ and therefore receive no information regarding the secret. □

In hierarchical schemes, we need to consider the case of k or more participants in an unauthorized set T.

Theorem 3. *Let $L \geq k$ in Definition 2. Assume that any set of L participants $T = \{P_{t_0}, \cdots, P_{t_{L-1}}\} \notin \Gamma$ agrees to recover the secret. Then, the secret cannot be recovered.*

Proof. From Lemma 5, when no participants at the highest level $\mathcal{U}_0$ cooperate to recover the secret, the secret cannot be recovered. Next, from Lemma 4, when a minimal number of higher-level participants are not included, $\mathrm{rank}(\mathbf{G}') < k(n_p-1)$. If $L > k$, each of the $L-k$ block rows should include $\hat{\mathbf{H}}^j_{(0)}$, shown in

Lemma 4, because the block row is not for the participant at the highest level $\mathcal{U}_0$. As a result, we can retain $\text{rank}(\mathbf{G}') < k(n_p - 1)$ even if $L > k$. Therefore, we can simplify the proof to the case of $L < k$ shown in Theorem 2 and receive no information regarding the secret. □

The proof is thus complete, and we conclude that both correctness and perfect privacy hold. Because every bit size of shares equals the bit size of the secret, our scheme is *ideal*.

4.5 Brief Example of Our Scheme

We consider a $(\{2,3\},5)$ hierarchical secret sharing scheme as an example. Let participants $P_0, P_1, P_2 \in \mathcal{U}_0$, and $P_3, P_4 \in \mathcal{U}_1$. We generate each share $w_i = w_{(i,0)}||\cdots||w_{(i,3)}$ for participant P_i with $\mathbf{w} = \mathbf{G} \cdot \begin{bmatrix} \mathbf{r} \\ \mathbf{s} \end{bmatrix}$. For example, we can view the $\mathbf{v}_{(0,0)}$ used to generate $w_{(0,0)}$ as $\mathbf{v}_{(0,0)} = (1000\ 10000\ 0000)$, i.e.,

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_0 \\ \mathbf{w}_1 \\ \mathbf{w}_2 \\ \mathbf{w}_3 \\ \mathbf{w}_4 \end{bmatrix} = \begin{bmatrix} w_{(0,0)} \\ w_{(0,1)} \\ w_{(0,2)} \\ w_{(0,3)} \\ w_{(1,0)} \\ w_{(1,1)} \\ \vdots \\ w_{(3,2)} \\ w_{(3,3)} \\ w_{(4,0)} \\ w_{(4,1)} \\ w_{(4,2)} \\ w_{(4,3)} \end{bmatrix} = \mathbf{G} \cdot \begin{bmatrix} \mathbf{r} \\ \mathbf{s} \end{bmatrix} = \begin{bmatrix} 1000\ 10000\ 0000 \\ 0100\ 01000\ 1000 \\ 0010\ 00100\ 0100 \\ 0001\ 00010\ 0010 \\ 1000\ 01000\ 0001 \\ 0100\ 00100\ 0000 \\ \vdots \\ 0010\ 11111\ 1111 \\ 0001\ 11111\ 1111 \\ 1000\ 11111\ 1111 \\ 0100\ 11111\ 1111 \\ 0010\ 11111\ 1111 \\ 0001\ 11111\ 1111 \end{bmatrix} \begin{bmatrix} r_0^0 \\ r_1^0 \\ r_2^0 \\ r_3^0 \\ r_0^1 \\ r_1^1 \\ r_2^1 \\ r_3^1 \\ r_4^1 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}.$$

When participants P_1, P_2, P_4 cooperate to recover the secret, the secret can be recovered because

$$[\mathbf{G}\ \mathbf{I}_{12}] = \begin{bmatrix} 1000\ 01000\ 0001 & 1000\ 0000\ 0000 \\ 0100\ 00100\ 0000 & 0100\ 0000\ 0000 \\ 0010\ 00010\ 1000 & 0010\ 0000\ 0000 \\ 0001\ 00001\ 0100 & 0001\ 0000\ 0000 \\ 1000\ 00100\ 0010 & 0000\ 1000\ 0000 \\ 0100\ 00010\ 0001 & 0000\ 0100\ 0000 \\ 0010\ 00001\ 0000 & 0000\ 0010\ 0000 \\ 0001\ 10000\ 1000 & 0000\ 0001\ 0000 \\ 1000\ 11111\ 1111 & 0000\ 0000\ 1000 \\ 0100\ 11111\ 1111 & 0000\ 0000\ 0100 \\ 0010\ 11111\ 1111 & 0000\ 0000\ 0010 \\ 0001\ 11111\ 1111 & 0000\ 0000\ 0001 \end{bmatrix},$$

$$[\mathbf{G}\ \mathbf{I}_{12}] \to [\bar{\mathbf{G}}\ \mathbf{J}] \to \left[\bar{\bar{\mathbf{G}}}\ \bar{\bar{\mathbf{J}}}\right] = \begin{bmatrix} 1000\ 01000\ 0001 & 1000\ 0000\ 0000 \\ 0100\ 00100\ 0000 & 0100\ 0000\ 0000 \\ 0010\ 00010\ 1000 & 0010\ 0000\ 0000 \\ 0001\ 00001\ 0100 & 0001\ 0000\ 0000 \\ 0000\ 11101\ 1111 & 1001\ 1001\ 0000 \\ 0000\ 01010\ 0010 & 1100\ 1100\ 0000 \\ 0000\ 00101\ 1001 & 0110\ 0110\ 0000 \\ 0000\ 00011\ 1000 & 0010\ 0010\ 0000 \\ 0000\ 00000\ \underline{1000} & \underline{1011\ 1001\ 0010} \\ 0000\ 00000\ \underline{0100} & \underline{0001\ 0010\ 0011} \\ 0000\ 00000\ \underline{0010} & \underline{0100\ 1000\ 1100} \\ 0000\ 00000\ \underline{0001} & \underline{1001\ 1101\ 0100} \end{bmatrix},$$

$$\mathbf{G}' = \begin{bmatrix} \mathbf{I}_4 & \mathbf{E}_{(1)} & \mathbf{E}_{(4)} \\ \mathbf{I}_4 & \mathbf{E}_{(2)} & \mathbf{E}_{(3)} \\ \mathbf{I}_4 & \mathbf{1} & \mathbf{1} \end{bmatrix}, \ \mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 1011 & 1001 & 0010 \\ 0001 & 0010 & 0011 \\ 0100 & 1000 & 1100 \\ 1001 & 1101 & 0100 \end{bmatrix} \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \mathbf{w}_4 \end{bmatrix}.$$

However, participants P_0, P_3, P_4 cannot recover the secret from Theorem 3.

5 Software Implementation

For the $(\mathbf{k}, n)$ hierarchical secret sharing scheme, we evaluated the proposed scheme using one general purpose machine, as described in Table 4. Furthermore, we used a file size of 888,710 bytes and provided some parameters of k and n.

Table 4. Test environment

CPU	Intel ® Celeron ® Processor G1820 2.70 GHz × 2, 2 MB cache
RAM	3.6 GB
OS	CentOS 7 Linux 3.10.0-229.20.1.el7.x86_64
Programing language	C
Compiler system	gcc 4.8.3 (-O3 -flto -DNDEBUG)

Table 5 presents our experimental results. We used $d = 64$ and used Xorshift for random number generation. The processing time to generate random numbers was included in the results of distribution. The implementation used in our experiments can be applied to any level, not specialized for, for example, $\mathbf{k} = \{1, 3\}$. We then assigned n participants such that participants in each level could cooperate to recover the secret and most of the remaining $n - k$ participants belonged to lower levels, especially the lowest level. We obtained almost the same results in both $(\{1, k\}, n)$ and $(\{2, 3, k\}, n)$ with $k = 5$ and $n = 11$.

Table 5. Experimental results

$(\mathbf{k}, n)$	Distribution (Mbps)	Recovery (Mbps)
({1,3},5)	1,390	6,750
({1,3},11)	130	1,720
({1,3},59)	4.07	206
({1,5},11)	62.6	1,070
({2,4},7)	321	2,340
({2,3,5},11)	62.5	1,090
({2,4,6,10},17)	11.5	195
({3,7,11,14,17},23)	3.52	34.7

Table 6 shows the computational costs. In general, the size of the secret exceeded $d(n_p - 1)$ bits. In our analysis, we refer to such an initial computation processed once for that recovery as a precomputation. We omit the detailed calculations here due to the page limitation, but we may view the cost of recovery in our scheme as the one in Kurihara et al.'s scheme [7,8] because no changes were made to them in terms of the number of necessary XOR operations. Furthermore, for example, the minimum computational cost of distribution can be obtained when all n participants belong to the highest level. In Table 6, we can see that our scheme is more efficient than Tassa's approach, which is a hierarchical scheme, if n_p is not extremely large. Note that the precomputation cost is not dominant in a large file, while our scheme is not more efficient than Tassa's approach. Furthermore, as Shima et al. [22] mentioned in Sect. 5.2.1, we can see that arithmetic operations required high computational costs.

Table 6. Computational costs

	Precomputation	Distribution	Recovery
Our scheme	$\mathcal{O}(k^3 n_p^3)$	$\mathcal{O}(kn)\|s\| \leq x \leq \mathcal{O}(kn_p n)$	$\mathcal{O}(kn_p)\|s\|$
Tassa's scheme	$\mathcal{O}(k^3)$	$\mathcal{O}(kn)$	$\mathcal{O}(k^3)$

- k is the maximal threshold, shown in Definition 1.
- $|s|$ denotes the bit length of the secret, i.e., $|s| = d(n_p - 1)$.
- A $t \times t$ determinant required $\mathcal{O}(t^3)$ when we used LU decomposition.
- Our scheme shows the costs of the bitwise XOR operations, and Tassa's scheme shows the costs of arithmetic operations.

6 Conclusions

In this paper, we focused on a fast $(\mathbf{k}, n)$ hierarchical secret sharing scheme that also supports efficient deletion of the secret. To achieve this, we applied a hierarchy to Kurihara et al.'s XOR-based secret sharing scheme and found that all elements corresponding to the secret and some random values of the binary generator matrix are set to 1 for the lower-level participants. In addition, we showed that the traditional allocation issue with participant identities could be removed. Given that the implementation used in our experiments can be applied to any level, we found that our implementation was able to recover a given secret with $\mathbf{k} = \{1, 3\}$ and $n = 5$ at a processing speed of approximately 6.75 Gbps.

Acknowledgments. The authors would like to thank the anonymous reviewers for their helpful and constructive comments. This work was supported by JSPS KAKENHI Grant Number JP18K11306.

References

1. Blakley, G.R.: Safeguarding cryptographic keys. AFIPS **48**, 313–317 (1979)
2. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
3. Tassa, T.: Hierarchical threshold secret sharing. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 473–490. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_26
4. Tassa, T.: Hierarchical Threshold secret sharing. J. Cryptol. **20**(2), 237–264 (2007)
5. Fujii, Y., Tada, M., Hosaka, N., Tochikubo, K., Kato, T.: A fast $(2, n)$-threshold scheme and its application. CSS **2005**, 631–636 (2005). (in Japanese)
6. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: A fast $(3, n)$-threshold secret sharing scheme using Exclusive-OR operations. IEICE Trans. Fundam. **E91-A**(1), 127–138 (2008)
7. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: On a fast (k, n)-threshold secret sharing scheme. IEICE Trans. Fundam. **E91-A**(9), 2365–2378 (2008)
8. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: A new (k,n)-threshold secret sharing scheme and its extension. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 455–470. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85886-7_31
9. Kurihara, J., Uyematsu, T.: A novel realization of threshold schemes over binary field extensions. IEICE Trans. Fundam. **E94-A**(6), 1375–1380 (2011)
10. Feng, G.-L., Deng, R.-H., Bao, F.: Packet-loss resilient coding scheme with only XOR operations. IEE Proc. Commun. **151**(4), 322–328 (2004)
11. Blömer, J., Kalfane, M., Karp, R., Karpinski, M., Luby, M., Zuckerman, D.: An XOR-Based Erasure-Resilient Coding Scheme. ICSI Technical report TR-95-048 (1995)
12. Chen, L., Laing, T.M., Martin, K.M.: Efficient, XOR-based, ideal $(t, n)-$threshold schemes. In: Foresti, S., Persiano, G. (eds.) CANS 2016. LNCS, vol. 10052, pp. 467–483. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48965-0_28
13. Selçuk, A.A., Kaşkaloğlu, K., Özbudak, F.: On hierarchical threshold secret sharing. IACR Cryptology ePrint Archive 2009, p. 450 (2009)
14. Simmons, G.J.: How to (Really) share a secret. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 390–448. Springer, New York (1990). https://doi.org/10.1007/0-387-34799-2_30
15. Brickell, E.F.: Some ideal secret sharing schemes. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 468–475. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-46885-4_45
16. Castiglione, A., De Santis, A., Masucci, B.: Hierarchical and shared key assignment. NBiS-2014, pp. 263–270 (2014)
17. Beimel, A.: Secret-sharing schemes: a survey. In: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) IWCC 2011. LNCS, vol. 6639, pp. 11–46. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20901-7_2
18. Blundo, C., De Santis, A., Gargano, L., Vaccaro, U.: On the information rate of secret sharing schemes. TCS **154**, 283–306 (1996)
19. Blundo, C., De Santis, A., Gargano, L., Vaccaro, U.: On the information rate of secret sharing schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 148–167. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-48071-4_11
20. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: A fast (k, L, n)-threshold ramp secret sharing scheme. IEICE Trans. Fundam. **E92-A**(8), 1808–1821 (2009)

21. Shima, K., Doi, H.: $(\{1,3\},n)$ hierarchical secret sharing scheme based on XOR operations for a small number of indispensable participants. AsiaJCIS 2016, pp. 108–114 (2016)
22. Shima, K., Doi, H.: A hierarchical secret sharing scheme over finite fields of characteristic 2. J. Inf. Process. **25**, 875–883 (2017)

Symmetric-Key Primitives

Integer Linear Programming for Three-Subset Meet-in-the-Middle Attacks: Application to GIFT

Yu Sasaki(✉)

NTT Secure Platform Laboratories,
3-9-11, Midori-cho, Musashino-shi, Tokyo 180-8585, Japan
sasaki.yu@lab.ntt.co.jp

Abstract. This article presents a new usage of integer-linear-programming (ILP) for block-cipher analysis, in particular for automating a procedure to search for optimal independent key bits used in a meet-in-the-middle (MitM) attack. The research is motivated by a recent lightweight block-cipher design GIFT, in which the evaluation by the designers has some room to be improved. The developed tool finds optimal choices of independent key bits, which improves the complexity of the 15-round MitM attack, the current best attack, on GIFT-64 from 2^{120} to 2^{112}.

Keywords: GIFT · Block cipher · Cryptanalysis · Symmetric-key Meet-in-the-middle · Integer linear programming

1 Introduction

Lightweight cryptography, which is one of the most actively discussed topics in the current symmetric-key community, studies cryptographic technologies that are particularly useful for extremely resource-constraint environments, e.g. sensor networks and radio frequency identifier (RFID) systems. Those technologies are important for the coming age of Internet-of-Things, in which a lot of sensitive data is measured by sensors that are not equipped with a powerful CPU and the data is communicated through public channels.

Block ciphers are one of the most fundamental primitives for symmetric-key cryptographic schemes. Since the pioneering ultra-lightweight block cipher design PRESENT [6], a huge number of lightweight block ciphers have been designed especially during the last decade. Readers may refer to [1] for a list of existing lightweight block-cipher designs.

Given a lot of lightweight block ciphers, it is now necessary for the community to choose good designs. Indeed, lightweight block ciphers have recently been discussed extensively not only in academia but also by standardization bodies such as ISO, NIST and CRYPTREC.

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 227–243, 2018.
https://doi.org/10.1007/978-3-319-97916-8_15

At CHES 2017, Banik et al. proposed a new lightweight block cipher called GIFT [3], which outperforms even SIMON [4] or SKINNY [5] in round-based hardware implementation. According to the designers, GIFT was designed by revisiting PRESENT's design strategy and by pushing its implementation cost to limits, while providing special care on security. For example, the maximum differential probability of its 4-bit S-box is $2^{-1.415}$ instead of standard 2^{-2} but the bit-shuffle was designed to avoid having such bad propagation many times. Such a compromise of security enables the designers to have lighter design choices. Moreover, from this innovation, the designers claim that GIFT improves PRESENT both in efficiency and security. Indeed, the number of rounds for 64-bit block and 128-bit key is 28, which is smaller than that of PRESENT. Due to this property, security analysis of GIFT is of great interest.

To the best of our knowledge, no third party analysis is available on GIFT. The designers provided several dedicated cryptanalyses. Two deeply discussed attacks are a 14-round integral attack and a 15-round meet-in-the-middle (MitM) attack, here the MitM attack is a rather classical one [9] (sometimes called three-subset MitM attack [7]) that divides the attack target into two independent parts.

In three-subset MitM attacks, the target cipher E_K is divided into two parts $E^1_{K_1}$ and $E^2_{K_2}$ so that $E_K = E^2_{K_2} \circ E^1_{K_1}$, where K_1 and K_2 are sets of key bits involved in the computations of E^1 and E^2, respectively. Let A_1 and A_2 be sets of key bits used in only E_1 and E_2, respectively. Then for each guess of the intersection of A_1 and A_2, $E_1(P)$ and $E_2^{-1}(C)$ for a pair of plaintext and ciphertext (P, C) are computed independently, thus the MitM attack is applied.

The designers, without explaining any rational, chose two sets of particular 8 key bits as A_1 and A_2 for GIFT-64. This raises the following issues.

- The optimality of the choice of key bits for A_1 and A_2 is unclear.
- More generally, is there any method, possibly automated, to identify the optimal choice of key bits for A_1 and A_2?

Our Contributions. In this paper, we positively answer the above questions by developing an optimal search method using integer-linear-programming (ILP). Thanks to this tool, we successfully find new choices of independent key bits for three-subset MitM attack that improve the complexity of the previous MitM attack against GIFT-64 from 2^{120} to 2^{112}, namely enlarges the size of A_1 and A_2 from 8 to 16 bits. Our approach also ensures that there does not exist any choice of independent key bits to achieve the complexity of 2^{111} or below. The attack results are summarized in Table 1.

Table 1. Summary of attacks against GIFT-64.

Rounds	Approach	Time	Data	Memory	Optimality	Ref.
14/28	Integral	2^{96}	2^{63}	2^{63}	N/A	[3]
15/28	MitM	2^{120}	2^{64}	2^{8}	–	[3]
15/28	MitM	2^{112}	2^{64}	2^{16}	✓	Ours

Applications of ILP and mixed integer linear programming (MILP) to cryptanalyses against block ciphers have been discussed actively, e.g. differential and linear cryptanalyses [17,20,24,25], division property [23,27], cube attack [16], impossible differential and zero-correlation cryptanalyses [8,21], and so on. We show that a part of the process to mount three-subset MitM attacks can also be automated and optimized by ILP.

From a technical viewpoint, our goal is to maximize the size of A_1 and A_2. Naturally, an objective function of our model is "maximize $|A_1|$" under the constraint that $|A_1| = |A_2|$.[1] Our model then deals with two constraints; (1) the bits affected from A_1 and A_2 do not overlap each other, and (2) the number of compared bits at the matching stage is sufficiently large (greater than or equal to $|A_1|$). An interesting feature is that those two constraints control two properties that are complement each other, i.e. the former is focusing on the independence and the latter is focusing on the match. Our model generates those constrains in almost the same way, which reduces the implementers workload.

Paper Outline. Section 2 describes the specification of `GIFT-64`. Section 3 recalls the framework of the three-subset MitM attack and shows the previous 15-round attack against `GIFT-64` presented by the designers. Section 4 explains how a problem of finding optimal separations for MitM attacks can be modeled by linear inequalities to be solved by ILP. Section 5 presents our improved MitM attack against 15-round `GIFT-64`. Section 6 discusses an application to XTEA [18]. We conclude this paper in Sect. 7.

2 Specification of `GIFT`

`GIFT` [3] is a lightweight block cipher supporting 64- and 128-bit block sizes and 128-bit key size. The former and the latter are called `GIFT-64` and `GIFT-128`, respectively. In this paper, we focus our attention on `GIFT-64`, thus omit the detailed description of `GIFT-128`.

A 64-bit plaintext P is loaded to a 64-bit state s_0. Then the state is updated by iteratively applying a round function $RF : \{0,1\}^{64} \times \{0,1\}^{32} \mapsto \{0,1\}^{64}$ 28 times as $s_i \leftarrow RF(s_{i-1}, k_{i-1})$ for $i = 1, 2, \cdots, 28$, where k_{i-1} are 28 round keys generated from a 128-bit user-specified key K by a key scheduling function $KF : \{0,1\}^{128} \mapsto (\{0,1\}^{32})^{28}$ as $(k_0, k_1, \cdots, k_{27}) \leftarrow KF(K)$. We call the computation for index i "*round* i." The last state, s_{28}, is a ciphertext C.

Round Function (*RF*). Let $x_{63}, x_{62}, \cdots, x_0$ be a 64-bit state value. The round function consists of three operations: SubCells, PermBits, and AddRoundKey.

SubCells: It applies a 4-bit to 4-bit S-box S shown in Table 2 to 16 nibbles $x_{4i+3}, x_{4i+2}, x_{4i+1}, x_{4i}$, $\forall i = 0, 1, \cdots, 15$ in parallel.

[1] Generally, the cost of two functions in the MitM attack can be unbalanced. Then choosing different weight of $|A_1|$ and $|A_2|$ may be optimal. If the application is limited to the three-subset MitM attacks, two functions are usually balanced.

Table 2. S-box.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	1	a	4	c	6	f	3	9	2	d	b	7	5	0	8	e

PermBits: A bit-permutation π shown in Table 3 is applied to the 64-bit state.

Table 3. Bit-permutation.

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi(x)$	0	17	34	51	48	1	18	35	32	49	2	19	16	33	50	3
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\pi(x)$	4	21	38	55	52	5	22	39	36	53	6	23	20	37	54	7
x	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$\pi(x)$	8	25	42	59	56	9	26	43	40	57	10	27	24	41	58	11
x	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$\pi(x)$	12	29	46	63	60	13	30	47	44	61	14	31	28	45	62	15

AddRoundKey: This step consists of adding a round key and a round constant. A 32-bit round key k_{i-1} is extracted from the key state, it is further partitioned into two 16-bit words $k_{i-1} = U\|V = u_{15}u_{14}\cdots u_0\|v_{15}v_{14}\cdots v_0$. For GIFT-64, U and V are XORed to x_{4i+1} and x_{4i} of the state respectively.

$$x_{4i+1} \leftarrow x_{4i+1} \oplus u_i, \quad x_{4i} \leftarrow x_{4i} \oplus v_i, \quad \forall i \in \{0, 1, \cdots, 15\}.$$

Then, a single bit '1' and a 6-bit round constant are XORed to the state at bit positions 63, 23, 19, 15, 11, 7 and 3. Round constants are generated by a simple linear feedback shift register. In our analysis, the round constants do not have any impact, thus we ignore them hereafter. The schematic diagram of the `GIFT` round function is shown in Fig. 1.

Key Schedule Function (*KF*). A 128-bit user-specified key K is loaded to a 128-bit key state that is composed of eight 16-bit words $\kappa_7, \kappa_6, \cdots, \kappa_0$. A round key is first extracted from the key state before the key state update. For `GIFT-64`, two 16-bit words of the key state are extracted as the round key $k_{i-1} = U\|V$,

$$U \leftarrow \kappa_1, \quad V \leftarrow \kappa_0.$$

The key state is then updated as follows,

$$\kappa_7\|\kappa_6\|\kappa_5\|\cdots\|\kappa_1\|\kappa_0 \leftarrow \kappa_1 \ggg 2\|\kappa_0 \ggg 12\|\kappa_7\|\cdots\|\kappa_3\|\kappa_2,$$

where $\ggg i$ is an i-bit right rotation within a 16-bit word. The schematic diagram of the `GIFT` key schedule function is illustrated in Fig. 2.

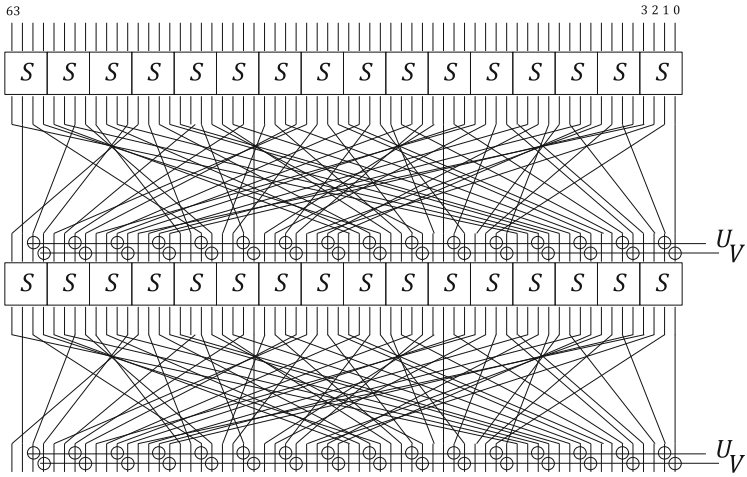


Fig. 1. Schematic diagram of two rounds of `GIFT-64`.

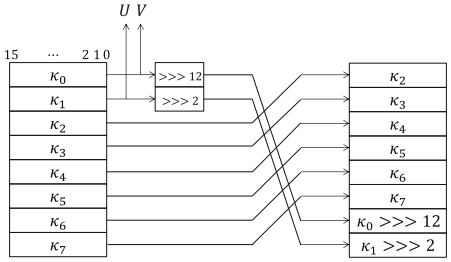


Fig. 2. Schematic diagram of key schedule function of `GIFT-64`.

Security. `GIFT` aims at single-key security. The designers do not claim any related-key security even though no attack is known in this model as of today.

3 Three-Subset Meet-in-the-Middle Attacks

A three-subset meet-in-the-middle attack was proposed by Bogdanov and Rechberger [7] and was well summarized by Isobe [12,13]. We first recall its framework in Sect. 3.1 and explain the previous attack on `GIFT-64` in Sect. 3.2.

3.1 General Framework

Basic Framework. A block-cipher E_K is divided into two parts $E^1_{K_1}$ and $E^2_{K_2}$ so that $E_K(x) = E^2_{K_2} \circ E^1_{K_1}(x), x \in \{0,1\}^b$, where K_1 and K_2 are sets of key bits involved in the computations of E^1 and E^2, respectively. Then, three sets of key bits A_0, A_1, and A_2 are defined as follows.

– A_0: a common set of key bits used in both E_1 and E_2, namely $A_0 = K_1 \cap K_2$.
– A_1: a set of key bits used only in E_1, namely $A_1 = K_1 \setminus K_1 \cap K_2$.
– A_2: a set of key bits used only in E_2, namely $A_2 = K_2 \setminus K_1 \cap K_2$.

The attack first prepares a pair of plaintext and ciphertext denoted by (P, C). The attacker exhaustively guesses A_0 and for each of them, compute $E^1_{A_1}(P)$ and $D^2_{A_2}(C)$ independently for all candidates of A_1 and A_2, where D^2 denotes a decryption of E^2. The correct key always leads to a match, thus the key space can be reduced efficiently. E^1 and D^2 are a chunk of rounds in a practical block-cipher design, thus called *a forward chunk* and *a backward chunk*, respectively. A formal description in an algorithm style is given in Algorithm 1.

Algorithm 1. Basic Three Subset Meet-in-the-Middle Attack

Input: $E^1, D^2, A_0, A_1, A_2, (P, C)$, a list L used to store key candidates
Output: a list of key candidates L

1: $L \leftarrow \phi$.
2: **for** all candidates of A_0 **do**
3: **for** all candidates of A_1 **do**
4: Compute $E^1_{A_1}(P)$, and store it along with A_1 in a table T.
5: **end for**
6: **for** all candidates of A_2 **do**
7: Compute $D^2_{A_2}(C)$.
8: **if** the same value as $D^2_{A_2}(C)$ exists in T **then**
9: $L \leftarrow L \cup (A_0 \cup A_1 \cup A_2)$.
10: **end if**
11: **end for**
12: **end for**
13: **return** L

In Step 4, the attack requires a memory to store $2^{|A_1|}$ values. The computational complexity of Algorithm 1 is $2^{|A_0|}(2^{|A_1|} + 2^{|A_2|})$. The attack can be faster than the brute force attack by a factor of $\min(2^{|A_1|}, 2^{|A_2|})$. Therefore it is crucial to find large A_1 and A_2 in the MitM attack.

Instead of separating E_K into $E^1_{K_1}$ and $E^2_{K_2}$, it is often separated into three parts such that $E_K = E^2_{K_2} \circ E^{\text{skip}}_{K_1,K_2} \circ E^1_{K_1}$ as illustrated in Fig. 3. In this case, $E^1_{K_1}(P)$ and $D^2_{K_2}(C)$ cannot directly match due to the existence of E_{skip}. However, in a typical block cipher design, full diffusion is not achieved only in 1 round, hence state values inside E_{skip} can partially be computed without the knowledge of round keys. As a result, by reducing the number of matched bits, the number of attacked rounds may increase. Let m be the number of matched bits. As long as $m \geq \min(|A_1|, |A_2|)$, the MitM attack is faster than the brute force attack by a factor of $2^{\min(|A_1|,|A_2|)}$. If $m < \min(|A_1|, |A_2|)$, the improved factor is reduced to 2^m. In other words, the improved factor is $2^{\min(|A_1|,|A_2|,m)}$.

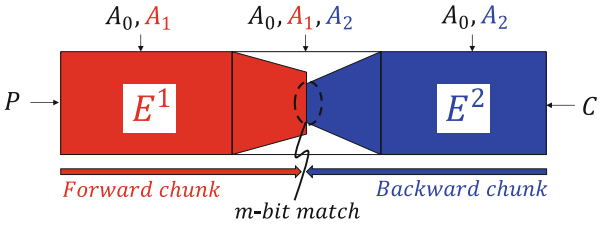


Fig. 3. Three-subset MitM attack with partial match.

Initial Structure and Splice-and-Cut. A few more techniques are known to extend the number of attacked rounds, e.g. *splice-and-cut* technique by Aoki and Sasaki [2] and the *initial-structure* technique by Sasaki and Aoki [19].

The splice-and-cut technique enables the attacker to divide E_K into $E^2_{K_2} \circ E_{\text{skip}} \circ E^1_{K_1} \circ \overline{E^2}_{K_2}$. The attacker first fixes the value of the state between $\overline{E^2}_{K_2}$ and $E^1_{K_1}$, which is denoted by X. In the backward chunk, the attacker computes $P = \overline{D^2}_{A_2}(X)$ for all possible values of A_2, and makes encryption queries P to obtain the corresponding ciphertext C. Then, $D^2_{A_2}(C)$ is computed. If the change of A_2 propagates to all bits in P, the attack requires the full code book.

The initial-structure technique enables to inject another layer depending on all of A_0, A_1, A_2 at the boundary of two chunks. It again exploits an imperfect diffusion in a small number of rounds. As long as the impact of changing A_1 in the forward direction and the impact of changing A_2 in the backward direction do not overlap each other, two chunks can be computed independently.

The three-subset MitM attack with the splice-and-cut and initial structure techniques is illustrated in Fig. 4. Those techniques are successfully applied to many ciphers, e.g. generic Feistel construction [15], 3-round Even-Mansour construction [10], dedicated designs KTANTAN [26] and XTEA [14,22].

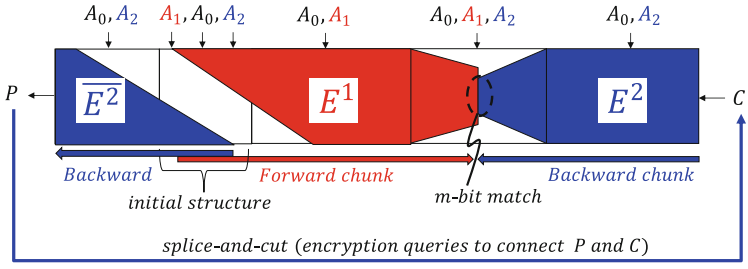


Fig. 4. Splice-and-cut and initial structure techniques.

3.2 Previous Three-Subset MitM Attack on 15-Round GIFT-64

The designers of GIFT presented a 15-round three-subset MitM attack against GIFT-64 [3]. In GIFT-64, two words of the key are rotationally used in every 4 rounds. Namely, (κ_0, κ_1) for round i when $i \mod 4 = 0$, (κ_2, κ_3) when $i \mod 4 = 1$, (κ_4, κ_5) when $i \mod 4 = 2$, and (κ_6, κ_7) when $i \mod 4 = 3$, though bit-positions inside each word are rotated.

Given this property and the framework in Fig. 4, (κ_6, κ_7) and (κ_2, κ_3) were chosen as sources of independent computations for 15 rounds. Those bits are called *neutral bits* and stressed by superscripts ^{F} and ^{B} for forward and backward chunks, respectively. The chunk separation is shown in Table 4.

Table 4. Chunk separation of the previous 15-round MitM attack.

Round		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
round key	U	κ_1	κ_3^B	κ_5	κ_7^F	κ_1	κ_3^B	κ_5	κ_7^F	κ_1	κ_3^B	κ_5	κ_7^F	κ_1	κ_3^B	κ_5
	V	κ_0	κ_2^B	κ_4	κ_6^F	κ_0	κ_2^B	κ_4	κ_6^F	κ_0	κ_2^B	κ_4	κ_6^F	κ_0	κ_2^B	κ_4
Remarks		$\longleftarrow$			IS			$\longrightarrow$			E^{skip}			$\longleftarrow$		

κ_2 and κ_3 never appear from rounds 7 to 9 and κ_6 and κ_7 never appear from rounds 3 to 1 and 15 to 13. Hence, independence of two chunks is ensured during these rounds. The remaining task for the attacker is to choose neutral bits satisfying the following two conditions simultaneously:

- Propagations in two chunks never overlap during rounds 4 to 6.
- Partial computations during rounds 10 to 12 have sufficient matching bits.

Choice by the Designers and Challenges. The designers chose 8 bits of (κ_6, κ_7) and 8 bits of (κ_2, κ_3) as neutral bits, which has 40 bits to match during rounds 10 to 12. As explained in Sect. 3.1, the complexity of the attack is determined by the minimum value between the number of neutral bits for each chunk and the number of matched bits. Thus, the attack complexity is $2^{128-\min(8,8,40)} = 2^{120}$.

This fact motivated us to tackle the following two problems.

- The choice of the previous attack is very unbalanced with respect to the number of the neutral bits and the matched bits (8 versus 40). More balanced choices may exist, which improves the attack complexity.
- More generally, for a given chunk separation, e.g. Table 4, is there any method, possibly automated one, to find the optimal choice of neutral bits?

In this paper, we positively answer those questions by developing an optimal search method using ILP, which improves the attack complexity to 2^{112}.

Remarks on GIFT-128. The designers [3] claimed that the MitM attack is harder for GIFT-128 than GIFT-64. This is because the size of the round key in GIFT-128 is 64 bits, which is a double of the one in GIFT-64. To be more precise, $\kappa_0, \kappa_1, \kappa_4, \kappa_5$ are used in round $2i+1$ and $\kappa_2, \kappa_3, \kappa_6, \kappa_7$ are used in round $2i$. Hence, two chunks must be shorter in GIFT-128 than in GIFT-64. With this reason, we do not discuss the MitM attack against GIFT-128.

4 ILP Model to Search for Optimal Neutral Bits

Our goal is, given the chunk separation by the designers shown in Table 4, to identify the optimal choices of neutral bits by using ILP. It may be helpful for readers to refer to illustrations of 3-round initial structure in Fig. 5 and 3-round partial-matching in Fig. 6 to obtain rough ideas of our modeling.

Overall Strategy. In high-level, we maximize the number of neutral bits for (κ_2, κ_3) and for (κ_6, κ_7) under the three constraints; (1) the number of neutral bits for $(\kappa_2,\ \kappa_3)$ and for (κ_6, κ_7) are balanced, (2) the bits affected from each neutral bits during the initial structure do not overlap, and (3) partial computations from two directions during the partial-match contain a sufficient number of bits in common.

4.1 Details of ILP Models

Binary Variables for Key Bits. To define whether or not each bit is chosen as a neutral bit, we assign a binary variable for each key bit denoted by $k_{i,j}, i \in \{2,3,6,7\}, j \in \{0,1,\cdots,15\}$. Namely,

$$k_{i,j} = \begin{cases} 1 & \text{if the } j\text{-th bit of } \kappa_i \text{ is a neutral bit} \\ 0 & \text{otherwise} \end{cases}$$

Then, the objective function is simply defined as

$$\text{maximize} \quad \sum_{i\in\{6,7\}} \sum_{j=0}^{15} k_{i,j}.$$

The constraint to balance the number of neutral bits in two chunks is given by

$$\sum_{i\in\{6,7\}} \sum_{j=0}^{15} k_{i,j} = \sum_{i\in\{2,3\}} \sum_{j=0}^{15} k_{i,j}.$$

Constraints for the Initial Structure. The initial structure covers rounds 4 to 6. Independence of two chunks is detected by checking the active S-box positions in round 6. Namely, we define 16 binary variables $\mathrm{IS}_j^F, j \in \{0,1,\cdots,15\}$ such that IS_j^F is 1 only if the j-th S-box in round 6 is active (gets affected by neutral bits) in

the forward chunk. Similarly, we define $\mathrm{IS}_j^B, j \in \{0, 1, \cdots, 15\}$ for the backward chunk. Then, the constraints to ensure the independence is given by

$$\mathrm{IS}_j^F + \mathrm{IS}_j^B \leq 1 \quad \text{for } j \in \{0, 1, \cdots, 15\},$$

which means that each S-box is affected by at most 1 chunk.

We then need to describe valid relationships between (k_6, k_7) and IS^F and between (k_2, k_3) and IS^B.

In the backward chunk, the j-th S-box in round 6 can be active only when bit position $\pi(4j)$ or $\pi(4j+1)$ in round 6 or both of them are activated by (k_2, k_3). Let $d^B_{\pi(4j)}$ and $d^B_{\pi(4j+1)}$ be binary dummy variables to denote whether the corresponding bit is active or not. Then IS_j^B becomes a bitwise-OR of $d^B_{\pi(4j)}$ and $d^B_{\pi(4j+1)}$, which can be modeled by three inequalities.

$$d^B_{\pi(4j)} + d^B_{\pi(4j+1)} - \mathrm{IS}_j^B \geq 0, \qquad -d^B_{\pi(4j)} + \mathrm{IS}_j^B \geq 0, \qquad -d^B_{\pi(4j+1)} + \mathrm{IS}_j^B \geq 0.$$

The relationship between the dummy variables and (k_2, k_3) are simply given by

$$d^B_{4j} = k_{2,j}, \quad d^B_{4j+1} = k_{3,j} \quad \text{for } j \in \{0, 1, \cdots, 15\}.$$

Description of the forward chunk is similar, thus we only explain it roughly. The j-th S-box in round 5 is active only when either $k_{7,j}$ or $k_{6,j}$ or both of them are active. 4 output bits from the j-th S-box in round 5 are $4j, 4j+1, 4j+2$, and $4j+3$. Here, we introduce 64 dummy binary variables d_j^F to denote whether the j-th bit after the S-box in round 5 is active or not, and they can be modeled by $d^F_{4j} = k_{7,j} \vee k_{6,j}$ and $d^F_{4j} = d^F_{4j+1}, d^F_{4j} = d^F_{4j+2}$, and $d^F_{4j} = d^F_{4j+3}$. Remember that a bitwise-OR can be modeled by 3 inequalities. Hence the above can be modeled by 6 (in)equalities. Finally, IS_j^F is a bitwise-OR of 4 input variables, i.e. $\mathrm{IS}_j^F = (d^F_{\pi^{-1}(4j+3)}, d^F_{\pi^{-1}(4j+2)}, d^F_{\pi^{-1}(4j+1)}, d^F_{\pi^{-1}(4j)})$, which can be modeled by

$$\begin{aligned}
&d^F_{\pi^{-1}(4j+3)} + d^F_{\pi^{-1}(4j+2)} + d^F_{\pi^{-1}(4j+1)} + d^F_{\pi^{-1}(4j)} - \mathrm{IS}_j^F \geq 0,\\
&-d^F_{\pi(4j)} + \mathrm{IS}_j^F \geq 0, \qquad -d^F_{\pi(4j+1)} + \mathrm{IS}_j^F \geq 0,\\
&-d^F_{\pi(4j+2)} + \mathrm{IS}_j^F \geq 0, \quad -d^F_{\pi(4j+3)} + \mathrm{IS}_j^F \geq 0.
\end{aligned}$$

Constraints for the Partial Match. First of all, bit positions of round keys are rotated due to the key schedule function. Hence, it is convenient to introduce another four 16-bit words $w_{2,j}$, $w_{3,j}$, $w_{6,j}$ and $w_{7,j}$ with

$$\begin{aligned}
w_{2,j} &= k_{2,(j+12) \bmod 16}, & w_{3,j} &= k_{3,(j+2) \bmod 16},\\
w_{6,j} &= k_{6,(j+24) \bmod 16}, & w_{7,j} &= k_{7,(j+4) \bmod 16},
\end{aligned}$$

and work with those words for making a system of the partial match.[2] Note that the rotation is applied twice between (κ_6, κ_7) in round 4 and (κ_6, κ_7) in round 12 as it can be checked in Table 4.

[2] Introducing $w_{i,j}$ is redundant. One can directly point out the corresponding bits in k_i to build the actual system. Also note that most of ILP solvers efficiently remove redundant inequalities and variables in the system with a *presolve* algorithm, hence having such simple redundant variables does not significantly slow down the speed.

The partial match covers rounds 10 to 12. We match computations from two chunks at the input of round 12. Namely, we define 64 binary variables $\mathrm{PM}_j^F, j \in \{0, 1, \cdots, 63\}$ such that PM_j^F is 1 only if the j-th bit is *impossible* to compute without knowing the neutral bits of w_3 and w_2. PM_j^F is 0 if they are *possible* to be computed independently of the neutral bits. Similarly, we define $\mathrm{PM}_j^B, j \in \{0, 1, \cdots, 63\}$ for the backward chunk. Then, the condition to use bit position j for the filter is

$$\mathrm{PM}_j^F + \mathrm{PM}_j^B = 0 \quad \text{for } j \in \{0, 1, \cdots, 15\},$$

which means that each bit can be used for the match when it can be computed from both directions. Recall that we need to ensure that the number of matched bits must be greater than or equal to the number of neutral bits, which is defined as $\sum_{i\in\{6,7\}} \sum_{j=0}^{15} k_{i,j}$. Hence, we introduce 64 dummy variables n_j for $j \in \{0, 1, \cdots, 63\}$ that takes 1 only when $\mathrm{PM}_j^F + \mathrm{PM}_j^B = 0$. This is a negation of bitwise-OR, thus can be modeled by 3 inequalities as

$$\mathrm{PM}_j^F + \mathrm{PM}_j^B + n_j \geq 1, \quad -\mathrm{PM}_j^F - n_j \geq -1, \quad -\mathrm{PM}_j^B - n_j \geq -1.$$

Then, the constraint to have sufficient number of matched bits is given by

$$\sum_{i\in\{6,7\}} \sum_{j=0}^{15} k_{i,j} \leq \sum_{j=0}^{63} n_j.$$

The remaining task is to describe valid relationships between (w_2, w_3) and PM^F and between (w_6, w_7) and PM^B. This part is almost the same as the models for the initial structure because we simply propagate the neutral bits during those rounds. To avoid the redundancy, we omit the detailed inequalities to connect (w_2, w_3) and PM^F and (w_6, w_7) and PM^B.

4.2 Search Results

We used Gurobi Optimizer 7.0 [11] to solve the system of inequalities. The system consists of 850 linear (in)equalities and 576 binary variables, which is quite small compared to standard differential and linear trail searches. Indeed, the ILP solver stopped in a second.

The search results are summarized in Table 5 along with the data of the attack by the designers [3]. We found three choices of 16 neutral bits for each chunk having 16 bits to match.

Note that ILP usually returns one solution. By using the idea in [24], after we obtained the solution, we added a constraint to remove the discovered pattern from the solution space and continued to search until no solution was detected. For example, by adding $\sum_{j=8}^{15} k_{6,j} \leq 7$, the first pattern was removed from the solution space, and the tool returned another pattern.

Table 5. Three chunk separations achieving optimal complexity. Numbers denote bit-positions of the neutral bits during the initial structure.

	k_6	k_7	$k_2 \ggg 12$	$k_3 \ggg 2$
[3]	$0 \cdots 3$	$0 \cdots 3$	$4 \cdots 7$	$8 \cdots 11$
Pattern 1	$8 \cdots 15$	$8 \cdots 15$	$0 \cdots 7$	$0 \cdots 3, 12 \cdots 15$
Pattern 2	$4 \cdots 7, 12 \cdots 15$	$4 \cdots 7, 12 \cdots 15$	$4 \cdots 7, 12 \cdots 15$	$0 \cdots 3, 8 \cdots 11$
Pattern 3	$4 \cdots 7, 8 \cdots 11$	$4 \cdots 11$	$4 \cdots 11$	$0 \cdots 7$

5 Improved MitM Attacks on GIFT-64

We provide the improved MitM attack against GIFT-64 by using pattern 1 in Table 5. The chunk separation is exactly the same as [3] shown in Table 4. We now set 16 neutral bits as specified in Table 5, and show that forward and backward computations do not overlap each other during the 3-round initial structure and 16 bits can be compared during the 3-round partial match.

Initial Structure. The three-round initial structure is schematically illustrated in Fig. 5. We first set that 16 bits of the state between PermBits and AddRoundKey in round 4 to some fixed value, say c. Then, whenever we choose the values of the neutral bits for the forward computation, (κ_6, κ_7), AddRoundKey can

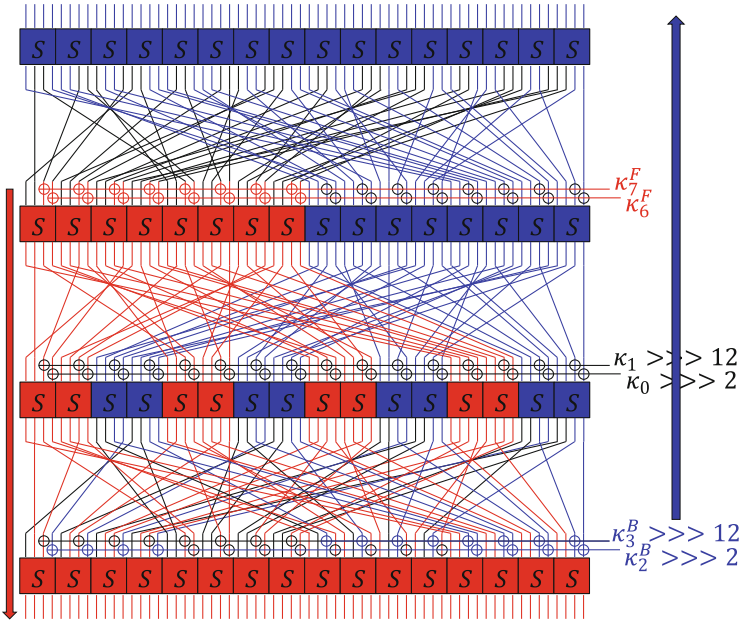


Fig. 5. Three-round initial structure of the improved MitM attack.

be computed as $c \oplus (\kappa_6, \kappa_7)$. In the backward computation, whenever those values are referred during the inverse of SubCells in round 4, those bits are treated as c. Hence, two chunks can be computed independently at the boundary of AddRoundKey in round 4. Similarly, we set 16 bits of the state between AddRoundKey in round 6 and SubCells in round 7 to another fixed value.

The neutral bits in (κ_6, κ_7) activate 8 left most S-boxes in round 5. The output 32 bits are distributed to 8 S-boxes in round 6, and then further distributed to 32-bits of the entire state for SubCells in round 7. On the other hand, neutral bits in (κ_2, κ_3) are chosen to avoid affecting the same bits. Indeed, after the inverse of PermBits in round 6, different 8 S-boxes are activated. Then from the property of the permutation, computations in two chunks never overlap during rounds 5 and 4.

Partial-Match. The three-round partial-match is schematically illustrated in Fig. 6. The neutral-bit positions are rotated from Fig. 5 due to the key schedule.

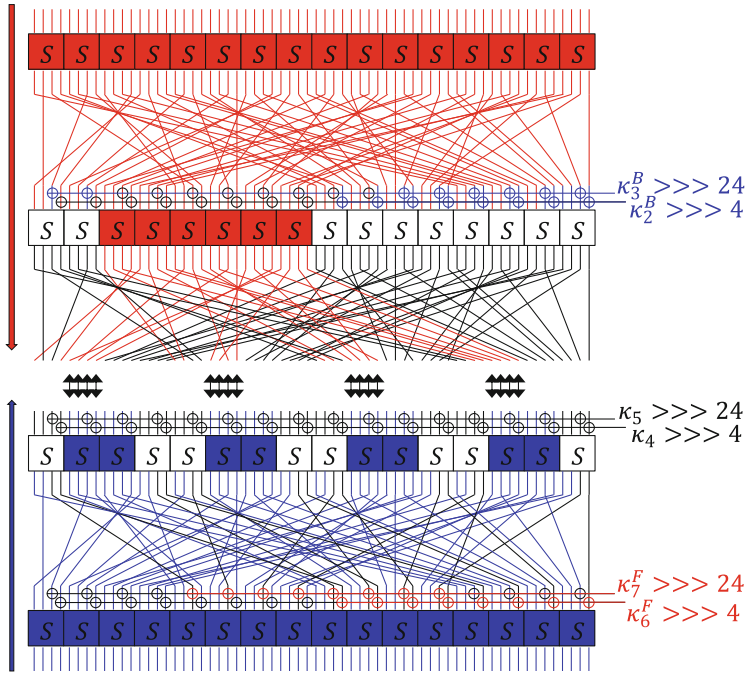


Fig. 6. Three-round partial-match of the improved MitM attack.

The forward computation starting from round 4 can continue until PermBits in round 10, but neutral bits for the backward chunk in the AddRoundKey in round 10 prevents the full-state computation. Here, as shown in Fig. 6, six

active S-boxes are independent with neutral bits in (κ_2, κ_3), which leads to 24-bit information after the PermBits in round 11.

The backward computation starts from round 6 to the plaintext. Then, the corresponding ciphertext is obtained through the encryption queries, and the backward computation continues until round 13. The backward computation loses 16-bit information after the inverse of AddRoundKey in round 12, but the other 48 bits are still computable. After the inverse PermBits, known-bit positions cover 4 bits of the S-box output for 8 S-boxes, which leads to 32-bit information right after the inverse of AddRoundKey in round 11.

The known-bit positions from both directions overlap in 16 bits (bit positions 8–11, 24–27, 40–43, and 56–59). Hence, 16-bit filter is applied.

Complexity Analysis. In the framework of the three-subset MitM attack in Sect. 3, $|A_1| = |A_2| = 16$, and $|A_0| = 96$. The attack exhaustively guesses 96-bit value of A_0, and for each of them, both of forward and backward chunks are computed for 2^{16} choices of neutral bits. This roughly takes $\frac{7}{15} \cdot 2^{96} \cdot 2^{16} \approx \frac{7}{15} \cdot 2^{112}$ round function operations for the forward chunk and $\frac{9}{15} \cdot 2^{96} \cdot 2^{16} \approx \frac{9}{15} \cdot 2^{112}$ round function operations for the backward chunk. The sum of two costs are $\frac{16}{15} \cdot 2^{112}$, but after some optimization, the constant factor can be removed. Here, we omit the details and truncating the complexity to 2^{112}.

For each guess of $|A_0|$, the number of pairs to match is $2^{16+16} = 2^{32}$ and about 2^{16} matched candidates are obtained after the 16-bit match at round 11. Hence, after 2^{96} iterations of A_0, $2^{96+16} = 2^{112}$ candidates will remain. Finally, those 2^{112} candidates can be tested by using 2 more plaintext-ciphertext pairs.

The attack complexity is $(Time, Data, Memory) = (3 \cdot 2^{112} \approx 2^{113.6}, 2^{64}, 2^{16})$. By applying the same metric as the designers [3], this complexity can be interpreted as $(Time, Data, Memory) = (2^{112}, 2^{64}, 2^{16})$.

6 Comments on Three-Subset MitM Attack on XTEA

The three-subset MitM was applied to a lightweight block cipher XTEA [18], e.g. 28-round attack [22] and 29-round attack [14], hence a natural question is whether or not our method can be applied to XTEA.

XTEA is a 64-bit block cipher having 2-branch Feistel network. Its internal state is represented by two 32-bit words L_i and R_i and the round function can be represented as follows.

$$
\begin{aligned}
L_{i+1} &\leftarrow R_i, \\
R_{i+1} &\leftarrow L_i \boxplus \big((const_i \boxplus K_i) \oplus ((R_i \ll 4 \oplus R_i \gg 5) \boxplus R_i)\big),
\end{aligned}
$$

where "$\boxplus$" is a 32-bit modular addition. The initial structure technique cannot be applied to the XTEA, thus the previous 29-round attack has the form of Fig. 3. Then, we only need to optimize the partial-matching phase.

Due to the carry during the $\boxplus$ operation, when the bit position $x, x \in \{0, 1, \cdots, 31\}$ is unknown, all bits from $x+1$ to 31 becomes unknown. This property makes searching for the optimal choices of neutral bits very easy. Indeed, it can be manually checked that the choice in the 29-round attack is optimal.

Our ILP model to optimize the choice of neutral bits can be applied to XTEA to automatically run this by hand analysis. Because the results obviously cannot improve the previous attack, we omit the exact models for XTEA.

7 Concluding Remarks

In this paper, we revisited the three-subset MitM attack on `GIFT-64`. We developed a new ILP model that takes input as the chunk separation and outputs the optimal choices of neutral bits in the key. The tool successfully returned all the optimal choices of neutral bits, which improves the complexity of the current best attack from $(Time, Data, Memory) = (2^{120}, 2^{64}, 2^8)$ to $(2^{112}, 2^{64}, 2^{16})$.

Towards lightweight design, a recent trend is making a key schedule function as simple as possible. We hope that the proposed tool will help cryptographers for optimizing details of three-subset MitM attacks.

References

1. Biryukov, A., Großschädl, J., Le Corre, Y.: CryptoLUX, Lightweight Cryptography (2015). https://www.cryptolux.org/index.php/Lightweight_Cryptography
2. Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04159-4_7
3. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: `GIFT`: a small present - towards reaching the limit of lightweight encryption. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 321–345. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_16
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013)
5. Beierle, C., et al.: The `SKINNY` family of block ciphers and its low-latency variant `MANTIS`. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 123–153. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_5
6. Bogdanov, A., et al.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74735-2_31
7. Bogdanov, A., Rechberger, C.: A 3-subset meet-in-the-middle attack: cryptanalysis of the lightweight block cipher KTANTAN. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19574-7_16
8. Cui, T., Jia, K., Fu, K., Chen, S., Wang, M.: New automatic search tool for impossible differentials and zero-correlation linear approximations. Cryptology ePrint Archive, Report 2016/689 (2016). https://eprint.iacr.org/2016/689

9. Diffie, W., Hellman, M.E.: Exhaustive cryptanalysis of the NBS data encryption standard. Comput. Issue **6**(10), 74–84 (1977)
10. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: Key recovery attacks on 3-round Even-Mansour, 8-step LED-128, and full AES^2. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 337–356. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42033-7_18
11. Gurobi Optimization Inc.: Gurobi optimizer 7.0. Official webpage (2015). http://www.gurobi.com/
12. Isobe, T.: A single-key attack on the full GOST block cipher. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 290–305. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21702-9_17
13. Isobe, T.: A single-key attack on the full GOST block cipher. J. Cryptol. **26**(1), 172–189 (2013)
14. Isobe, T., Shibutani, K.: Security analysis of the lightweight block ciphers XTEA, LED and Piccolo. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 71–86. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31448-3_6
15. Isobe, T., Shibutani, K.: Generic key recovery attack on feistel scheme. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 464–485. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42033-7_24
16. Li, Z., Bi, W., Dong, X., Wang, X.: Improved conditional cube attacks on keccak keyed modes with MILP method. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 99–127. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_4
17. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C.-K., Yung, M., Lin, D. (eds.) Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34704-7_5
18. Needham, R.M., Wheeler, D.J.: TEA extensions. Technical report, Computer Laboratory, University of Cambridge (1997)
19. Sasaki, Y., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_8
20. Sasaki, Y., Todo, Y.: New differential bounds and division property of LILLIPUT: block cipher with extended generalized feistel network. In: Avanzi, R., Heys, H. (eds.) SAC 2016. LNCS, vol. 10532, pp. 264–283. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69453-5_15
21. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part III. LNCS, vol. 10212, pp. 185–215. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_7
22. Sasaki, Y., Wang, L., Sakai, Y., Sakiyama, K., Ohta, K.: Three-subset meet-in-the-middle attack on reduced XTEA. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 138–154. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31410-0_9
23. Sun, L., Wang, W., Wang, M.: Automatic search of bit-based division property for ARX ciphers and word-based division property. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 128–157. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_5

24. Sun, S., et al.: Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. Cryptology ePrint Archive, Report 2014/747 (2014)
25. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_9
26. Wei, L., Rechberger, C., Guo, J., Wu, H., Wang, H., Ling, S.: Improved meet-in-the-middle cryptanalysis of KTANTAN (poster). In: Parampalli, U., Hawkes, P. (eds.) ACISP 2011. LNCS, vol. 6812, pp. 433–438. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22497-3_31
27. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 648–678. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_24

Symbolic-Like Computation and Conditional Differential Cryptanalysis of QUARK

Jingchun Yang[1,2], Meicheng Liu[1,2], Dongdai Lin[1,2(✉)], and Wenhao Wang[1,2]

[1] State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{yangjingchun,liumeicheng,ddlin,wangwenhao}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences,
Beijing, China

Abstract. At ASIACRYPT 2010, Knellwolf *et al.* proposed a general analysis of NFSR-based cryptosystems, called *conditional differential cryptanalysis*. The main idea of this technique is to impose conditions on the internal state to get a deterministic differential characteristic for a large number of rounds. In this paper, we propose a method, called *symbolic-like computation*, to simulate the differential propagation of an iterated cryptosystem. By coding the internal state bits and modeling the bit operations, it can determine the constantness of the differential expression with linear time complexity. Based on this method, we can obtain a list of good input differences. We apply this technique to the conditional differential cryptanalysis of QUARK, a family of lightweight hash functions proposed by Aumasson *et al.* at CHES 2010. By controlling the propagation of differences both backwards and forwards, we can observe the bias of output difference at a higher round. Eventually, we can distinguish U-QUARK/D-QUARK/S-QUARK/C-QUARK up to 155/166/259/452 rounds respectively. Our distinguishers are very practical and have been fully verified by experiments on a single PC. To the best of our knowledge, all these results are the best thus far.

Keywords: Cryptanalysis · Conditional differential cryptanalysis
NFSR · Symbolic-like computation · QUARK

1 Introduction

In modern cryptographic primitives, nonlinear feedback shift register (NFSR) plays an important role in the applications with constrained environments, like

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61379139 and 61672516), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA06010701), and the Fundamental Theory and Cutting Edge Technology Research Program of Institute of Information Engineering, CAS (Grant No. Y7Z0331102).

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 244–261, 2018.
https://doi.org/10.1007/978-3-319-97916-8_16

the radio-frequency identification devices (RFID) and sensor networks. Based on the structure of NFSR, many famous ciphers have been proposed, such as the well known stream cipher Trivium [8] and Grain [12], the lightweight block cipher family KATAN/KTANTAN [9], and the hash function family QUARK [2–4]. All these algorithms possess an efficient hardware implementation and a high level security at the same time.

In cryptanalysis, NFSR-based cryptosystems are often analyzed as a boolean function f whose variables derive from the state bits in the registers. In recent years, cube attacks [10,11] and conditional differential cryptanalysis [13,21] have proved to be very effective against such ciphers. Cube attacks work on finding many derivatives of the boolean function f, and then deriving the linear equations in the state bits. While conditional differential cryptanalysis consists in analysing the bias of the output of derivatives of f on well chosen inputs.

Conditional differential cryptanalysis was proposed by Knellwolf *et al.* at ASIACRYPT 2010 [13]. In this attack, attackers analyze the propagation of differences round by round. At the early iterations of the cipher, conditions are imposed on the internal state bits to control the propagation of the difference. Attackers generate the input samples according to the conditions and introduce the input difference at the initial rounds. In the end, they will be able to observe a bias in the output difference. In this way, Knellwolf *et al.* obtained a distinguisher for Grain v1 up to 104 rounds.

Actually, the notion of conditional differential cryptanalysis was inspired by message modification techniques as they were introduced in [20] to speed up the collision search for hash functions. The main idea of message modification techniques is to choose the message words and internal chaining variables in an attack on the hash function to fulfill the conditions imposed by the differential characteristic in a deterministic way. This technique led to the collision attacks on full MD5 [20], SHA-1 [19] and reduced SHA-256 [7].

Up to now, conditional differential cryptanalysis has attracted a lot of attention. In 2015, Banik [5] proposed the conditional differential cryptanalysis for 105-round Grain v1. The complexity for the key recovery attack was faster than exhaustive search by 2^9. In 2016, Ma *et al.* [16] proposed two conditional differential attacks towards Grain-128a. In attack A, they could retrieve 18 secret key expressions for 169-round Grain-128a. In attack B, they extended the distinguishing attack against Grain-128a up to 195 rounds in a weak-key setting. In the same year, Ma *et al.* [17] improved the conditional differential attacks, they could retrieve 31 distinct secret key expressions for 107-round Grain v1 and 15 distinct secret key expressions for 110-round Grain v1. In 2016, Watanabe *et al.* [22] proposed a new method to find conditional differential characteristics on NFSR-based stream ciphers. Using this method, they could distinguish Grain v1 up to 114 rounds. Later at ACISP 2017, Watanabe *et al.* [21] proposed the method of arrangement of differences and conditions to obtain good higher-order conditional differential characteristics. They applied it to Kreyvium, an NFSR-based stream cipher oriented to homomorphic-ciphertext compression, and obtained a distinguisher on Kreyvium with 899 rounds.

With the increasing need for lightweight hash functions oriented to hardware implementation, Aumasson *et al.* proposed a novel design philosophy [2] for lightweight hash functions at CHES 2010. Inspired by the design of the stream cipher Grain and the block cipher KATAN, they presented the hash function family QUARK, composed of three instances: U-QUARK, D-QUARK, and S-QUARK. In 2012, for the need of 256-bit security, Aumasson *et al.* proposed a new design C-QUARK [4] which has a internal state of 384 bits. In 2015, Zhang *et al.* [23] improved the conditional differential cryptanalysis, and they could distinguish U-QUARK/D-QUARK/S-QUARK/C-QUARK up to 153/159/248/445 rounds respectively.

1.1 Our Contributions

In this paper, we propose a general technique, called *symbolic-like computation*, to simulate the differential propagation of an iterated cryptosystem.

It is well known that symbolic computation can accurately give the boolean expression of the output difference for an r-round iterated cryptosystem. However, with the increase of the value of r, the time/memory complexity of symbolic computation would be growing exponentially.

Sometimes, we only want to know whether the boolean expression is a constant. In this circumstance, we adopt a much faster way. First we code the internal state bits with different numbers; then we model the two basic operations AND and XOR in boolean functions, by the rule of state update, we can compute the codes of the next state iteratively; finally, we compute the codes of newly generated bits round by round, for any round r, we can determine the constantness of the differential expression of a specific state bit or an output bit according to its code. Based on this method, we can obtain a list of good input differences with a linear time complexity, which is very efficient compared to the symbolic computation.

We apply this technique to the conditional differential analysis of QUARK. Given the single-bit difference s_p, we first utilize the backward computation to derive a multi-bit input difference which leads to the single-bit difference s_p after q rounds. In the meantime, we limit the propagation of the single-bit difference by nullifying the differential expression of newly generated bits at $t \geq q$. For each imposed condition, we convert it to the equivalent condition at $t = 0$ to obtain the final conditional differential characteristic. Finally we verify this characteritic by experiments to show the maximum round where the bias of output difference can be observed.

Our results show that we have obtained new distinguishers for all instances of QUARK with the above improved techniques. Take S-QUARK as an example, we can distinguish the cipher up to 259 rounds with a complexity of 2^{15}, while the best previous analysis could only distinguish the cipher up to 248 rounds with a much higher complexity. We summarize our results in Table 1 with the comparisons with the previous results. All our results are the best thus far and have been fully verified by experiments on a single PC.

Table 1. Results on QUARK.

Cipher	#Rounds	Complexity	Ref.
U-QUARK	136	2^{27}	[3]
	153	2^{18}	[23]
	155	2^{27}	Sect. 5
D-QUARK	159	2^{27}	[3]
	159	2^{22}	[23]
	166	2^{25}	Sect. 5
S-QUARK	237	2^{27}	[3]
	248	2^{24}	[23]
	259	2^{15}	Sect. 5
C-QUARK	396	2^{20}	[4]
	445	2^{20}	[23]
	452	2^{28}	Sect. 5

In [23], the complexity is computed as $\frac{1}{\epsilon^2}$, where ϵ is the observed bias of the output difference. We also take this calculation.

The rest of the paper is organized as follows. In Sect. 2 we introduce some basic definitions and theories. Section 3 describes the algorithm of QUARK, with the method of computing backward update functions provided. In Sect. 4, we propose an iterative method for finding good input differences. In Sect. 5 we elaborate on the conditional differential cryptanalysis of QUARK. We take D-QUARK as an example, and give the cryptanalysis results for all flavors of QUARK. Section 6 concludes the paper.

2 Preliminaries

Derivatives of Boolean Functions [14,15]. Let $\mathbb{F}_2$ denote the binary field and $\mathbb{F}_2^n$ denote the n-dimensional vector space over $\mathbb{F}_2$. An n-variable Boolean function f is a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. Denote by $+$ the addition in $\mathbb{F}_2$ and $\oplus$ the addition in $\mathbb{F}_2^n$. The derivative of f with respect to $a \in \mathbb{F}_2^n$ is defined as follows:

$$\Delta_a f(x) = f(x \oplus a) + f(x).$$

The derivative of f is also a boolean funtion. If $\sigma = \{a_1, \ldots, a_i\}$ is a set of vectors in $\mathbb{F}_2^n$, let $\mathcal{L}(\sigma)$ denote the set of all 2^i linear combinations of elements in σ. The i-th derivative of f with respect to σ is defined as follows:

$$\Delta_\sigma^{(i)} f(x) = \sum_{c \in \mathcal{L}(\sigma)} f(x \oplus c).$$

In other words, the i-th derivative of f can be evaluated by summing up all 2^i evaluations of f at $\mathcal{L}(\sigma)$.

Random Boolean Functions and Frequency Test. Let D be a non-empty subspace of $\mathbb{F}_2^n$. If a boolean function f mapping from D to $\mathbb{F}_2$ whose output is an independent uniformly distributed variable, then we say f is a *random boolean function*. According to the law of large numbers, for sufficiently many inputs $x_1, \ldots, x_N \in D$ the value

$$y = \frac{2\sum_{k=1}^{N} f(x_k) - N}{\sqrt{N}}$$

approximately follows a standard normal distribution. We say f *passes the frequency test* on D at a significance level α (where α is often taken as 0.005) if

$$\Phi(y) < 1 - \frac{\alpha}{2}.$$

In cryptanalysis, frequency test can be used to distinguish a cipher from an ideal random boolean function. Let β denote the probability that a boolean function passes the frequency test. Then the distinguishing advantage is given by $1-\alpha-\beta$.

Conditional Differential Cryptanalysis [13]. Here we briefly introduce the process of conditional differential cryptanalysis. Attackers first choose some bits in the initial state as input difference. The positions should be "good" enough such that after many initialization rounds, the deterministic output difference can still be observed. Then attackers analyze the conditions on the initial state bits carefully to prevent the propagation of differences. At last, they derive a set of conditions and the differential characteristic, they also need to verify this characteristic by experiments to show the maximum round where the bias of output difference can be observed. If we can observe the bias at a higher round than ever before, then we obtain a new distinguisher for the cipher.

In this paper, we just focus on the first-order derivative. Suppose we need S pairs of input samples to detect the bias ϵ of the output difference of r-round QUARK. Let $U = \sum_{k=1}^{S} \Delta_a f(x_k)$ and $y = \frac{2U-S}{\sqrt{S}}$. Without loss of generality, let ϵ satisfy that $\Pr(\Delta_a f(x_k) = 1) = \frac{1+\epsilon}{2}$. The expected values of U and y are thus $\frac{1+\epsilon}{2} \cdot S$ and $\epsilon \cdot \sqrt{S}$ respectively. According to the formula $\Phi(y) < 1 - \frac{\alpha}{2}$, we can obtain that

$$S > \frac{u^2}{\epsilon^2}, \tag{1}$$

where $\Phi(u) = 1 - \frac{\alpha}{2}$. This inequality indicate that we need at least $\frac{u^2}{\epsilon^2}$ pairs of input samples to observe the bias ϵ of the output difference. If α is taken as 0.005, then $u^2 \approx 7.9$ by the normal distribution table.

3 A Brief Description of QUARK

At CHES 2010, Aumasson *et al.* proposed a lightweight hash function family QUARK [2]. Inspired by the design of the stream cipher Grain and the block cipher KATAN, QUARK initially composed of three instances: U-QUARK,

D-QUARK, and S-QUARK. The digest length n are 136, 176, and 256 bits respectively. As a sponge construction, QUARK can be used for message authentication, stream encryption, or authenticated encryption. In 2012, for the need of 256-bit security, Aumasson *et al.* proposed a new design C-QUARK [4] which has a internal state of 384 bits. In *Journal of Cryptology* 2013, QUARK was slightly modified [3] to address a flaw in the initial analysis.

3.1 Sponge Construction

Following the notations introduced in [6], a QUARK instance is parametrized by an *output length* n, a *rate* (or block lengh) r, and a *capacity* c. The *width* $b = r + c$ of a sponge construction is the size of its internal state. We denote this internal state $s = (s_0, \ldots, s_{b-1})$, where s_0 is referred to as the *first* bit of the state. Given a predefined initial state of b bits, the sponge construction processes a message m in three steps:

1. **Initialization.** Padding the message by appending a '1' bit and the minimal number of '0' bits such that the total length is a multiple of r.
2. **Absorbing phase.** The r-bit message blocks are XOR'd with the last r bits of the state, interleaved with the call to the permutation P.
3. **Squeezing phase.** The last r bits of the state are returned as output, interleaved with applications of the permutation P, until n bits are returned.

3.2 Permutation

The permutation P of QUARK is depicted in Fig. 1. Three feedback shift registers (FSRs) constitute the internal state of P: two nonlinear ones (NFSRs) of b/2 bits each, and a linear one (LFSR) of l bits. The state at round $t \geq 0$ is composed of

- An NFSR X of b/2 bits, denoted $X^t = (X_0^t, \ldots, X_{b/2-1}^t)$;
- An NFSR Y of b/2 bits, denoted $Y^t = (Y_0^t, \ldots, Y_{b/2-1}^t)$;
- An LFSR L of l bits, denoted $L^t = (L_0^t, \ldots, L_{l-1}^t)$.

The following lines describe the procedure of P.

Initialization. Given input of the b-bit internal state of the sponge construction $s = (s_0, \ldots, s_{b-1})$, P initializes its internal state as follows:

1. $(X_0^0, \ldots, X_{b/2-1}^0) := (s_0, \ldots, s_{b/2-1})$.
2. $(Y_0^0, \ldots, Y_{b/2-1}^0) := (s_{b/2}, \ldots, s_{b-1})$.
3. $(L_0^0, \ldots, L_{l-1}^0) := (1, \ldots, 1)$.

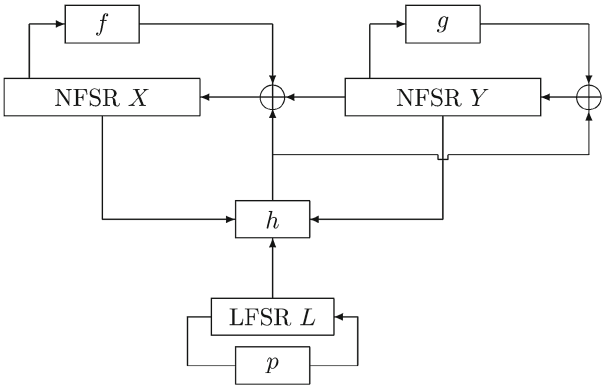


Fig. 1. The permutation of QUARK.

State Update. The current internal state (X^t, Y^t, L^t) determines the next state $(X^{t+1}, Y^{t+1}, L^{t+1})$. The update mechanism works as follows:

1. $h^t := h(X^t, Y^t, L^t)$.
2. $(X_0^{t+1}, \ldots, X_{b/2-1}^{t+1}) := (X_1^t, \ldots, X_{b/2-1}^t, Y_0^t + f(X^t) + h^t)$.
3. $(Y_0^{t+1}, \ldots, Y_{b/2-1}^{t+1}) := (Y_1^t, \ldots, Y_{b/2-1}^t, g(Y^t) + h^t)$.
4. $(L_0^{t+1}, \ldots, L_{l-1}^{t+1}) := (L_1^t, \ldots, L_{l-1}^t, p(L^t))$.

Computation of the Output. Once initialized, the state of QUARK is updated R times. The output is defined as the final value of the NFSRs X and Y:

$$s = (s_0, \ldots, s_{b-1}) = (X_0^R, X_1^R, \ldots, Y_{b/2-2}^R, Y_{b/2-1}^R).$$

3.3 Proposed Instances

Up to now, QUARK has 4 proposed versions, that is: U-QUARK, D-QUARK, S-QUARK, and C-QUARK. For each instance, we list its rate r, capacity c, width b, rounds R, and digest length n, see in Table 2. For C-QUARK, the length of the LFSR is $l = 16$, and the corresponding feedback polynomial $p(L^t)$ returns $L_0^t + L_2^t + L_3^t + L_5^t$; while for other three instances, $l = \lceil \log 4b \rceil = 10$ and $p(L^t) = L_0^t + L_3^t$.

For all flavors of QUARK, function f, g, and h are nonlinear functions of the internal state variables. For more details of the nonlinear functions of QUARK, we refer to the original papers [3,4] and the C source code [1] given by the authors.

3.4 Backward Update Functions

Notice that the backward update functions of permutation P can be easily derived. In this situation, we contrive to obtain the internal state at round t

Table 2. Parameters of the proposed instances of QUARK.

Instance	Rate (r)	Capacity (c)	Width (b)	Rounds (R)	Digest (n)
U-QUARK	8	128	136	544	136
D-QUARK	16	160	176	704	176
S-QUARK	32	224	256	1024	256
C-QUARK	64	320	384	768	384

from the state at round $t+1$. Suppose the state of X, Y, and L at round $t+1$ is $(X_0^{t+1}, \ldots, X_{b/2-1}^{t+1})$, $(Y_0^{t+1}, \ldots, Y_{b/2-1}^{t+1})$, and $(L_0^{t+1}, \ldots, L_{l-1}^{t+1})$ respectively. By the shifting of register X, we have

$$X_i^t = X_{i-1}^{t+1}, \quad i \in \{1, \ldots, b/2-1\}. \tag{2}$$

Similarly, we have

$$Y_i^t = Y_{i-1}^{t+1}, \quad i \in \{1, \ldots, b/2-1\}, \tag{3}$$

and

$$L_i^t = L_{i-1}^{t+1}, \quad i \in \{1, \ldots, l-1\}. \tag{4}$$

Denote $L_{l-1}^{t+1} = p(L^t) = L_0^t + p^*(L^t)$, where $p^*(L^t)$ does not depend on L_0^t. Since all the variables in $p^*(L^t)$ can be directly represented by the variables at round $t+1$ from Eq. (4), then $L_0^t = L_{l-1}^{t+1} + p^*(L^t)$ can be derived. So we have recovered the state of L at round t.

Denote $f(X^t) = X_0^t + f^*(X^t)$ and $g(Y^t) = Y_0^t + g^*(Y^t)$, where $f^*(X^t)$ does not depend on X_0^t, and $g^*(Y^t)$ does not depend on Y_0^t. Once again, all the variables in $f^*(X^t)$ and $g^*(Y^t)$ can be directly represented by the variables at round $t+1$ from Eqs. (2) and (3). According to the rule of state update, we have

$$\begin{aligned} X_{b/2-1}^{t+1} + Y_{b/2-1}^{t+1} =& Y_0^t + f(X^t) + g(Y^t) \\ =& Y_0^t + X_0^t + f^*(X^t) + Y_0^t + g^*(Y^t) \\ =& X_0^t + f^*(X^t) + g^*(Y^t), \end{aligned} \tag{5}$$

and

$$\begin{aligned} Y_{b/2-1}^{t+1} =& g(Y^t) + h(X^t, Y^t, L^t) \\ =& Y_0^t + g^*(Y^t) + h(X^t, Y^t, L^t). \end{aligned} \tag{6}$$

Except for L_0^t, all the variables in $h(X^t, Y^t, L^t)$ can be directly represented by the variables at round $t+1$ from Eqs. (2) and (3). Thus, from Eqs. (5) and (6), we can recover the expressions of X_0^t and Y_0^t:

$$\begin{aligned} X_0^t =& X_{b/2-1}^{t+1} + Y_{b/2-1}^{t+1} + f^*(X^t) + g^*(Y^t), \\ Y_0^t =& Y_{b/2-1}^{t+1} + g^*(Y^t) + h(X^t, Y^t, L^t). \end{aligned}$$

In this way, we have derived the backward update functions of permutation P.

4 Symbolic-Like Computation

In this section, we propose a general method for the purpose of simulating the differential propagation, called *symbolic-like computation*.

In cryptanalysis, symbolic computation has been widely used to analyze the algebraic normal form (ANF) of a cryptographic algorithm. It is well known that symbolic computation can accurately give the boolean expression of the output difference for an r-round iterated cryptosystem. However, with the increase of the value of r, the time/memory complexity of symbolic computation would be growing exponentially.

Sometimes, we only want to know whether the boolean expression is a constant. In this circumstance, we adopt a much faster way. First we code the internal state bits with different numbers; then we model the two basic operations AND and XOR in boolean functions, by the rule of state update, we can compute the codes of the next state; finally, we compute the codes of newly generated bits round by round, for any round r, we can determine the constantness of the differential expression of a specific state bit or an output bit according to its code.

To illustrate our method, we take the first-order derivative as an example. Suppose an NFSR-based cipher has a nonlinear feedback shift register. Initially, secret variables (*e.g.*, key bits) and public variables (*e.g.*, plaintext bits or IV bits) are loaded to the registers. Thus the internal state is filled with '0' or '1' bits. We denote the internal state by $(s_0, s_1, \ldots, s_{Nl-1})$. Suppose $s_{i_0}, \ldots, s_{i_{Vl-1}}$ are loaded with IV bits. Let $v = s_d$ be the single-bit difference, where $d \in \{0, \ldots, Nl-1\} \setminus \{i_0, \ldots, i_{Vl-1}\}$. The values of other non-IV variables are unknown, we denote each of them by u. Notice that the relation of any two u variables is also unknown.

Now we code the initial state bits with some specific numbers. Fixed '0' or '1' bits are just coded with number 0 or 1. The 'u' bits are coded with number 2, and the single-bit difference 'v' is coded with number 3. With the update of internal state, the boolean operations AND and XOR are frequently applied to the state bits, so more forms of state bits will emerge. For example, a '1' bit XORing a 'v' bit would give rise to a '$v+1$' bit. We summarize all forms in Table 3. For each of them, we code it with a unique number.

Table 3. Coding of the internal state bits.

Form	0	1	u	v	$v+1$	$v+u$	uv	$uv+1$	$uv+u$
Code	0	1	2	3	4	5	6	7	8

Actually, the result of AND/XOR operation of any two kinds of bits listed in Table 3 will be still in the table. The general form of all the state bits can be represented as $\alpha \cdot v + \beta \cdot 1$, where $\alpha, \beta \in \{0, 1, u\}$. In this form, v and 1 can be interpreted as some "bases", while α and β are the "co-ordinates". Since the set $\{0, 1, u\}$ is closed under the AND/XOR operation, the generating set $\alpha \cdot v + \beta \cdot 1$

is also closed under the AND/XOR operation. Notice that, if $\alpha = 0$, then the difference with respect to v is equal to 0, the corresponding codes $c_0 \in \{0, 1, 2\}$; if $\alpha = 1$, then the difference with respect to v is equal to 1, the corresponding codes $c_1 \in \{3, 4, 5\}$.

Let a and b be two bits, and their codes be $\texttt{code}(a)$ and $\texttt{code}(b)$ respectively. Define `sand` and `sxor` as the operations of $\texttt{code}(a)$ and $\texttt{code}(b)$ such that $\texttt{code}(a)$ `sand` $\texttt{code}(b) = \texttt{code}(a \cdot b)$ and $\texttt{code}(a)$ `sxor` $\texttt{code}(b) = \texttt{code}(a + b)$. As mentioned above, we do not know the value of u bits, and the relation of two u bits is also unknown, so $u + u = u$, *i.e.*, 2 `sxor` 2 = 2. Since we only introduce a single-bit difference at round $t = 0$, any two v bits in the internal state exactly have the same value, so we have $v + v = 0$, *i.e.*, 3 `sxor` 3 = 0. Similarly, the AND of uv and $v + 1$ is 0, since $v(v + 1) = 0$. We exhaust the `sand`/`sxor` of any two codes, and list the results in `sand` table and `sxor` table in Appendix A.

Using the `sand` table and the `sxor` table, one can compute the code value of the state update function whose variables are all coded.

Example 1. Suppose the length of the register is 8, and the update function is $s_t = s_{t-3}s_{t-7} + s_{t-1}s_{t-5} + s_{t-8}$. Initially, we introduce the difference at s_1, and set s_4, s_5, s_6, s_7 to 1. At $t = 8$, we compute the code value of s_8 as follows:

$$
\begin{aligned}
&\texttt{code}(s_8)\\
=&\texttt{code}(s_5s_1 + s_7s_3 + s_0)\\
=&\texttt{sxor}(\texttt{sxor}(\texttt{code}(s_5s_1), \texttt{code}(s_7s_3)), \texttt{code}(s_0))\\
=&\texttt{sxor}(\texttt{sxor}(\texttt{sand}(\texttt{code}(s_5), \texttt{code}(s_1)), \texttt{sand}(\texttt{code}(s_7), \texttt{code}(s_3))), \texttt{code}(s_0))\\
=&\texttt{sxor}(\texttt{sxor}(\texttt{sand}(1, 3), \texttt{sand}(1, 2)), 2)\\
=&5.
\end{aligned}
$$

Finding Good Single-Bit Input Differences. Now, we apply the symbolic-like computation to the search of good single-bit input differences. Given the single-bit input difference, one can iteratively compute the code of the newly generated bit by the state update function. If the code of a newly generated bit s^r_{Nl-1} at round r is equal to c_0, where $c_0 \in \{0, 1, 2\}$, then the difference in s^r_{Nl-1} is equal to 0. If the code of s^r_{Nl-1} is equal to c_1, where $c_1 \in \{3, 4, 5\}$, then the difference in s^r_{Nl-1} is equal to 1. At last, we can find a maximum round S, where a deterministic difference still exists in the internal state. For each possible single-bit input difference v_i, we repeat the above procedure to compute its maximum round S_i. Finally, we store all the (i, S_i) pairs and list them in descending order by S_i. The top k items are the candidates of good single-bit input differences.

5 Conditional Differential Cryptanalysis of QUARK

In this section, we apply the techniques described in Sect. 4 to the conditional differential cryptanalysis of QUARK. We first take D-QUARK as an example and give the complete process of analysis, then we apply it to other instances of QUARK. Our results for all flavors of QUARK are presented in Subsect. 5.2.

5.1 Analysis of D-QUARK

As illustrated in Sect. 2, the procedure of conditional differential cryptanalysis consists of the following steps:

1. **Selection of the input difference**: attackers choose some bits in the internal state as input difference. The positions should satisfy that after many initialization rounds, the deterministic output difference can still be observed.
2. **Imposing conditions**: attackers analyze the conditions on the initial state bits to prevent the propagation of differences.
3. **Verification**: attackers derive a set of conditions and the differential characteristic, and verify this characteristic by experiments to show the maximum round where the bias of output difference can be observed.

Selection of the Input Difference. The result of conditional differential analysis largely depends on the selection of both input difference and conditions of state bits. In Sect. 4, we have illustrated the process of finding good single-bit input difference. With symbolic-like computation, we can find a set of differences which can spread far enough. We apply it to D-QUARK, and obtain the following list:

Table 4. Candidates of single-bit input differences of D-QUARK.

Input difference bit i	Maximum round S_i
148	125
147	124
146	123
34	122
134	122
145	122
⋮	⋮

Imposing Conditions. In [4] Aumasson *et al.* proposed two approaches to control the propagation of differences:

1. Control the propagation of the single-bit input difference in a given state bit s_p as far as possible;
2. Find an input difference of arbitrary weight that leads to a single-bit difference in s_p after q rounds.

By controlling the propagation of s_p both backwards and forwards, they found a distinguisher for C-QUARK up to 396 rounds. In [23], Zhang *et al.* proposed a method to find a suitable s_p, and they could distinguish U-QUARK/D-QUARK/S-QUARK/C-QUARK up to 153/159/248/445 rounds respectively.

In order to further improve the results, we should control the propagation as far as possible. However, it is challenging to impose conditions for more rounds. One of the challenges is that we may introduce much more conditions to convert the intermediate conditions. As for D-QUARK, the length of the register X is 88. Let $q = 10$, suppose we need to impose a condition $X_{75}^{10} = 0$ at round $t = 10$. Since this condition cannot be imposed directly in the initial state, we convert this condition to the equivalent condition $X_{85}^{0} = 0$ by the shifting of register. However, if this condition is $X_{85}^{10} = 0$, to convert it to the equivalent conditions at $t = 0$, a lot more conditions will be added rather than a single condition. Another tricky problem is that the conditions derived by the two approaches may contradict (*e.g.*, the conditions $X_{34}^{0} = 0$ and $X_{34}^{0} = 1$) and cannot be satisfied simultaneously.

Therefore, we should make a tradeoff among the backward round q, the forward round, and the number of conditions. The total rounds should be large enough while the number of conditions should be as small as possible. We follow this rule and apply it to the analysis of D-QUARK.

In Table 4, we have obtained several candidates of single-bit input differences. For each of them, we use the above two approaches to impose conditions and then verify the characteristic by experiments. Our experiments show that choosing the single-bit difference at s_{34} and setting $q = 7$ might be a good choice.

Now we describe the process of imposing conditions. Given the single-bit difference at s_{34} and the backward round $q = 7$, we first consider using the backward computation to derive the input difference which leads to a single-bit difference at s_{34} after $q = 7$ rounds. With the help of SAGE [18], a software on symbolic computation, we can compute the differential expressions in the updated state bits at early rounds of the state update. In Sect. 3, we have discussed about how to compute the backward update functions. When we move backwards, X_0 and Y_0 are updated at each round. Since $q = 7$, we need to compute the differential expressions in the updated state bits X_0^t and Y_0^t for $t \in \{0, \ldots, 6\}$ from the internal state at $t = 7$. Note that we should first compute the state of the register L at $t = 7$ by the linear feedback function. The following conditions will make sure that for the initial 7 rounds, each differential expression in the updated state bit is equal to a constant (0 or 1):

$$X_{56}^{7} Y_{3}^{7} + X_{56}^{7} Y_{78}^{7} = 0,$$

$$X_{9}^{7} X_{17}^{7} X_{25}^{7} X_{40}^{7} + X_{9}^{7} X_{56}^{7} X_{77}^{7} + X_{25}^{7} X_{40}^{7} X_{45}^{7} X_{56}^{7} X_{65}^{7} + X_{25}^{7} X_{40}^{7} + 1 = 0.$$

A sample that satisfies the above two conditions can be generated by fixing the following bits to a constant:

$$\begin{aligned} X_{56}^{7} &= 1, Y_{3}^{7} = 1, X_{9}^{7} = 1, \\ Y_{78}^{7} &= 1, X_{77}^{7} = 1, X_{25}^{7} = 0. \end{aligned} \tag{7}$$

Except for $Y_0^1 = Y_1^0 = s_{89}$, the differential expressions in all other updated bits (*i.e.*, $X_0^i,\ i \in \{0, \ldots, 6\}$ and $Y_0^j,\ j \in \{0, 2, 3, 4, 5, 6\}$) are equal to zero. Note that the single-bit difference $s_{34} = X_{34}^7$ has moved to $X_{41}^0 = s_{41}$. Thus, the initial difference that has differences at s_{41} and s_{89} will boil down to a single-bit difference s_{34} after 7 rounds, with the conditions in Eq. (7) satisfied. Obviously, these conditions can be directly converted to the conditions in the initial state bits:

$$\begin{aligned} X_{63}^0 =& 1, Y_{10}^0 = 1, X_{16}^0 = 1, \\ Y_{85}^0 =& 1, X_{84}^0 = 1, X_{32}^0 = 0. \end{aligned} \tag{8}$$

In the same way, we can impose conditions forwards to restrain the propagation of s_{34}. In this period, 29 state bits are fixed to make sure that for 31 rounds, each differential expression in the updated state bit is equal to a constant (only 4 of them are equal to 1). Therefore, we have fixed $6 + 29 = 35$ initial state bits in total to prevent the propagation of differences.

Verification. Finally, we verify this characteristic by experiments. We first introduce the difference at s_{41} and s_{89}, then impose the 35 conditions on the initial state bits. Other state bits are set to random values. For one pair of input samples, the only difference for sample $\mathcal{A}$ and sample $\mathcal{B}$ is that, in sample $\mathcal{A}$, $s_{41} = s_{89} = 0$, while in sample $\mathcal{B}$, $s_{41} = s_{89} = 1$. Each pair of input samples would give a value of the output difference. We randomly generate 2^{32} pairs of input samples, and observe the bias of the difference in the updated state bit round by round. After 166 rounds, the bias of the difference in the state bit s_{88} is 0.00016. According to Eq. 1 in Sect. 2, this bias is strong enough to distinguish the cipher from random.

5.2 Cryptanalysis Results for All Flavors of QUARK

In the previous subsection, the analysis of D-QUARK has been elaborated. Now we apply these methods to the conditional differential cryptanalysis of other versions of QUARK. Table 5 shows the cryptanalysis results for all flavors of QUARK. The imposed conditions for each version are listed in Appendix B. Our distinguishers are very practical. Take S-QUARK as an example, we can distinguish 259 rounds with a strong bias of 0.00522, while the best previous analysis could only distinguish the cipher up to 248 rounds with a much higher complexity. To the best of our knowledge, all our results for QUARK are the best thus far.

Table 5. Cryptanalysis results for all flavors of QUARK.

Instance	Input difference bits	Output difference bit	#Rounds	Bias	#Pairs of input samples
U-QUARK	$s_1, s_3, s_5, s_{19}, s_{68}, s_{69}, s_{71}, s_{77}, s_{85}, s_{87}, s_{129}$	s_0	155	0.00008	2^{32}
D-QUARK	s_{41}, s_{89}	s_{88}	166	0.00016	2^{32}
S-QUARK	$s_3, s_5, s_{13}, s_{22}, s_{74}, s_{128}, s_{130}, s_{137}, s_{140}, s_{144}, s_{146}, s_{149}$	s_0	259	0.00522	2^{32}
C-QUARK	$s_1, s_5, s_6, s_{12}, s_{14}, s_{16}, s_{18}, s_{25}, s_{29}, s_{31}, s_{32}, s_{33}, s_{34}, s_{37}, s_{39}, s_{41}, s_{66}, s_{100}, s_{192}, s_{198}, s_{202}, s_{203}, s_{205}, s_{213}, s_{215}, s_{217}, s_{219}, s_{221}, s_{223}, s_{225}, s_{229}, s_{230}, s_{233}, s_{235}, s_{241}, s_{245}, s_{249}, s_{254}, s_{255}, s_{266}, s_{268}, s_{270}, s_{271}, s_{272}, s_{274}, s_{276}, s_{278}, s_{280}, s_{282}, s_{284}, s_{286}, s_{288}$	s_0	452	0.00007	2^{33}

Note that the number of pairs of input samples we choose are actually very large to ensure the validity of results. According to Eq. 1 in Sect. 2, about 2^{18} pairs of samples are enough for S-QUARK.

6 Conclusions

In this paper, we proposed a general method to simulate the differential propagation of an iterated cryptosystem. Based on this method, we can obtain a list of good input differences. We applied this method to the conditional differential cryptanalysis of QUARK family. By controlling the propagation of differences both backwards and forwards, we can observe the bias of output difference at a higher round. To the best of our knowledge, all results are the best thus far and have been fully verified by experiments. It is worthy of applying it to the cryptanalysis of other cryptosystems in the future.

Acknowledgments. We are grateful to Ming Li, Xiaojuan Zhang, and anonymous reviewers of IWSEC 2018 for their fruitful discussions and helpful comments.

A Simulating the Bit Operations

See Tables 6 and 7.

Table 6. Simulating the AND operation.

sand	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8
2	0	2	2	6	8	8	6	8	8
3	0	3	6	3	0	6	6	6	6
4	0	4	8	0	4	8	0	4	8
5	0	5	8	6	8	8	6	8	8
6	0	6	6	6	0	6	6	6	6
7	0	7	8	6	4	8	6	7	8
8	0	8	8	6	8	8	6	8	8

Table 7. Simulating the XOR operation.

sxor	0	1	2	3	4	5	6	7	8
0	0	1	2	3	4	5	6	7	8
1	1	0	2	4	3	5	7	6	8
2	2	2	2	5	5	5	8	8	8
3	3	4	5	0	1	2	6	7	8
4	4	3	5	1	0	2	7	6	8
5	5	5	5	2	2	2	8	8	8
6	6	7	8	6	7	8	6	7	8
7	7	6	8	7	6	8	7	6	8
8	8	8	8	8	8	8	8	8	8

B Conditions for All Flavors of QUARK

See Table 8.

Table 8. Conditions for all flavors of QUARK.

Instance	Conditions	
U-QUARK	'0' bits	s_2, s_6, s_8, s_9, s_{10}, s_{11}, s_{15}, s_{22}, s_{23}, s_{24}, s_{28}, s_{29}, s_{33}, s_{36}, s_{38}, s_{40}, s_{43}, s_{47}, s_{48}, s_{50}, s_{51}, s_{52}, s_{53}, s_{55}, s_{56}, s_{57}, s_{60}, s_{61}, s_{62}, s_{70}, s_{72}, s_{75}, s_{80}, s_{86}, s_{89}, s_{91}, s_{93}, s_{95}, s_{97}, s_{101}, s_{103}, s_{106}, s_{108}, s_{112}, s_{116}, s_{117}, s_{122}, s_{125}, s_{131}, s_{132}
	'1' bits	s_{18}, s_{25}, s_{32}, s_{34}, s_{37}, s_{41}, s_{46}, s_{58}, s_{59}, s_{63}, s_{64}, s_{65}, s_{73}, s_{79}, s_{81}, s_{83}, s_{88}, s_{90}, s_{92}, s_{94}, s_{96}, s_{100}, s_{102}, s_{105}, s_{107}, s_{110}, s_{113}, s_{114}, s_{115}, s_{118}, s_{119}, s_{120}, s_{124}, s_{126}, s_{127}, s_{128}, s_{130}, s_{133}, s_{134}
	$s_{74} = s_4 + s_7 + s_{17} + s_{31} + s_{49} + 1$	
D-QUARK	'0' bits	s_{15}, s_{20}, s_{32}, s_{33}, s_{49}, s_{51}, s_{55}, s_{57}, s_{61}, s_{62}, s_{66}, s_{68}, s_{72}, s_{73}, s_{79}, s_{82}, s_{87}, s_{103}, s_{105}, s_{107}, s_{115}, s_{127}, s_{158}
	'1' bits	s_{16}, s_{34}, s_{42}, s_{50}, s_{56}, s_{63}, s_{83}, s_{84}, s_{86}, s_{98}, s_{164}, s_{173}
S-QUARK	'0' bits	s_{38}, s_{49}, s_{52}, s_{58}, s_{61}, s_{62}, s_{63}, s_{66}, s_{73}, s_{76}, s_{80}, s_{82}, s_{85}, s_{88}, s_{99}, s_{101}, s_{102}, s_{108}, s_{110}, s_{119}, s_{159}, s_{187}, s_{189}, s_{192}, s_{241}
	'1' bits	s_{34}, s_{86}, s_{87}, s_{96}, s_{104}, s_{113}, s_{161}, s_{164}
C-QUARK	'0' bits	s_{36}, s_{38}, s_{45}, s_{46}, s_{48}, s_{49}, s_{50}, s_{52}, s_{53}, s_{56}, s_{58}, s_{59}, s_{60}, s_{61}, s_{62}, s_{63}, s_{67}, s_{68}, s_{69}, s_{70}, s_{71}, s_{72}, s_{73}, s_{74}, s_{75}, s_{77}, s_{78}, s_{79}, s_{81}, s_{82}, s_{83}, s_{84}, s_{86}, s_{87}, s_{88}, s_{92}, s_{93}, s_{95}, s_{96}, s_{97}, s_{103}, s_{105}, s_{106}, s_{109}, s_{115}, s_{117}, s_{118}, s_{119}, s_{120}, s_{121}, s_{122}, s_{123}, s_{125}, s_{126}, s_{127}, s_{128}, s_{129}, s_{131}, s_{133}, s_{135}, s_{136}, s_{137}, s_{138}, s_{139}, s_{140}, s_{142}, s_{144}, s_{146}, s_{148}, s_{149}, s_{150}, s_{151}, s_{154}, s_{156}, s_{160}, s_{162}, s_{164}, s_{180}, s_{200}, s_{218}, s_{226}, s_{228}, s_{231}, s_{234}, s_{239}, s_{242}, s_{244}, s_{246}, s_{247}, s_{251}, s_{252}, s_{253}, s_{256}, s_{257}, s_{258}, s_{259}, s_{260}, s_{261}, s_{262}, s_{263}, s_{264}, s_{265}, s_{267}, s_{269}, s_{273}, s_{279}, s_{290}, s_{292}, s_{294}, s_{296}, s_{298}, s_{300}, s_{302}, s_{304}, s_{305}, s_{306}, s_{308}, s_{310}, s_{312}, s_{318}, s_{320}, s_{322}, s_{325}, s_{327}, s_{328}, s_{333}, s_{335}, s_{338}, s_{343}, s_{344}, s_{345}, s_{346}, s_{347}, s_{348}, s_{349}, s_{351}, s_{353}, s_{355}, s_{357}, s_{359}, s_{361}, s_{371}, s_{377}, s_{378}, s_{380}
	'1' bits	s_{43}, s_{54}, s_{55}, s_{57}, s_{64}, s_{65}, s_{76}, s_{80}, s_{91}, s_{101}, s_{114}, s_{130}, s_{132}, s_{134}, s_{204}, s_{214}, s_{220}, s_{232}, s_{236}, s_{238}, s_{240}, s_{248}, s_{250}, s_{281}, s_{283}, s_{285}, s_{287}, s_{289}, s_{291}, s_{293}, s_{295}, s_{297}, s_{299}, s_{301}, s_{303}, s_{307}, s_{314}, s_{316}, s_{332}, s_{334}, s_{342}, s_{352}, s_{368}

References

1. Aumasson, J.P.: Github - veorq/quark: Lightweight cryptographic hash functions (reference code). https://github.com/veorq/Quark/
2. Aumasson, J.-P., Henzen, L., Meier, W., Naya-Plasencia, M.: Quark: a lightweight hash. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 1–15. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15031-9_1
3. Aumasson, J.P., Henzen, L., Meier, W., Naya-Plasencia, M.: Quark: a lightweight hash. J. Cryptol. **26**(2), 313–339 (2013)
4. Aumasson, J.P., Knellwolf, S., Meier, W.: Heavy Quark for secure AEAD. DIAC-Directions in Authenticated Ciphers (2012)
5. Banik, S.: Conditional differential cryptanalysis of 105 round Grain v1. Crypt. Commun. **8**(1), 113–137 (2016)
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_11
7. Biryukov, A., Lamberger, M., Mendel, F., Nikolić, I.: Second-order differential collisions for reduced SHA-256. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 270–287. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_15
8. De Cannière, C.: Trivium: a stream cipher construction inspired by block cipher design principles. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 171–186. Springer, Heidelberg (2006). https://doi.org/10.1007/11836810_13
9. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04138-9_20
10. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_16
11. Dinur, I., Shamir, A.: Breaking Grain-128 with dynamic cube attacks. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21702-9_10
12. Hell, M., Johansson, T., Maximov, A., Meier, W.: The Grain family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 179–190. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68351-3_14
13. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of NLFSR-based cryptosystems. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 130–145. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_8
14. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60590-8_16
15. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (eds.) Communications and Cryptography. SECS, vol. 276, pp. 227–233. Springer, Boston (1994). https://doi.org/10.1007/978-1-4615-2694-0_23

16. Ma, Z., Tian, T., Qi, W.F.: Conditional differential attacks on Grain-128a stream cipher. IET Inf. Secur. **11**(3), 139–145 (2016)
17. Ma, Z., Tian, T., Qi, W.F.: Improved conditional differential attacks on Grain v1. IET Inf. Secur. **11**(1), 46–53 (2016)
18. Stein, W., Joyner, D.: Sage: system for algebra and geometry experimentation. ACM SIGSAM Bull. **39**(2), 61–64 (2005)
19. 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). https://doi.org/10.1007/11535218_2
20. 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). https://doi.org/10.1007/11426639_2
21. Watanabe, Y., Isobe, T., Morii, M.: Conditional differential cryptanalysis for Kreyvium. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 421–434. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_22
22. Watanabe, Y., Todo, Y., Morii, M.: New conditional differential cryptanalysis for NLFSR-based stream ciphers and application to Grain v1. In: 2016 11th Asia Joint Conference on Information Security (AsiaJCIS), pp. 115–123. IEEE (2016)
23. Zhang, K., Guan, J., Fei, X.: Improved conditional differential cryptanalysis. Secur. Commun. Netw. **8**(9), 1801–1811 (2015)

Lightweight Recursive MDS Matrices with Generalized Feistel Network

Qiuping Li[1], Baofeng Wu[2(✉)], and Zhuojun Liu[3]

[1] University of Chinese Academy of Sciences, Beijing, China
[2] State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
wubaofeng@iie.ac.cn
[3] Key Laboratory of Mathematics Mechanization Academy of Mathematics
and Systems Science, Chinese Academy of Sciences, Beijing, China

Abstract. Maximum distance separable (MDS) matrices are often used to construct optimal linear diffusion layers in many block ciphers. With the development of lightweight cryptography, the recursive MDS matrices play as good candidates. The recursive MDS matrices can be computed as powers of sparse matrices. In this paper, we consider searching recursive MDS matrices from Generalized Feistel Structure (GFN) matrices. The advantage of constructing MDS matrices based on GFN matrices mainly displays two aspects. First, the recursive GFN MDS matrix can be implemented by parallel computation that would reduce the running time. Second, the inverse matrix of recursive GFN MDS matrix is also a recursive GFN MDS matrix and they have the same XOR count. We provide some computational experiments to show we do find some lightweight 4×4 and 8×8 recursive GFN MDS matrices over $\mathbb{F}_{2^n}$. Especially, the 8×8 recursive GFN MDS matrices over $\mathbb{F}_{2^8}$ have lower XOR count than the previous recursive MDS matrices.

Keywords: Lightweight MDS matrix · Recursive · XOR count
Generalized Feistel Network

1 Introduction

In the designing of symmetric-key ciphers, there are two fundamental concepts required for the overall security of the cipher, they are the confusion and the diffusion properties described by Shannon [19]. The diffusion layer of a cipher is often achieved by a matrix that transforms an input vector to some output vector through linear operations. The quality of a linear diffusion layer is connected to its branch number. A high branch number implies that changing a single bit of the input will change the output a lot, which is exactly what one expects for a good diffusion layer. Therefore, maximum distance separable (MDS) matrices are quite good choices for the construction of diffusion layers since their branch numbers are maximum. However, different MDS matrices have different implementation costs; not all MDS matrices can lead to efficient hardware or software implementations.

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 262–278, 2018.
https://doi.org/10.1007/978-3-319-97916-8_17

To estimate the hardware implementation cost of an MDS matrix, Khoo *et al.* [13] provide the definition of XOR count, which is called d-XOR count in [12], roughly speaking it is the number of XOR operations needed to perform multiplication of the matrix with any vector. It is expected to find the MDS matrix with as little d-XOR count as possible. After the d-XOR count, Jean *et al.* [12] proposed a new metric called s-XOR count, which can reuse intermediate results to decrease the XOR count and does not need the temporary registers since these are wires between gates. Since the s-XOR count is less than or equal to the d-XOR count, in this paper, we use the s-XOR count as the metric to estimate the hardware implementation cost of an MDS matrix.

To improve efficiency of hardware implementation, a clever method is to obtain MDS matrices by recursive constructions. The recursive MDS matrix can be constructed from a $k \times k$ sparse matrix over $\mathbb{F}_{2^n}$, where n is the size of the Sboxes, such that k iterations of this matrix is an MDS matrix. On the one hand, the recursive MDS matrix can save storage space and on the other hand, it needs to spend more running time. This is a trade-off. The main idea of using recursive matrices to construct MDS matrices was first adopted in the design of lightweight hash function PHOTON [8] and block cipher LED [9]. Afterwards, the definition of recursive MDS matrices was given in [20]. Following the work of [20], some recent papers [1–4,10,14,24,25] are devoted to the construction of recursive MDS matrices. In [20], Sajadieh *et al.* replaced the finite field operations by simpler $\mathbb{F}_2$-linear operations, again improving the implementation efficiency of the construction. The same design strategy was used by Wu *et al.* [24] to obtain optimal diffusion layer, even including bit-level LFSRs case. In [8], the authors constructed lightweight MDS matrices from companion matrices by exhaustive search. In [10], authors developed techniques to test if a given $k \times k$ matrix is an MDS matrix. They also construct lightweight 4×4 and 5×5 MDS matrices over $\mathbb{F}_{2^n}$ for all $n \geq 4$. In [3,4,11], authors construct recursive MDS matrices using the BCH code and Gabidulin code.

These previous works only consider whether the MDS matrix is lightweight or not, however, the XOR count of inverse matrix of the MDS matrix was rarely discussed. In this paper, we introduce some properties of recursive MDS matrix with Generalized Feistel Network (GFN) and construct some lightweight recursive GFN MDS matrices. We find the inverse matrix of a GFN MDS matrix is also a GFN MDS matrix and they have the same XOR counts. In addition, the recursive GFN MDS matrix and its inverse matrix can be implemented by parallel computing that can reduce the running time. We also give some results to reduce the searching space. Computational experiments show that lightweight 4×4 recursive GFN MDS matrices over $\mathbb{F}_{2^n}$ can be obtained. For 8×8 MDS matrix, we find some recursive GFN MDS matrices that outperform previous lightweight recursive MDS matrices over $\mathbb{F}_{2^8}$. They can be used to replace the diffusion layers of previous block ciphers and hash functions to gain better performance.

This paper is organized as follows. In Sect. 2, we give some notations of MDS matrix and XOR count. In Sect. 3, we discuss some properties of Generalized

Feistel Network matrix. In Sect. 4, we give some propositions to reduce the searching scope and find some lightweight recursive GFN MDS matrices. In Sect. 5, we conclude the paper.

2 Preliminaries

Let $\mathbb{F}_2 = \{0, 1\}$ be the binary finite field and $\mathbb{F}_{2^n}$ be the finite field of 2^n elements. $\mathbb{F}_{2^n}$ is isomorphic to polynomials in $\mathbb{F}_2[x]$ modulo an irreducible polynomial $p(x)$ of degree n, namely, each element of $\mathbb{F}_{2^n}$ can be represented as a polynomial of degree less than n over $\mathbb{F}_2$. Suppose $\beta \in \mathbb{F}_{2^n}$, then β can be represented as $\sum_{i=0}^{n-1} b_i \alpha^i$, where $b_i \in \mathbb{F}_2$ and α is a root of a generating polynomial of $\mathbb{F}_{2^n}$. The element β can be viewed as an n-bit string $(b_{n-1}, b_{n-2}, \ldots, b_0)$, traditionally, the n-bit string is represented in a hexadecimal representation, which will be prefixed with 0x. For example, in $\mathbb{F}_{2^8}$, the 8-bit string 01101010 corresponds to the element $\alpha^6 + \alpha^5 + \alpha^3 + \alpha$, written 0x6a in hexadecimal.

The addition operation on $\mathbb{F}_{2^n}$ is simply defined as a bitwise XOR and does not depend on the choice of the irreducible polynomial $p(x)$ of degree n. However, for multiplication, one needs to specify the irreducible polynomial $p(x)$ of degree n. We denote this field as $\mathbb{F}_{2^n}/p(x)$, where $p(x)$ can be given in hexadecimal representation.

2.1 MDS Matrix and Its Properties

Maximum Distance Separable (MDS) matrices are crucial components in cryptographic designs, as they ensure a perfect diffusion layer. In the following, we give the definition of MDS matrix.

Definition 1 (see [7]). *The differential branch number of a $k \times k$ matrix M over $\mathbb{F}_{2^n}$ is*

$$B_d(M) = \min_{\boldsymbol{x} \neq 0} \{w_b(\boldsymbol{x}) + w_b(M\boldsymbol{x})\},$$

and the linear branch number of M is

$$B_\ell(M) = \min_{\boldsymbol{x} \neq 0} \left\{w_b(\boldsymbol{x}) + w_b(M^T\boldsymbol{x})\right\}.$$

where $w_b(\boldsymbol{x})$ is the number of nonzero elements in $\boldsymbol{x}$, and when the optimal value $B_d(M) = B_\ell(M) = k + 1$ is attained, we say M is an MDS matrix.

There are various ways to verify if a matrix is MDS, in this paper we state a common way to identify MDS matrix.

Proposition 1 (see [5]). *A square matrix M is an MDS matrix if and only if all square sub-matrices of M are nonsingular.*

Based on above proposition and the property of the adjoint matrix, we have the following property of MDS matrix.

Proposition 2. *If the matrix M is an MDS matrix, then the matrix M^{-1} is also an MDS matrix.*

The following proposition is a crucial result in this paper.

Proposition 3 (see [16]). *For any permutation matrices P and Q, the branch numbers of M and PMQ are the same.*

2.2 XOR Count

To estimate the hardware implementation cost of a diffusion layer, Khoo *et al.* [13] provide the definition of XOR count(denoted by d-XOR count [12]).

Definition 2 (see [13]). *The d-XOR count of an element β in the field $\mathbb{F}_{2^n}/p(x)$ is the number of XOR operations required to implement the multiplication of β with an arbitrary η over $\mathbb{F}_{2^n}/p(x)$ and denoted by d-XOR(β), where $\eta \in \mathbb{F}_{2^n}/p(x)$.*

After this, Jean *et al.* [12] provided an improved XOR count(denoted by s-XOR count [12]) which can reuse intermediate results to decrease the XOR count and it does not need temporary register since these are wires between gates.

Definition 3 (see [12]). *The s-XOR count of an element β in the field $\mathbb{F}_{2^n}/p(x)$ is the minimum number of XOR operations in the sequence of instructions for implementing the field element multiplication and denoted by s-XOR(β).*

Example 1. Let $\beta = 7$ is the element of $\mathbb{F}_{2^3}/0xb$. Suppose (b_2, b_1, b_0) to be the binary representation of an arbitrary element η in the field $\mathbb{F}_{2^3}/0xb$. We have:

$$\begin{aligned}(1,1,1)\cdot(b_2,b_1,b_0) &= (b_2\oplus b_0, b_2\oplus b_1, b_1)\oplus(b_1, b_2\oplus b_0, b_2)\oplus(b_2,b_1,b_0)\\ &= (b_1\oplus b_0, b_0, b_2\oplus b_1\oplus b_0),\end{aligned}$$

which corresponds to 3 d-XORs. In practice, $b_1 \oplus b_0$ can be reused, so we have the s-XOR(β) = 2 < d-XOR(β).

In 2017, [15] proposed a more efficient way to compute the XOR count of MDS matrices, but this way is not suitable for the recursive MDS matrices. The recursive MDS matrix can be computed as powers of a sparse matrix, so the XOR count of recursive MDS matrix is some multiples of the XOR count of a sparse matrix. In this paper, we use the new metric(s-XOR count) to evaluate the implementation cost.

The XOR count of a matrix $M = (m_{i,j})_{k\times k},\ m_{i,j} \in \mathbb{F}_{2^n}$, is defined as [22].

$$XOR(M) = \sum_{i,j=1}^{k} XOR(m_{i,j}) + \sum_{i=1}^{k}(r_i - 1)n, \tag{1}$$

where r_i is the number of nonzero elements in the i-th row.

3 Generalized Feistel Network Matrix

In this section, we firstly introduce some properties of previous recursive MDS matrices and their XOR count formulas. After that we talk about the Generalized Feistel Network matrix and its properties about MDS matrix.

3.1 Recursive MDS Matrix

In [8], the authors propose the idea of constructing recursive MDS matrix with companion matrix. Using the notation of [8], we define the companion matrix of $f(x) = z_1 + z_2x + \cdots + z_kx^{k-1} + x^k$ as

$$C = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ z_1 & z_2 & z_3 & \cdots & z_k \end{bmatrix}_{k \times k},$$

where $z_i \in \mathbb{F}_{2^n}, i = 1, 2, \ldots, k$.

From the formula (1), the XOR count of implementing the matrix C is

$$(k-1) \cdot n + \sum_{i=1}^{k} XOR(z_i).$$

The recursive MDS matrix $M = C^k$ can be implemented by Linear Feedback Shift Register (LFSR, Fig. 1). So the XOR count of recursive MDS matrix with companion matrix is

$$k[(k-1) \cdot n + \sum_{i=1}^{k} XOR(z_i)]. \tag{2}$$

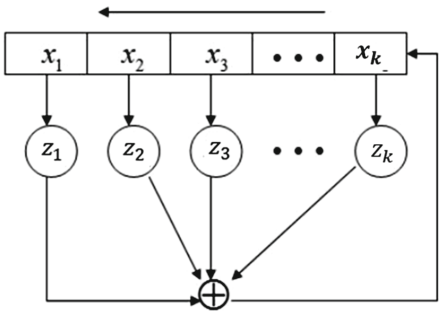


Fig. 1. Linear Feedback Shift Register.

In [24], the authors propose to construct recursive MDS matrices with Type-II Generalized Feistel Structure (GFS, Fig. 2). This Type-II Generalized Feistel Structure is a extension of the Type-II Generalized Feistel Network. Using the notation of [24], we define the matrix representation of Type-II Generalized Feistel Structure (GFS) as follows.

$$A_{gfs2}[z_1, z_2, \ldots, z_k] = \begin{bmatrix} T & U_2 & 0 & \cdots & 0 & 0 \\ 0 & T & U_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & T & U_{\frac{k}{2}} \\ U_1 & 0 & 0 & \cdots & 0 & T \end{bmatrix}_{k \times k}$$

where "0" is a 2×2 zero matrix over $\mathbb{F}_{2^n}$ while $T = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ and $U_i = \begin{bmatrix} 0 & 0 \\ z_{2i-1} & z_{2i} \end{bmatrix}$ $1 \leq i \leq \frac{k}{2}$ are 2×2 block matrices, $z_1, z_2, z_3, \ldots, z_k \in \mathbb{F}_{2^n}$ for some n.

From the formula (1), the XOR counts of implementing the matrix is

$$\frac{k}{2} \cdot n + \sum_{i=1}^{k} XOR(z_i).$$

Similarly, the XOR counts of recursive MDS matrix with Type-II GFS is

$$k[\frac{k}{2} \cdot n + \sum_{i=1}^{k} XOR(z_i)]. \tag{3}$$

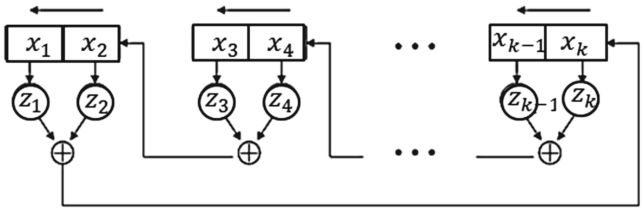


Fig. 2. Type-II Generalized Feistel Structure.

From Fig. 1, we notice the LFSR can not be implemented by parallel computation to decrease the running time. From the matrix A_{gfs2}, the inverse matrix of Type-II Generalized Feistel Structure (GFS) matrix is not always Type-II GFS matrix, so the inverse matrix of the recursive Type-II GFS MDS matrix is not always implemented by Type-II GFS. In the following, we will use the general definition matrix of the Generalized Feistel Network to construct recursive MDS matrices.

3.2 Generalized Feistel Network Matrix

In this paper, we construct recursive MDS matrix with Generalized Feistel Network (GFN) matrix. We hope the GFN has lesser XOR count and can be implemented by parallel computation so that the GFN can save the hardware implement cost and reduce the running time. From [6], we have the following definition of GFN matrix.

Definition 4 (see [6]). *A $k \times k$ matrix B with coefficients in $\{0, 1, z_1, \ldots, z_t\}$ is a Generalized Feistel Network (GFN) matrix if it can be written as $B = PF$ such that P is a permutation matrix and the matrix F satisfies the following conditions:*

(1) the main diagonal is filled with 1,
(2) the off-diagonal coefficients are either 0 or z_i,
(3) for each index $i (1 \leq i \leq k)$, if row(column) i has an z_j coefficient then column(row) i just has 0 coefficient except the main diagonal is 1, where $z_j \in \mathbb{F}_{2^n}, j = 1, 2, \ldots, t$.

Example 2. The following matrix is F of a Generalized Feistel Network.

$$F = \begin{bmatrix} 1 & z_1 & z_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & z_3 & 0 & 1 \end{bmatrix}$$

From this example, we can see that the column 2 and 3 have the coefficient z_1 and z_2, so the row 2 and 3 just have 0 coefficient except the main diagonal is 1.

Example 3. If the 4×4 Type-II Generalized Feistel Structure matrix $A_{gfs2} = PF$, then $P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$ and $F = \begin{bmatrix} z_1 & z_2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & z_3 & z_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. We can see when $z_1 = z_3 = 1$, the Type-II GFS satisfies the conditions of GFN matrix. So the Type-II GFS MDS matrices have not always the advantage of GFN MDS matrices.

According to [6], we easily have the following theorem.

Theorem 1 (see [6]). *Let $B = PF$ be a Generalized Feistel Network (GFN) matrix according to Definition 4. Then F is an involution matrix.*

Since F is involution matrix, we naturally want to know whether $B^k = (PF)^k$ is involution matrix or not. Next, we will talk about this problem. The following lemma is a well-known property of permutation matrix.

Lemma 1. *Let P be a permutation matrix. Then $P^{-1} = P^T$.*

From the definition of GFN matrix, the non-linear layer F has the following property.

Lemma 2. *Let $B = PF$ be a Generalized Feistel Network (GFN) matrix from Definition 4 and $F' = QFQ^{-1}$, where Q is a permutation matrix. Then F' satisfies the three conditions of Definition ??.*

Proof. Let $f_{i,j}$ denote the coefficient of F at row i and column j, $f'_{l,q}$ denote the coefficient of $F^{'}$ at row l and colutm q, σ is the permutation representation of the permutation matrix Q. From Lemma 1, we have $Q^{-1} = Q^T$. Since Q^{-1} transform the column of the matrix F and $Q^{TT} = Q$, we have σ is the permutation representation of the permutation matrix Q^{-1}. So we have $f'_{l,q} = f_{\sigma(i),\sigma(j)}$. Then

(1) when $i = j$, we have $f_{i,i} = 1$. So $f_{\sigma(i),\sigma(i)} = 1$.
(2) when $i \neq j$, we have $f_{i,j}$ is either 0 or $z_s (1 \leq s \leq t)$ and $\sigma(i) \neq \sigma(j)$. So $f_{\sigma(i),\sigma(j)}$ is either 0 or $z_s (1 \leq s \leq t)$.
(3) when $i \neq j$ and $f_{i,j} = z_s (1 \leq s \leq t)$, we have $\sigma(i) \neq \sigma(j)$ and $f_{p,i} = f_{j,l} = 0$, for all $p \neq i, l \neq j$. So when $\sigma(i) \neq \sigma(j)$ and $f_{\sigma(i),\sigma(j)} = z_s (1 \leq s \leq t)$, we have $f_{\sigma(p),\sigma(i)} = f_{\sigma(j),\sigma(l)} = 0$, for all $\sigma(p) \neq \sigma(i), \sigma(l) \neq \sigma(j)$.

So the F' satisfies the three conditions of Definition 4.

Depended on above theorem and lemma, it easily has the following theorem.

Theorem 2. *Let $B = PF$ be a Generalized Feistel Network (GFN) matrix from Definition 4 and $M = B^k$ is MDS matrix. Then $B^{-1} = FP^{-1} = P^{-1}F'$ is also a GFN matrix and $M^{-1} = (B^{-1})^k$ is also MDS matrix, where $F' = PFP^{-1}$.*

Although this theorem is simple, it brings us many benefits. The inverse matrix of a permutation matrix is naturally a permutation matrix. Applying above theorem, we have that the inverse matrix of a recursive MDS matrix with GFN matrix can also be implemented by GFN and the XOR count of the recursive MDS matrix is the same with its inverse matrix.

4 MDS Properties of Generalized Feistel Network

In this section, we firstly provide some propositions to reduce the searching scope, and then give some lightweight MDS matrices. Based on Proposition 1, if the entries of matrix B^k have zero element, then B^k is not an MDS matrix. Applying this judgement, some MDS properties of recursive GFN matrices will be discussed. Depended on these properties, some lightweight recursive GFN MDS matrices are found.

4.1 Some Propositions of $k \times k$ Recursive GFN MDS Matrices

Firstly, we give a normalized form of $B^m = \left[P(I + \sum_{s=1}^{t} E_{i_s,j_s})\right]^m$.

Proposition 4. *If* $B = P(I + \sum_{s=1}^{t} E_{i_s,j_s})$ *is a* $k \times k$ *matrix, then* $B^m = P\left[\prod_{l=0}^{m-1}(I + \sum_{s=1}^{t} E_{\sigma^l(i_s),\sigma^l(j_s)})\right] P^{m-1}$, *where* P *is a permutation matrix,* σ *is the permutation representation of the row transform matrix* P *and the matrix* $E_{i,j}$ *is consisted of all zeros except in the* i*-th row of the* j*-th column.*

Proof. Since σ is the permutation representation of the row transform matrix P, we have the σ also represents the column transform matrix P^{-1}. So

$$P(I + \sum_{s=1}^{t} E_{i_s,j_s}) = (I + \sum_{s=1}^{t} E_{\sigma(i_s),j_s} P^{-1})P = (I + \sum_{s=1}^{t} E_{\sigma(i_s),\sigma(j_s)})P.$$

Then

$$\begin{aligned}
B^m &= \left[P(I + \sum_{s=1}^{t} E_{i_s,j_s})\right]^m \\
&= P(I + \sum_{s=1}^{t} E_{i_s,j_s})P(I + \sum_{s=1}^{t} E_{i_s,j_s}) \cdots P(I + \sum_{s=1}^{t} E_{i_s,j_s}) \\
&= P(I + \sum_{s=1}^{t} E_{i_s,j_s})(I + \sum_{s=1}^{t} E_{\sigma(i_s),\sigma(j_s)})P^2(I + \sum_{s=1}^{t} E_{i_s,j_s}) \cdots P(I + \sum_{s=1}^{t} E_{i_s,j_s}) \\
&= P(I + \sum_{s=1}^{t} E_{i_s,j_s})(I + \sum_{s=1}^{t} E_{\sigma(i_s),\sigma(j_s)}) \cdots (I + \sum_{s=1}^{t} E_{\sigma^{m-1}(i_s),\sigma^{m-1}(j_s)})P^{m-1} \\
&= P\left[\prod_{l=0}^{m-1}(I + \sum_{s=1}^{t} E_{\sigma^l(i_s),\sigma^l(j_s)})\right] P^{m-1}.
\end{aligned}$$

Nextly, we want to reduce the searching scope of GFN matrix $B = PF$, it can be considered from two aspects. On the one hand, it can simplify the matrix F. On the other hand, it can exclude some impossible permutation matrices. Firstly, we will simplify the matrix F.

Definition 5 (see [6]). *Two GFNs matrices* B *and* B' *are equivalent if there exists a permutation matrix* Q *of the* k *blocks such that* $B' = QBQ^{-1}$.

Remark 1. From the condition (3) of the definition of GFN matrix and Definition 5, we just need to search the matrix form

$$F = \begin{bmatrix} I & 0 \\ A & I \end{bmatrix}_{k \times k},$$

where A is a $\lceil \frac{k}{2} \rceil \times \lfloor \frac{k}{2} \rfloor$ matrix.

From the condition (3) of the definition of GFN matrix, if row(column) i has an z_j coefficient then column(row) i just has 0 coefficient except the main diagonal is 1. Therefore, we have all row(column) index of the nonzero elements $z_j (1 \leq j \leq t)$ of F accounts for half of the order of the matrix F. Based on Definition 5, there

exist a permutation matrix Q_2 such that the row index of the nonzero elements $z_j(1 \le j \le t)$ of the $Q_2FQ_2^{-1}$ is greater than or equal $\frac{k}{2}$. We also have the permutation matrix Q_3 such that the column index of the nonzero elements $z_j(1 \le j \le t)$ of $Q_2FQ_2^{-1}Q_3^{-1}$ is lesser than or equal $\frac{k}{2}$. So we just need search F is the nonzero elements $z_j(1 \le j \le t)$ belong to lower left quarter of F, i.e. we just need to search the matrix

$$F = \begin{bmatrix} I & 0 \\ A & I \end{bmatrix}_{k \times k},$$

where A is a $\lceil \frac{k}{2} \rceil \times \lfloor \frac{k}{2} \rfloor$ matrix.

In addition, we also find when all the coefficients $z_s \in \mathbb{F}_{2^n}$ are in the first column or in the last row of the matrix F, the matrix $B^k = (PF)^k$ is not an MDS matrix.

Proposition 5. *If $B = PF = P(I + \sum_{s=1}^{t} E_{k,s})$ is a $k \times k$ GFN matrix, then $B^k (k > 2)$ is not an MDS matrix.*

Proof. It is easy to see that

$$E_{i_s,j_s} E_{i_{s'},j_{s'}} = \begin{cases} 0 & i_{s'} \neq j_s \\ E_{i_s,j_{s'}} & i_{s'} = j_s \end{cases} \tag{4}$$

Let σ be the permutation representation of the row transform matrix P. Then σ also represent the column transform matrix P^{-1}. From Proposition 4, we have

$$B^k = P \left[\prod_{l=0}^{k-1} (I + \sum_{s=1}^{t} E_{\sigma^l(k),\sigma^l(s)}) \right] P^{k-1},$$

So the row index of B^k is $k, \sigma(k), \sigma^2(k), \ldots, \sigma^{k-1}(k)$.

(a) If the length of the cycle $\sigma(k)$ is less than k, then the B^k must have a row that just has one nonzero element be from P^k. So the B^k is not MDS matrix.
(b) If the length of the cycle $\sigma(k)$ is k. From formula (1), we know the row index $\sigma^{k-1}(k)$ will be eliminated by $E_{\sigma^l(k),\sigma^l(s)} (l = 0, \ldots, k-2)$. Then the $\sigma^{k-1}(k)-$th row just has $E_{\sigma^{k-1}(k),\sigma^{k-1}(s)} (s = 1, \ldots, t)$ and 1 that are the non-zero elements. From Remark 1, we have $t \le \frac{k}{2}$. If B^k is MDS matrix, then $\frac{k}{2} + 1 \ge k, k \le 2$. So the $B^k (k > 2)$ is not MDS matrix.

From above conditions, we know $B^k (k > 2)$ is not MDS matrix.

Similarly, we have the following proposition.

Proposition 6. *If $B = PF = P(I + \sum_{s=1}^{t} E_{s,1})$ is a $k \times k$ GFN matrix, then $B^k (k > 2)$ is not MDS matrix.*

In the following, we give a proposition to exclude some impossible permutation matrices. It is similar to Theorem 7 of [6], we can have some MDS properties of GFN matrix.

Proposition 7. *If $B = PF$ is a $k \times k$ GFN matrix, $F = \begin{bmatrix} I & 0 \\ A & I \end{bmatrix}_{k\times k}$ and $P = \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}_{k\times k}$, then B^k is not an MDS matrix, where P, P_1, P_2 are permutation matrices and A is a $\lceil \frac{k}{2} \rceil \times \lfloor \frac{k}{2} \rfloor$ matrix.*

Proof. Since

$$B^2 = (PF)^2 = \begin{bmatrix} P_1 & 0 \\ P_2A & P_2 \end{bmatrix} \begin{bmatrix} P_1 & 0 \\ P_2A & P_2 \end{bmatrix} = \begin{bmatrix} P_1^2 & 0 \\ P_2AP_1 + P_2^2A & P_2^2 \end{bmatrix},$$

we know the upper right quarter of B^k is zero matrix. So the B^k is not an MDS matrix.

From the above preliminary judge, the searching scope can be reduced. In the next subsection we will search some lightweight recursive GFN MDS matrices.

4.2 Some Lightweight Recursive GFN MDS Matrices

In this subsection, we search some lightweight $k \times k$ recursive GFN MDS matrices over $\mathbb{F}_{2^n}$, where $k = 4, 8$ and n is arbitrary integer.

It is natural that we want to find lightweight $k \times k$ recursive GFN MDS matrix, then we need the t as small as possible for

$$B^k = P\left[\prod_{l=0}^{k-1}(I + \sum_{s=1}^{t} E_{\sigma^l(s),\sigma^l(1)})\right]P^{k-1}.$$

When $k = 4$, we first consider the situations of $t = 1$ and $t = 2$. In these situations, we need to consider $4! \times 12 + 4! \times 12 \times 9 = 2880$ GFN matrices. From Remark 1, we just need to consider the situations of $B = P(I + E_{4,1})$, $B = P(I + E_{4,1} + E_{4,2})$, $B = P(I + E_{4,1} + E_{3,1})$, $B = P(I + E_{4,1} + E_{3,2})$, where the matrix $E_{i,j}$ is consisted of all zeros except in the $i-$th row and $j-$th column is $z_s \in \mathbb{F}_{2^n}$. According to Propositions 5 and 6, we only need to consider the situation of $B = P(I + E_{4,1} + E_{3,2})$. Based on Proposition 7, we finally need to consider $4! - (2 \times 2) = 20$ GFN matrices. The searching result is as follows.

Lemma 3. *Let $B = \begin{bmatrix} z_1 & 0 & 0 & 1 \\ 0 & z_2 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ be defined over $\mathbb{F}_{2^n}$. Then B^4 is an MDS matrix, if*

$$0 \neq z_1 \neq z_2 \neq 0, \quad z_1^4 + 1 \neq 0, \quad z_2^4 + 1 \neq 0,$$
$$z_1^2 + z_1z_2 + z_2^2 \neq 0,\ z_1^2z_2^2 + 1 \neq 0,\ z_1^4z_2^4 + z_1^2z_2^2 + 1 \neq 0,$$
$$z_1^2z_2^3 + z_1 + z_2 \neq 0,\ z_1z_2^3 + 1 \neq 0,\ z_1^2 + z_2^2 + z_1^3z_2^3 \neq 0,$$
$$z_1^3z_2^2 + z_1 + z_2 \neq 0,\ z_2z_1^3 + 1 \neq 0,\ z_1^2 + z_2^2 + z_1^3z_2^3 + z_1z_2 \neq 0.$$

Proof. It can be computed that

$$B^4 = \begin{bmatrix} z_1^4+1 & z_1^2+z_1z_2+z_2^2 & z_1+z_2 & z_1^3 \\ z_1^2+z_1z_2+z_2^2 & z_2^4+1 & z_2^3 & z_1+z_2 \\ z_1^3 & z_1+z_2 & 1 & z_1^2 \\ z_1+z_2 & z_2^3 & z_2^2 & 1 \end{bmatrix}$$

From Proposition 1, we need that all the minors of B^4 are non-zero, then we have

$B^4[1,4] = z_1^3 \neq 0$ $B^4[1,2] = z_1^2+z_1z_2+z_2^2 \neq 0$ $B^4[1,1] = z_1^4+1 = (z_1+1)^4 \neq 0$
$B^4[2,3] = z_2^3 \neq 0$ $B^4[2,2] = z_2^4+1 = (z_2+1)^4 \neq 0$ $B^4[1,3] = z_1+z_2 \neq 0$

It is easy to check that the number of 2×2 minors of B^4 is 36. There are 18 different polynomials in the following list.

z_1^2	z_2^2	$z_1^4z_2^3+z_1^3$	$z_1^2z_2^2+1$	$z_1^4z_2^2+z_1^2$	$z_1^3z_2^2+z_1+z_2$
$z_1^2z_2$	z_1z_2	$z_1^3z_2^4+z_2^3$	$z_1z_2^3+1$	$z_1^3z_2^3+z_1^2+z_2^2$	$z_1^4z_2^4+z_1^2z_2^2+1$
$z_1z_2^2$	$z_1^2z_2^2$	$z_1^2z_2^4+z_2^2$	$z_2z_1^3+1$	$z_1^2z_2^3+z_1+z_2$	$z_1^3z_2^3+z_1^2+z_1z_2+z_2^2$

Then we have

$z_1^2z_2^2+1 = (z_1z_2+1)^2 \neq 0$ $z_1z_2^3+1 \neq 0$ $z_2z_1^3+1 \neq 0$
$z_1^3z_2^3+z_1^2+z_2^2 \neq 0$ $z_1^3z_2^2+z_1+z_2 \neq 0$ $z_1^2z_2^3+z_1+z_2 \neq 0$
$z_1^3z_2^3+z_1^2+z_1z_2+z_2^2 \neq 0$ $z_1^4z_2^4+z_1^2z_2^2+1 = (z_1^2z_2^2+z_1z_2+1)^2 \neq 0$

We can also compute that the number of 3×3 minors of B^4 is 16. There are 8 different polynomials in the following list.

1 z_1^2 z_2^2 z_1z_2 $z_1z_2^2$ $z_1^2z_2$ z_1+z_2 $z_1^2z_2^2+1$

So B^4 is an MDS matrix, if

$0 \neq z_1 \neq z_2 \neq 0$, $z_1+1 \neq 0$, $z_2+1 \neq 0$,
$z_1^2+z_1z_2+z_2^2 \neq 0$, $z_1z_2+1 \neq 0$, $z_1^2z_2^2+z_1z_2+1 \neq 0$,
$z_1^2z_2^3+z_1+z_2 \neq 0$, $z_1z_2^3+1 \neq 0$, $z_1^2+z_2^2+z_1^3z_2^3 \neq 0$,
$z_1^3z_2^2+z_1+z_2 \neq 0$, $z_2z_1^3+1 \neq 0$, $z_1^2+z_2^2+z_1^3z_2^3+z_1z_2 \neq 0$.

From Lemma 3, we can implement the GFN matrix of Lemma 3 and its inverse matrix by Figs. 3 and 4.

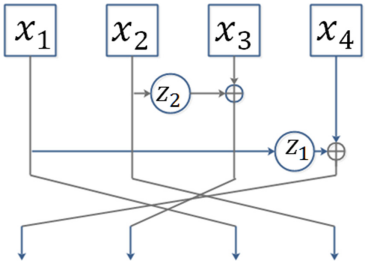


Fig. 3. Generalized Feistel Network

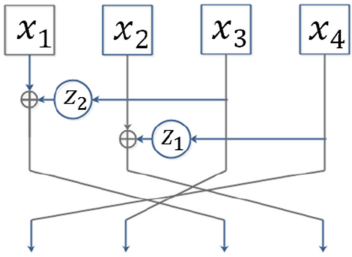


Fig. 4. Inverse structure of Generalized Feistel Network

It is obviously that the GFN MDS matrix only needs to reuse the existing memory with neither temporary storage nor additional control logic required, these properties are the same with Type-II GFS and LFSR. Furthermore, the inverse matrix of the GFN MDS matrix has the same XOR count and they can be implemented by parallel computing, but the inverse matrices of Type-II GFS and LFSR do not have these properties.

From Lemma 3, we know z_1, z_2 are not 0, 1 and $z_1 z_2 \neq 1$. By using the methods of table lookups, the following corollaries propose some lightweight 4×4 recursive GFN MDS matrices.

Corollary 1. *Let* $B = \begin{bmatrix} \alpha & 0 & 0 & 1 \\ 0 & \alpha^2 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ *be defined over* $\mathbb{F}_{2^n}$ *and* $n > 1$, *where* α *is a root of generating polynomial of* $\mathbb{F}_{2^n}$. *Then* B^4 *is an MDS matrix for all* $n \geq 4$ *except* α *is a root of any of the following polynomials*

$$x^4 + x^3 + x^2 + x + 1 = 0,\ x^4 + x^3 + 1 = 0,\ x^6 + x + 1 = 0,$$
$$x^5 + x^4 + x^2 + x + 1 = 0,\ x^6 + x^3 + 1 = 0,\ x^7 + x + 1 = 0.$$

Proof. From Lemma 3, if the α satisfies the following polynomials then B^4 is an MDS matrix.

$0 \neq \alpha \neq \alpha^2 \neq 0$,
$\alpha + 1 \neq 0$,
$\alpha^2 + 1 = (\alpha + 1)^2 \neq 0$,
$\alpha^2 + \alpha\alpha^2 + \alpha^4 = \alpha^2(\alpha^2 + \alpha + 1) \neq 0$,
$\alpha\alpha^2 + 1 = (\alpha + 1)(\alpha^2 + \alpha + 1) \neq 0$,
$\alpha^2\alpha^4 + \alpha\alpha^2 + 1 = \alpha^6 + \alpha^3 + 1 \neq 0$,
$\alpha^2\alpha^6 + \alpha + \alpha^2 = \alpha(\alpha^7 + \alpha + 1) \neq 0$,
$\alpha\alpha^6 + 1 = (\alpha + 1)(\alpha^3 + \alpha + 1)(\alpha^3 + \alpha^2 + 1) \neq 0$,
$\alpha^2 + \alpha^4 + \alpha^3\alpha^6 = (\alpha^2 + \alpha + 1)(\alpha^5 + \alpha^4 + \alpha^2 + \alpha + 1) \neq 0$,
$\alpha^3\alpha^4 + \alpha + \alpha^2 = \alpha(\alpha^6 + \alpha + 1) \neq 0$,
$\alpha^2\alpha^3 + 1 = (\alpha + 1)(\alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1) \neq 0$,
$\alpha^2 + \alpha^4 + \alpha^3\alpha^6 + \alpha\alpha^2 = (\alpha^4 + \alpha^3 + 1)(\alpha + 1)^3 \neq 0$,

That is to say that α is not a root of any of the following polynomials.

$x = 0$, $x + 1 = 0$, $x^4 + x^3 + x^2 + x + 1 = 0$, $x^4 + x^3 + 1 = 0$,
$x^2 + x + 1 = 0$, $x^3 + x + 1 = 0$, $x^5 + x^4 + x^2 + x + 1 = 0$, $x^6 + x^3 + 1 = 0$,
$x^3 + x^2 + 1 = 0$, $x^6 + x + 1 = 0$, $x^7 + x + 1 = 0$.

So B^4 is an MDS matrix for all $n \geq 4$ and α is not a root of any of the following polynomials.

$x^4 + x^3 + x^2 + x + 1 = 0,\ x^4 + x^3 + 1 = 0,\ x^6 + x + 1 = 0,$
$x^5 + x^4 + x^2 + x + 1 = 0,\ x^6 + x^3 + 1 = 0,\ x^7 + x + 1 = 0.$

Similarly, we have the following corollary.

Corollary 2. *Let* $B = \begin{bmatrix} \alpha^{-1} & 0 & 0 & 1 \\ 0 & \alpha^2 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ *be defined over* $\mathbb{F}_{2^n}$ *and* $n > 1$*, where* α *is a root of generating polynomial of* $\mathbb{F}_{2^n}$*. Then* B^4 *is an MDS matrix for all* $n \geq 4$ *except* α *is a root of any of the following polynomials*

$$x^4 + x^3 + x^2 + x + 1 = 0, \quad x^5 + x^3 + 1 = 0, \quad x^6 + x^5 + 1 = 0, \quad x^6 + x^3 + 1 = 0.$$

We applied above corollaries to construct recursive MDS matrices. We found some recursive MDS matrices with fewer XOR count than the previous MDS matrices and list them in following table.

Table 1. Comparison of the XOR count of 4×4 MDS matrices.

Field/Ring	Reference	Matrix type	XORs	XORs of inverse matrix
$\mathbb{F}_{2^4}$/0x13	Corollary 1	GFN	$4 \times (3 + 2 \times 4) = 44$	$4 \times (3 + 2 \times 4) = 44$
$\mathbb{F}_{2^4}$/0x13	[13]	LFSR	$4 \times (3 + 3 \times 4) = 60$	$4 \times (3 + 3 \times 4) = 60$
$GL(4, \mathbb{F}_2)$	[24]	Type-II GFS	$4 \times (2 + 2 \times 4) = 40$	$4 \times (3 + 2 \times 4) = 44$
$\mathbb{F}_{2^8}$/0x1c3	Corollary 1	GFN	$4 \times (8 + 2 \times 8) = 96$	$4 \times (8 + 2 \times 8) = 96$
$\mathbb{F}_{2^8}$/0x11d	[13]	LFSR	$4 \times (9 + 3 \times 8) = 132$	$4 \times (9 + 3 \times 8) = 132$
$GL(8, \mathbb{F}_2)$	[24]	Type-II GFS	$4 \times (2 + 2 \times 8) = 72$	$4 \times (3 + 2 \times 8) = 76$

In Table 1, we compare the XOR count of 4×4 MDS matrices over $\mathbb{F}_{2^4}$ and $\mathbb{F}_{2^8}$. Although, the XOR count of our construction is larger than Type-II GFS MDS matrix, the inverse matrix of Type-II GFS matrix is not a Type-II GFS matrix and their XOR counts are different. Similarly, we also search some lightweight 8×8 recursive MDS matrices from GFN matrices.

Corollary 3. *If* $B = \begin{bmatrix} 0 & 0 & \alpha & \alpha & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ \alpha^2 & \alpha & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$*, then the* B^8 *is MDS matrix for infinitely many* n *and* $n > 7$*, where* α *is a root of generating polynomial of* $\mathbb{F}_{2^n}$*.*

Remark 2. When we prove above result, we find that all minors of B^8 contain finitely many irreducible polynomial of degree larger than 7. However there are infinitely many irreducible polynomial of degree larger than 7 in finite field. So we can find B^8 is MDS matrix for infinitely many n and $n > 7$.

Table 2. Comparison of the XOR count of 8×8 MDS matrices.

Field/Ring	Reference	Matrix type	XORs	XORs of its inverse matrix
$\mathbb{F}_{2^8}/0x187$	Corollary 3	GFN	$8 \times (14 + 6 \times 8) = 496$	$8 \times (14 + 6 \times 8) = 496$
$[\mathbb{F}_{2^4}/0x13]^2$	[23]	LFSR	$8 \times (2 \times 36) = 576$	$8 \times (2 \times 36) = 576$
$GL(8, \mathbb{F}_2)$	[24]	LFSR	$8 \times (9 + 7 \times 8) = 520$	$8 \times (9 + 7 \times 8) = 520$

From above corollary, we found some 8×8 recursive MDS matrices with fewer XOR counts than the previous MDS matrices and list them in the following table.

From Table 2, we compare the XOR counts of 8×8 MDS matrices over $\mathbb{F}_{2^8}$. We find the recursive MDS matrix with GFN and its inverse matrix are more lightweight than previous recursive MDS matrices.

5 Conclusion

In this paper, we search the recursive MDS matrices from GFN matrices. The GFN matrix and its inverse matrix can be implemented by parallel computation to decrease the running time and have the same XOR count. Computational experiments show that lightweight recursive MDS matrices can be obtained from GFN matrices, in particular, the 8×8 recursive GFN MDS matrices are more lightweight than previous recursive MDS matrices. Since these 8×8 recursive GFN MDS matrices are consistent with 8-bit Sboxes, they can be used to replace the diffusion layers in some lightweight block ciphers and lightweight hash functions and have less hardware implementation cost.

References

1. Augot, D., Finiasz, M.: Exhaustive search for small dimension recursive MDS diffusion layers for block ciphers and hash functions. In: Proceedings of 2013 IEEE International Symposium on Information Theory (ISIT), pp. C1551–C1555. IEEE (2013)
2. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçın, T.: Block ciphers – focus on the linear layer (feat. PRIDE). In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 57–76. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_4
3. Augot, D., Finiasz, M.: Direct construction of recursive MDS diffusion layers using shortened BCH codes. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 3–17. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46706-0_1
4. Berger, T.P.: Construction of recursive MDS diffusion layers from gabidulin codes. In: Paul, G., Vaudenay, S. (eds.) INDOCRYPT 2013. LNCS, vol. 8250, pp. 274–285. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03515-4_18
5. Blaum, M., Roth, R.M.: On lowest density MDS codes. IEEE Trans. Inf. Theory **45**(1), 46–59 (1999)

6. Berger, T.P., Minier, M., Thomas, G.: Extended generalized feistel networks using matrix representation. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 289–305. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43414-7_15
7. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-662-04722-4
8. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON family of lightweight hash functions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 222–239. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_13
9. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_22
10. Gupta, K.C., Ray, I.G.: On constructions of MDS matrices from companion matrices for lightweight cryptography. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.) CD-ARES 2013. LNCS, vol. 8128, pp. 29–43. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40588-4_3
11. Gupta, K.C., Pandey, S.K., Venkateswarlu, A.: On the direct construction of recursive MDS matrices. Des. Codes Crypt. **82**(1), 77–94 (2017)
12. Jean, J., Peyrin, T., Sim, S.M.: Optimizing implementations of lightweight building blocks. IACR Trans. Symmetric Cryptol. **2017**(4), 130–168 (2017)
13. Khoo, K., Peyrin, T., Poschmann, A.Y., Yap, H.: FOAM: searching for hardware-optimal SPN structures and components with a fair comparison. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 433–450. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44709-3_24
14. Kolay, S., Mukhopadhyay, D.: Lightweight diffusion layer from the kth root of the MDS matrix. IACR Cryptology ePrint Archive 2014, 498 (2014)
15. Kranz, T., Leander, G., Stoffelen, K., Wiemer, F.: Shorter linear straight-line programs for MDS matrices yet another XOR count paper. IACR Trans. Symmetric Cryptol. **2017**(4), 188–211 (2017). https://doi.org/10.13154/tosc.v2017.i4.188-211
16. Liu, M., Sim, S.M.: Lightweight MDS generalized circulant matrices. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 101–120. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_6
17. Li, Y., Wang, M.: On the construction of lightweight circulant involutory MDS matrices. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 121–139. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_7
18. Li, C., Wang, Q.: Design of lightweight linear diffusion layers from near-MDS matrices. IACR Trans. Symmetric Cryptol. **2017**(1), 129–155 (2017)
19. Shannon, C.E.: Communication theory of secrecy systems. Bell Labs Tech. J. **28**(4), 656–715 (1949)
20. Sajadieh, M., Dakhilalian, M., Mala, H., Sepehrdad, P.: Recursive diffusion layers for block ciphers and hash functions. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 385–401. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34047-5_22
21. Sim, S.M., Khoo, K., Oggier, F., Peyrin, T.: Lightweight MDS involution matrices. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 471–493. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48116-5_23
22. Sarkar, S., Syed, H.: Lightweight diffusion layer: importance of toeplitz matrices. IACR Trans. Symmetric Cryptol. **2016**(1), 95–113 (2016)
23. Toh, D., Teo, J., Khoo, K., Sim, S.M.: Lightweight MDS serial-type matrices with minimal fixed XOR count. IACR Cryptology ePrint Archive 2017, 1084 (2017)

24. Wu, S., Wang, M., Wu, W.: Recursive diffusion layers for (lightweight) block ciphers and hash functions. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 355–371. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35999-6_23
25. Zhao, R., Zhang, R., Li, Y., Wu, B.: On constructions of a sort of MDS block diffusion matrices for block ciphers and hash functions. IACR Cryptology ePrint Archive 2015, 449 (2015)

Provable Security

How to Prove KDM Security of BHHO

Hayato Tada(✉), Akinaga Ueda, and Kaoru Kurosawa

Ibaraki University, Hitachi, Japan
{17nm710h,kaoru.kurosawa.kk}@vc.ibaraki.ac.jp

Abstract. Boneh et al. showed a KDM secure public-key encryption scheme under the DDH assumption. The KDM security means that even when any affine function of the secret keys is encrypted, it is guaranteed to be secure. In this paper, we show a more tight proof. The reduction loss to the DDH assumption is 2/3 times smaller in the KDM$^{(1)}$-security, and 5/6 times smaller in the KDM$^{(n)}$-security, where n is the number of users. Our proof is also conceptually simpler.

Keywords: Public key · Encryption scheme · KDM security
Matrix DDH

1 Introduction

1.1 Background

In the usual model of encryption schemes, it is assumed that a plaintext is independent of the secret key sk. The key dependent message (KDM) security is a more demanding notion such that the security is guaranteed even when sk is encrypted. In general, a KDM$^{(n)}$ secure encryption scheme with respect to a set of functions $\mathcal{F}$ provides security even when one encrypts $f(sk_1, \ldots, sk_n)$ under a public key pk_j for any function $f \in \mathcal{F}$, where sk_i is the secret-key of user i for $i = 1, \ldots, n$.

Boneh et al. (BHHO) [4] constructed the first KDM$^{(n)}$ secure public-key encryption scheme under the DDH assumption. It is so with respect to all affine functions. They first introduced the matrix DDH assumption as a main technical tool, and showed that it is reduced to the DDH assumption with some reduction loss. They then introduced the so called the expanded system, and proved the KDM$^{(n)}$ security.

Applebaum et al. [1] proved that a variant of Regev's scheme [7] is KDM$^{(n)}$ secure w.r.t. all affine functions under the LWE assumption. Brakerski and Goldwasser [2] showed that a variant of BHHO scheme is KDM$^{(n)}$ secure w.r.t. all affine functions under the subgroup indistinguishability assumption. They proved it by introducing a technique of the interactive vector game. Malkin et al. [6] showed a KDM$^{(n)}$ secure encryption scheme w.r.t. the set of functions which are computable by a polynomial-size modular arithmetic circuit. Wee [8] showed that a KDM$^{(1)}$-secure encryption scheme, where there is a single user, can be constructed from homomorphic smooth projective hash functions.

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 281–296, 2018.
https://doi.org/10.1007/978-3-319-97916-8_18

1.2 Our Contribution

In this paper, we show a more tight proof of the KDM security for BHHO scheme [4]. The reduction loss to the DDH assumption is 2/3 times smaller in the KDM$^{(1)}$-security, and 5/6 times smaller in the KDM$^{(n)}$-security. Our proof is also conceptually simpler.

In the model of IND-CPA security, an adversary queries to the encryption oracle once. In the model of KDM security, on the other hand, an adversary is allowed to query to the encryption oracle multiple times. To deal with such multiple queries, we introduce a multi-query DDH assumption. Let

$$[(a_1, \ldots, a_\ell)] = (g^{a_1}, \ldots, g^{a_\ell}),$$

where g is a generator of a group $\mathbb{G}$ of prime order p. Then the multi-query DDH assumption says

$$([\mathbf{R}_1], [r_2\mathbf{R}_1], \ldots, [r_Q\mathbf{R}_1]) \text{ and } ([\mathbf{R}_1], [\mathbf{R}_2], \ldots, [\mathbf{R}_Q])$$

are computationally indistinguishable for polynomially large Q, where $r_i \xleftarrow{\$} \mathbb{Z}_p$, $\mathbf{R}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$ and $\ell \geq 2$. We then show that it is tightly reduced to the matrix DDH assumption with the matrix size ℓ. In particular, the reduction loss is independent of Q.[1]

Now our proof of the KDM security consists of the following arguments.

(1) The KDM$^{(1)}$-security for a single query.
We prove this under the usual DDH assumption by using the idea of the interactive vector game [2].

(2) The KDM$^{(1)}$-security for multiple queries.
If we use a hybrid argument, then the reduction loss will be proportional to the number of queries Q. To make it independent of Q, we rely on the multi-query DDH assumption.

(3) How to simulate $n-1$ public keys from a single public key pk_1.
We can simulate pk_2 from pk_1 under the DDH assumption. If we further use a hybrid argument to simulate the rest of the public keys, then the reduction loss will be linear in n. To make it independent of n, we rely on the multi-query DDH assumption.

(4) We combine (2) and (3) in parallel to prove more tight KDM$^{(n)}$ security.

Our proof makes it clear that the matrix DDH assumption (through our multi-query DDH assumption) is useful to make the reduction loss small.

Finally we derive a more tight reduction from the matrix DDH assumption to the DDH assumption than Boneh et al. The loss factor is reduced from $O(\ell)$ to $O(\log \ell)$, where ℓ is the size of the matrix. Then both the KDM$^{(1)}$ security and the KDM$^{(n)}$ security of BHHO encryption scheme are further improved.

Remark 1. Galindo et al. [5] showed an almost tight reduction from the matrix DLIN assumption to the DLIN assumption in constructing an ID-based KDM-secure encryption scheme.

[1] It is tightly reduced to the DDH assumption if $\ell = 2$ or $Q = 2$. Otherwise, however, it is not.

2 Preliminaries

Let $\mathbb{G}$ be a group of prime order p with a generator g. Define L_{DDH} and L_R as

$$L_{DDH} = \left\{ \mathbf{D} \mid \mathbf{D} = \begin{pmatrix} 1, & t \\ \gamma, & \gamma t \end{pmatrix}, t \in \mathbb{Z}_p, \gamma \in \mathbb{Z}_p \right\}$$
$$L_R = \left\{ \mathbf{D} \mid \mathbf{D} = \begin{pmatrix} 1, & t \\ \gamma_1, & \gamma_2 \end{pmatrix}, t \in \mathbb{Z}_p, \gamma_1 \in \mathbb{Z}_p, \gamma_2 \in \mathbb{Z}_p, \gamma_2 \neq \gamma_1 t \right\}$$

For a matrix $\mathbf{A} = (a_{i,j})$ over $\mathbb{Z}_p$, let $[\mathbf{A}]$ be a matrix $(g^{a_{i,j}})$ over $\mathbb{G}$. In particular, if $\mathbf{u} = (a_1, \ldots, a_\ell)$, then

$$[\mathbf{u}] = [(a_1, \ldots, a_\ell)] = (g^{a_1}, \ldots, g^{a_\ell}).$$

When X is a set, $x \xleftarrow{\$} X$ means that x is randomly chosen from X. When $A(x; r)$ is a probabilistic algorithm, $y \leftarrow A(x)$ means that $y = A(x; r)$ is computed for a random string r. The cardinality of a set X is denoted by $|X|$. The bit length of a string Y is denoted by $|Y|$. All the algorithms in this paper run in probabilistic polynomial time, and the security parameter is denoted by k.

2.1 DDH Assumption

The DDH assumption is that L_{DDH} and L_R are indistinguishable. Formally consider the following game between a challenger and an adversary $\mathcal{A}$.

1. The challenger chooses $b \xleftarrow{\$} \{0, 1\}$ and

$$\mathbf{D} \xleftarrow{\$} \begin{cases} L_{DDH} & if\ b = 0 \\ L_R & if\ b = 1 \end{cases}$$

   He then sends $[\mathbf{D}]$ to the adversary $\mathcal{A}$.
2. The adversary $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

Define

$$\mathsf{DDHAdv}(\mathcal{A}) = |\Pr(b' = 1 | b = 0) - \Pr(b' = 1 | b = 1)|$$

and

$$\mathsf{DDHAdv} = \max_{\mathcal{A}} \mathsf{DDHAdv}(\mathcal{A}).$$

2.2 Matrix DDH Assumption

Boneh et al. [4] introduced the matrix DDH assumption. Let $Rk_i(\mathbb{Z}_p^{\ell_1 \times \ell_2})$ be the set of all $\ell_1 \times \ell_2$ matrices over $\mathbb{Z}_p$ with rank i. Then the matrix DDH assumption is that $\{[\mathbf{D}] \mid \mathbf{D} \in Rk_{r_1}(\mathbb{Z}_p^{\ell_1 \times \ell_2})\}$ and $\{[\mathbf{D}] \mid \mathbf{D} \in Rk_{r_2}(\mathbb{Z}_p^{\ell_1 \times \ell_2})\}$ are indistinguishable, where $1 \leq r_1 < r_2 \leq \min\{\ell_1, \ell_2\}$.

Formally we consider the following game.

1. The challenger chooses $b \xleftarrow{\$} \{0,1\}$ and

$$\mathbf{D} \xleftarrow{\$} \begin{cases} Rk_{r_1}(\mathbb{Z}_p^{\ell_1 \times \ell_2}) \; if \; b = 0 \\ Rk_{r_2}(\mathbb{Z}_p^{\ell_1 \times \ell_2}) \; if \; b = 1 \end{cases}$$

He then sends $[\mathbf{D}]$ to the adversary $\mathcal{A}$.
2. The adversary $\mathcal{A}$ outputs $b' \in \{0,1\}$.

Define

$$\mathsf{MatrixAdv}^{(r_1,r_2;\ell_1,\ell_2)}(\mathcal{A}) = |\Pr(b' = 1|b = 0) - \Pr(b' = 1|b = 1)|$$

Boneh et al. [4] proved that the DDH assumption implies the matrix DDH assumption.

Proposition 1. *For any $\mathcal{A}$, there exists $\mathcal{B}$, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{MatrixAdv}^{(r_1,r_2;\ell_1,\ell_2)}(\mathcal{A}) \leq (r_2 - r_1)\mathsf{DDHAdv}(\mathcal{B}).$$

3 KDM Secure Encryption Scheme

3.1 KDM Security

Let $\mathcal{E} = (K, E, D)$ be a public-key encryption scheme, where K is a key generation algorithm, E is an encryption algorithm and D is a decryption algorithm. For a security parameter k, let S_k be the space of secret keys sk and M_k be that of messages m, where $(sk, pk) \leftarrow K(1^k)$.

The KDM security of $\mathcal{E} = (K, E, D)$ is parameterized by the number of users n and a set of functions $F_k = \{f \mid f : S_k^n \rightarrow M_k\}$ in such a way that the encryption of $f(sk_1, \ldots, sk_n)$ should be indistinguishable from that of 0, where $f \in F_k$. Formally, we consider a game between a challenger and an adversary $\mathcal{A}$ as follows.

Initialize. First the challenger choose $b \xleftarrow{\$} \{0,1\}$. Next he generates $(sk_i, pk_i) \leftarrow K(1^k)$ for $i = 1, \ldots, n$, and sends n public keys $(pk_1, \ldots, pk_n)$ to the adversary $\mathcal{A}$.

Query. The adversary $\mathcal{A}$ queries $(j, f) \in \{1, \ldots, n\} \times F_k$ to the challenger. For each query, the challenger computes $x = f(sk_1, \ldots, sk_n)$ and sends the following ciphertext c to the adversary $\mathcal{A}$.

$$c = \begin{cases} E_{pk_j}(x) \; if \; b = 0 \\ E_{pk_j}(0) \; if \; b = 1 \end{cases}$$

Finish. The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.

Define

$$\begin{aligned}\mathsf{KDM}^{(n)}\mathsf{Adv}(\mathcal{A}) &= |\Pr(b'=1|b=0) - \Pr(b'=1|b=1)| \\ &= |2\Pr(b'=b) - 1| \end{aligned} \tag{1}$$

and

$$\mathsf{KDM}^{(n)}\mathsf{Adv} = \max_{\mathcal{A}} \mathsf{KDM}^{(n)}\mathsf{Adv}(\mathcal{A}).$$

We say that $\mathcal{E}$ is KDM$^{(n)}$-secure with respect to the function classes $\mathcal{F} = \{F_k\}$ if $\mathsf{KDM}^{(n)}\mathsf{Adv}$ is negligible. We also say that $\mathcal{A}$ is a $\mathcal{F}$-KDM$^{(n)}$ adversary.

3.2 BHHO Encryption Scheme

Boneh et al. [4] showed the KDM$^{(n)}$-secure encryption scheme with respect to all affine functions such as follows.

Key Generation. For $\ell = \lceil 3 \log_2 p \rceil$, choose $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^\ell$ and $\mathbf{sk} \xleftarrow{\$} \{0,1\}^\ell$. Then output a public key $pk = [\mathbf{u}, -\mathbf{u} \cdot \mathbf{sk}^T]$ and a secret key $\mathbf{sk}$.

Encryption. To encrypt $[m] \in \mathbb{G}$, choose $r \xleftarrow{\$} \mathbb{Z}_p$ and output a ciphertext such that $E_{pk}(m;r) = [r\mathbf{u}, -r\mathbf{u} \cdot \mathbf{sk}^T + m]$.

Decryption. For a ciphertext $[(\mathbf{c}, d)]$ and a secret key $\mathbf{sk}$, output $[m] = [d + \mathbf{c} \cdot \mathbf{sk}^T)$, where $\mathbf{c} \in \mathbb{Z}_p^\ell$.

Let $\mathbf{sk}_1, \ldots, \mathbf{sk}_n$ be n secret keys. For $a_0 \in \mathbb{Z}_p$ and $\mathbf{a}_1 \in \mathbb{Z}_p^\ell, \ldots, \mathbf{a}_n \in \mathbb{Z}_p^\ell$, define

$$f_{a_0,\mathbf{a}_1,\ldots,\mathbf{a}_n}(\mathbf{sk}_1, \ldots, \mathbf{sk}_n) = [a_0 + \sum_i \mathbf{a}_i \cdot \mathbf{sk}_i^T].$$

Let $\mathcal{F}_{\text{affine}}$ be the set of functions $f_{a_0,\mathbf{a}_1,\ldots,\mathbf{a}_n}$. Then Boneh et al. [4] proved the following proposition.

Proposition 1. *For $\mathcal{F}_{\text{affine}}$-KDM$^{(n)}$ adversary $\mathcal{A}$, there exist $\mathcal{B}_1$ and $\mathcal{B}_2$, both running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{KDM}^{(n)}\mathsf{Adv} \le 4 \cdot \mathsf{MatrixAdv}^{(1,\ell;\ell+1,\ell)}(\mathcal{B}_1) + 2 \cdot \mathsf{MatrixAdv}^{(1,\ell+1;\ell+1,\ell+1)}(\mathcal{B}_2) + 2/p.$$

From Proposition 1, we obtain the following corollary.

Corollary 1. *For $\mathcal{F}_{\text{affine}}$-KDM$^{(n)}$ adversaries, $\mathcal{A}$, there exists $\mathcal{B}$ running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{KDM}^{(n)}\mathsf{Adv}(\mathcal{A}) \le (6\ell - 4)\mathsf{DDHAdv}(\mathcal{B}) + 2/p.$$

Remark 2. Boneh et al. [4] defined $\mathsf{KDM}^{(n)}\mathsf{Adv}(\mathcal{A})$ as $|\Pr(b'=b) - 1/2|$ instead of $|2\Pr(b'=b) - 1|$. Hence it was written as

$$\mathsf{KDM}^{(n)}\mathsf{Adv}(\mathcal{A}) \le (3\ell - 2)\mathsf{DDHAdv}(\mathcal{B}) + 1/p$$

in [4, Theorem 1].

4 Outline of Our Proof

Recall that an adversary is allowed to query to the encryption oracle multiple times in the model of KDM security. We write $U \approx V$ to denote that two random variables U and V are computationally indistinguishable.

4.1 KDM$^{(1)}$ Security for a Single Query

In BHHO encryption scheme, a secret key is $\mathbf{sk} \xleftarrow{\$} \{0,1\}^\ell$ and the public key is given by $pk = [\mathbf{u}, -\mathbf{u}\cdot\mathbf{sk}^T]$, where $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^\ell$. It is well known that $(\mathbf{u}, r\mathbf{u}) \approx (\mathbf{u}, \mathbf{R})$ under the DDH assumption, where $r \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{R} \xleftarrow{\$} \mathbb{Z}_p^\ell$, and the reduction is tight. Then we can prove the KDM$^{(1)}$ security for a single query as follows. (We use the idea of the interactive vector game [2] here.)

$$
\begin{aligned}
E_{pk}([a_0 + \mathbf{a}\cdot\mathbf{sk}^T]) &= [r(\mathbf{u}, -\mathbf{u}\cdot\mathbf{sk}^T) + (0^\ell, a_0 + \mathbf{a}\cdot\mathbf{sk}^T)], \text{ where } r \xleftarrow{\$} \mathbb{Z}_p. \\
&\approx [(\mathbf{R}, -\mathbf{R}\cdot\mathbf{sk}^T) + (0^\ell, a_0 + \mathbf{a}\cdot\mathbf{sk}^T)] \qquad : \mathbf{R} \leftarrow r\mathbf{u} \text{ from DDH} \\
&= [\mathbf{R}, a_0 + (\mathbf{a} - \mathbf{R})\cdot\mathbf{sk}^T] \\
&\equiv [\mathbf{R} + \mathbf{a}, a_0 - \mathbf{R}\cdot\mathbf{sk}^T] \qquad : \mathbf{R} \leftarrow (\mathbf{R} - \mathbf{a}) \text{ since } \mathbf{R} \text{ is random} \\
&\approx [r\mathbf{u} + \mathbf{a}, a_0 - r\mathbf{u}\cdot\mathbf{sk}^T] \qquad : r\mathbf{u} \leftarrow \mathbf{R} \text{ from DDH} \\
&\approx [r\mathbf{u} + \mathbf{a}, a_0 - rs_0] \qquad : s_0 \xleftarrow{\$} \mathbb{Z}_p \text{ from the leftover hash lemma} \\
&\qquad\qquad\qquad\qquad\qquad\quad pk = [\mathbf{u}, -s_0] \leftarrow pk = [\mathbf{u}, -\mathbf{u}\cdot\mathbf{sk}^T] \\
&\approx [\mathbf{R} + \mathbf{a}, a_0 - r] \qquad : (\mathbf{R}, -r) \leftarrow r(\mathbf{u}, -s_0) \text{ from DDH} \\
&\equiv [\mathbf{R}, -r] \qquad : \text{since } (\mathbf{R}, -r) \text{ is random} \\
&\approx [r(\mathbf{u}, -s_0)] \qquad : \text{from DDH} \\
&\approx [r(\mathbf{u}, -\mathbf{u}\cdot\mathbf{sk}^T)] \qquad : \text{from the leftover hash lemma} \\
&\qquad\qquad\qquad\qquad\qquad\quad pk = [\mathbf{u}, -\mathbf{u}\cdot\mathbf{sk}^T] \leftarrow pk = [\mathbf{u}, -s_0] \\
&= E_{pk}([0]) \qquad (2)
\end{aligned}
$$

For the leftover hash lemma, see [4, Lemma 2, Corollary 1].

4.2 KDM$^{(1)}$ Security for Multiple Queries

If we use a hybrid argument to extend the above result to multiple queries, however, the reduction loss will be proportional to the number of queries Q. To make it independent of Q, we introduce a multi-query DDH assumption. It says

$$[\mathbf{u}, r_1\mathbf{u}, \ldots, r_Q\mathbf{u}] \approx [\mathbf{u}, \mathbf{R}_1, \ldots, \mathbf{R}_Q] \qquad (3)$$

for polynomially large Q, where $r_i \xleftarrow{\$} \mathbb{Z}_p$ and $\mathbf{R}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$. We then prove that it is tightly reduced to the matrix DDH assumption, in fact, let $\mathbf{D}$ be an $\ell \times \ell$ matrix and $\mathbf{u}$ be the first row. Then a random linear combination of the rows of $\mathbf{D}$ yields $r_i \cdot \mathbf{u}$ if $rank(\mathbf{D}) = 1$, and $\mathbf{R}_i$ if $rank(\mathbf{D}) = \ell$. Hence the indistinguishability is tightly reduced to the matrix DDH assumption and independent of Q.

Now in Eq. (2), replace the DDH assumption with the multi-query DDH assumption.

4.3 KDM$^{(n)}$ Security

For $i = 1, \ldots, n$, let $pk_i = [\mathbf{u}_i, -\mathbf{u}_i \cdot \mathbf{sk}_i^T]$ be the public key of user i, where $\mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$ and $\mathbf{sk}_i \xleftarrow{\$} \{0,1\}^\ell$ is the secret key. For $i = 2, \ldots, n$, we can write $\mathbf{sk}_i$ as

$$\mathbf{sk}_i = \mathbf{sk}_1 \oplus \mathbf{e}_i = \mathbf{sk}_1 \cdot \mathbf{T}(\mathbf{e}_i) + \mathbf{e}_i, \tag{4}$$

where $\mathbf{e}_i \xleftarrow{\$} \{0,1\}^\ell$ and $\mathbf{T}(\mathbf{e}) = diag((-1)^{e_i})$. Then we simulate pk_i by using pk_1 under the multi-query DDH assumption as follows.

$$\begin{aligned}
pk_i &= [\mathbf{u}_i, -\mathbf{u}_i \cdot \mathbf{sk}_i^T] \\
&\equiv [\mathbf{u}_i \cdot \mathbf{T}(\mathbf{e}_i)^{-1}, -\mathbf{u}_i \cdot \mathbf{T}(\mathbf{e}_i)^{-1} \cdot \mathbf{sk}_i^T] && : \text{since } \mathbf{u}_i \text{ is random} \\
&\approx [\gamma_i \mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_i)^{-1}, -\gamma_i \mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_i)^{-1} \cdot \mathbf{sk}_i^T] && : \text{where } \gamma_i \xleftarrow{\$} \mathbb{Z}_p, \\
& && \quad \text{from multi-query DDH} \\
&= [\gamma_i \cdot \mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_i)^{-1}, -(\gamma_i \cdot \mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_i)^{-1}) \cdot (\mathbf{sk}_1 \cdot \mathbf{T}(\mathbf{e}_i) + \mathbf{e}_i)^T] && : \text{from Eq. (4)} \\
&= [\gamma_i \cdot \mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_i)^{-1}, -\gamma_i (\mathbf{u}_1 \cdot \mathbf{sk}_1^T) - \gamma_i (\mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_i)^{-1} \cdot \mathbf{e}_i^T)]
\end{aligned}$$

Thus we can simulate pk_i from $pk_1 = [\mathbf{u}_1, -\mathbf{u}_1 \cdot \mathbf{sk}_1^T]$ and $\mathbf{e}_i$. Further $a_0 + \sum_{i=1}^n \mathbf{a}_i \cdot \mathbf{s}_\mathbf{i}^\intercal$ is expressed as an affine function of $\mathbf{sk}_1$ from Eq. (4).

We combine the above argument with that of Sect. 4.2 in parallel to prove more tight KDM$^{(n)}$ security.

4.4 Final Result

Our proofs show that

$$\mathsf{KDM}^{(1)}\mathsf{Adv} \leq 4\ell \cdot \mathsf{DDHAdv} + 2/p \tag{5}$$

$$\mathsf{KDM}^{(n)}\mathsf{Adv} \leq 5\ell \cdot \mathsf{DDHAdv} + 2/p. \tag{6}$$

while Boneh et al. showed only

$$\mathsf{KDM}^{(n)}\mathsf{Adv} \leq 6\ell \cdot \mathsf{DDHAdv} + 2/p$$

as shown in Corollary 1 (see Remark 2 also.) Finally we improve our results to

$$\mathsf{KDM}^{(1)}\mathsf{Adv} \leq 4\log(\ell) \cdot \mathsf{DDHAdv} + 2/p$$

$$\mathsf{KDM}^{(n)}\mathsf{Adv} \leq 5\log(\ell) \cdot \mathsf{DDHAdv} + 2/p$$

by deriving a more tight reduction from the matrix DDH assumption to the DDH assumption than Boneh et al. [4].

5 Multi-query DDH Assumption

In this section, we introduce a multi-query DDH assumption, and prove that it is tightly reduced to the matrix DDH assumption.

5.1 Basic Form

Suppose that an adversary $\mathcal{A}$ is given $[\mathbf{u}]$ such that $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^\ell$. Let $\mathcal{O}_{\mathbf{u}}$ be the oracle which outputs $[r\mathbf{u}]$ such that $r \xleftarrow{\$} \mathbb{Z}_p$. Let $\mathcal{O}_R$ be the oracle which outputs $[\mathbf{R}]$ such that $\mathbf{R} \xleftarrow{\$} \mathbb{Z}_p^\ell$.

Define

$$\mathsf{MultiDDH}^{(\ell)}\mathsf{Adv}(\mathcal{A}) = |\Pr(\mathcal{A}^{\mathcal{O}_{\mathbf{u}}}([\mathbf{u}]) = 1 - \Pr(\mathcal{A}^{\mathcal{O}_R}([\mathbf{u}]) = 1|.$$

The multi-query DDH assumption is that $\mathsf{MultiDDH}^{(\ell)}\mathsf{Adv}_{\mathcal{A}}$ is negligible for any PPT $\mathcal{A}$.

Theorem 1. *For any adversary $\mathcal{A}$ of above, there exists a matrix DDH adversary $\mathcal{B}$, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{MultiDDH}^{(\ell)}\mathsf{Adv}(\mathcal{A}) \leq \mathsf{MatrixAdv}^{(1,\ell;\ell,\ell)}(\mathcal{B}).$$

Proof. We construct a matrix DDH adversary $\mathcal{B}$ as follows.

1. $\mathcal{B}$ receives $[\mathbf{D}]$ from the challenger, where $\mathbf{D} \xleftarrow{\$} Rk_1(\mathbb{Z}_p^{\ell\times\ell})$ or $\mathbf{D} \xleftarrow{\$} Rk_\ell(\mathbb{Z}_p^{\ell\times\ell})$.
2. Let $[\mathbf{u}]$ be the first row of $[\mathbf{D}]$. $\mathcal{B}$ gives $[\mathbf{u}]$ to $\mathcal{A}$.
3. If $\mathcal{A}$ queries to the oracle, then $\mathcal{B}$ chooses $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_p^\ell$, and returns $[\mathbf{x}\cdot\mathbf{D}]$ to $\mathcal{A}$.
4. Finally $\mathcal{B}$ receives b' from $\mathcal{A}$, and sends it to the challenger.

If $\mathbf{D} \xleftarrow{\$} Rk_1(\mathbb{Z}_p^{\ell\times\ell})$, then it is easy to see that $\mathbf{x}\cdot\mathbf{D} = r\mathbf{u}$, where r is uniformly distributed over $\mathbb{Z}_p$. If $\mathbf{D} \xleftarrow{\$} Rk_\ell(\mathbb{Z}_p^{\ell\times\ell})$, then it is also easy to see that $\mathbf{x}\cdot\mathbf{D}$ is uniformly distributed over $\mathbb{Z}_p^\ell$. Therefore the theorem holds. □

5.2 Generalized Form

Suppose that an adversary $\mathcal{A}$ is given $[\mathbf{u}_1, \ldots, \mathbf{u}_n]$ such that $\mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$ for $i = 1, \ldots, n$. Let $\mathcal{O}_{\mathbf{u}_1,\ldots,\mathbf{u}_n}$ be the oracle which outputs $[r_1\mathbf{u}_1, \ldots, r_n\mathbf{u}_n]$ such that $r_i \xleftarrow{\$} \mathbb{Z}_p$ for $i = 1, \ldots, n$. Let $\mathcal{O}_{R^n}$ be the oracle which outputs $[\mathbf{R}_1, \ldots, \mathbf{R}_n]$ such that $\mathbf{R}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$ for $i = 1, \ldots, n$.

Define

$$\begin{aligned}\mathsf{MultiDDH}^{(n,\ell)}\mathsf{Adv}(\mathcal{A}) = |\Pr(\mathcal{A}^{\mathcal{O}_{\mathbf{u}_1,\ldots,\mathbf{u}_n}}([\mathbf{u}_1, \ldots, \mathbf{u}_n]) = 1 \\ - \Pr(\mathcal{A}^{\mathcal{O}_R^n}([\mathbf{u}_1, \ldots, \mathbf{u}_n]) = 1|.\end{aligned}$$

The generalized multi-query DDH assumption is that $\mathsf{MultiDDH}^{(n,\ell)}\mathsf{Adv}_{\mathcal{A}}$ is negligible for any PPT $\mathcal{A}$.

Theorem 2. *For any adversary $\mathcal{A}$ of above, there exists a matrix DDH adversary $\mathcal{B}$, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{MultiDDH}^{(n,\ell)}\mathsf{Adv}(\mathcal{A}) \leq \mathsf{MatrixAdv}^{(1,\ell;\ell,\ell)}(\mathcal{B}).$$

Proof. We construct a matrix DDH adversary $\mathcal{B}$ as follows.

1. $\mathcal{B}$ receives $[\mathbf{D}]$ from the challenger, where $\mathbf{D} \xleftarrow{\$} Rk_1(\mathbb{Z}_p^{\ell\times\ell})$ or $\mathbf{D} \xleftarrow{\$} Rk_\ell(\mathbb{Z}_p^{\ell\times\ell})$.
2. For $i = 1, \ldots, n$, $\mathcal{B}$ chooses $L_i \xleftarrow{\$} Rk_\ell(\mathbb{Z}_p^{\ell\times\ell})$, $R_i \xleftarrow{\$} Rk_\ell(\mathbb{Z}_p^{\ell\times\ell})$ and computes
$$[\mathbf{D}_i] = [L_i \cdot \mathbf{D} \cdot R_i].$$
Let $[\mathbf{u}_i]$ be the first row of $[\mathbf{D}_i]$. $\mathcal{B}$ gives $[\mathbf{u}_1, \ldots, \mathbf{u}_n]$ to $\mathcal{A}$.
3. If $\mathcal{A}$ queries to the oracle, then $\mathcal{B}$ chooses $\mathbf{x}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$, and returns $[\mathbf{x}_i \cdot \mathbf{D}_i]$ to $\mathcal{A}$ for $i = 1, \ldots, n$.
4. Finally $\mathcal{B}$ receives b' from $\mathcal{A}$, and sends it to the challenger.

The rest of the proof is the same as that of Theorem 1. □

6 More Tight KDM Security of BHHO

In this section, we show that BHHO encryption scheme has more tight KDM security. Let
$$\Delta(A, B) = \frac{1}{2} \sum_{x\in S} |\Pr(A = x) - \Pr(B = x)|$$
denote the statistical distance between two random variables A and B. Then the following proposition holds from the leftover hash lemma [4, Lemma 2, Corollary 1].

Proposition 2. *Le* $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^\ell, \mathbf{sk} \xleftarrow{\$} \{0,1\}^\ell$ *and* $s_0 \xleftarrow{\$} \mathbb{Z}_p$. *Then*
$$\Delta\Big((\mathbf{u}, \mathbf{u} \cdot \mathbf{sk}^T); (\mathbf{u}, s_0)\Big) \leq 1/p,$$
where $\ell = \lceil 3 \log_2 p \rceil$.

6.1 How to Prove KDM$^{(1)}$ Security

Theorem 3. *In BHHO encryption scheme, for any* $\mathcal{F}_{\text{affine}}$*-KDM*$^{(1)}$ *adversary* $\mathcal{A}$*, there exist multi-query DDH adversaries* $\mathcal{B}_1$ *and* $\mathcal{B}_2$*, both running in about the same time as* $\mathcal{A}$*, such that*
$$\mathsf{KDM}^{(1)}\mathsf{Adv}(\mathcal{A}) \leq 2 \cdot \mathsf{MultiDDH}^{(\ell)}(\mathcal{B}_1) + 2 \cdot \mathsf{MultiDDH}^{(\ell+1)}(\mathcal{B}_2) + 2/p,$$
where $\ell = \lceil 3 \log_2 p \rceil$.

Proof. Let $pk = [\mathbf{u}, \mathbf{u} \cdot \mathbf{sk}^T]$ be a public key of BHHO encryption scheme, where $\mathbf{sk}$ is the secret key. In Eq. (2), we replace the DDH assumption with the multi-query DDH assumption as follows.

$$
\begin{aligned}
E_{pk}([a_0 + \mathbf{a} \cdot \mathbf{sk}^T]) &= [r(\mathbf{u}, -\mathbf{u} \cdot \mathbf{sk}^T) + (0^\ell, a_0 + \mathbf{a} \cdot \mathbf{sk}^T)], \text{ where } r \xleftarrow{\$} \mathbb{Z}_p. \\
&\approx [(\mathbf{R}, -\mathbf{R} \cdot \mathbf{sk}^T) + (0^\ell, a_0 + \mathbf{a} \cdot \mathbf{sk}^T)] \quad : \mathbf{R} \leftarrow r\mathbf{u} \\
& \qquad\qquad\qquad\qquad\qquad\qquad \text{from multi-query DDH of length } \ell \\
&= [\mathbf{R}, a_0 + (\mathbf{a} - \mathbf{R}) \cdot \mathbf{sk}^T] \\
&\equiv [\mathbf{R} + \mathbf{a}, a_0 - \mathbf{R} \cdot \mathbf{sk}^T] \qquad : \mathbf{R} \leftarrow (\mathbf{R} - \mathbf{a}) \text{ since } \mathbf{R} \text{ is random} \\
&\approx [r\mathbf{u} + \mathbf{a}, a_0 - r\mathbf{u} \cdot \mathbf{sk}^T] \qquad : r\mathbf{u} \leftarrow \mathbf{R} \\
& \qquad\qquad\qquad\qquad\qquad\qquad \text{from multi-query DDH of length } \ell \\
&\approx [r\mathbf{u} + \mathbf{a}, a_0 - rs_0] \qquad : s_0 \xleftarrow{\$} \mathbb{Z}_p \text{ from Proposition 2} \\
& \qquad\qquad\qquad\qquad\qquad\qquad pk = [\mathbf{u}, -s_0] \leftarrow pk = [\mathbf{u}, -\mathbf{u} \cdot \mathbf{sk}^T] \\
&\approx [\mathbf{R} + \mathbf{a}, a_0 - r] \qquad : (\mathbf{R}, -r) \leftarrow r(\mathbf{u}, -s_0) \\
& \qquad\qquad\qquad\qquad\qquad\qquad \text{from multi-query DDH of length } \ell + 1 \\
&\equiv [\mathbf{R}, -r] \qquad : \text{since } (\mathbf{R}, -r) \text{ is random} \\
&\approx [r(\mathbf{u}, -s_0)] \qquad : \text{from multi-query DDH of length } \ell + 1 \\
&\approx [r(\mathbf{u}, -\mathbf{u} \cdot \mathbf{sk}^T)] \qquad : \text{from Proposition 2} \\
& \qquad\qquad\qquad\qquad\qquad\qquad pk = [\mathbf{u}, -\mathbf{u} \cdot \mathbf{sk}^T] \leftarrow pk = [\mathbf{u}, -s_0] \\
&= E_{pk}([0]) \qquad (7)
\end{aligned}
$$

Therefore the theorem holds.

□

Corollary 2. *In BHHO encryption scheme, for any $\mathcal{F}_{\text{affine}}$-KDM$^{(1)}$ adversary $\mathcal{A}$, there exist multi-query DDH adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$, both running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{KDM}^{(1)}\mathsf{Adv}(\mathcal{A}) \leq 2 \cdot \mathsf{MatrixAdv}^{(1,\ell;\ell,\ell)}(\mathcal{B}_1) + 2 \cdot \mathsf{MatrixAdv}^{(1,\ell+1;\ell+1,\ell+1)}(\mathcal{B}_2) + 2/p,$$

where $\ell = \lceil 3 \log_2 p \rceil$.

Proof. From Theorem 1. □

If we use Corollary 1, we obtain the following corollary.

Corollary 3. *In BHHO encryption scheme, for any $\mathcal{F}_{\text{affine}}$-KDM$^{(1)}$ adversary $\mathcal{A}$, there exists a DDH adversary $\mathcal{B}$, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{KDM}^{(1)}\mathsf{Adv}(\mathcal{A}) \leq (4\ell - 2)\mathsf{DDHAdv}(\mathcal{B}) + 2/p.$$

6.2 How to Prove KDM$^{(n)}$ Security

Lemma 1. *For $\mathbf{e} = (e_1, \ldots, e_\ell) \in \{0,1\}^\ell$, define an $\ell \times \ell$ matrix $\mathbf{T}(\mathbf{e})$ as $\mathbf{T}(\mathbf{e}) = diag((-1)^{e_i})$ Then for $\mathbf{s} = (s_1, \ldots, s_\ell) \in \{0,1\}^\ell$, it holds that*

$$\mathbf{s} \oplus \mathbf{e} = \mathbf{s} \cdot \mathbf{T}(\mathbf{e}) + \mathbf{e}$$

Proof. We can see that

$$s_i \oplus e_i = (-1)^{e_i} s_i + e_i.$$

Hence the result follows. □

Theorem 4. *In BHHO encryption scheme, for any $\mathcal{F}_{\text{affine}}$-KDM$^{(n)}$ adversary $\mathcal{A}$, there exist multi-query DDH adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$, each running in about the same time as $\mathcal{A}$, such that*

$$\begin{aligned}\mathsf{KDM}^{(n)}\mathsf{Adv}(\mathcal{A}) \leq\ & \mathsf{MultiDDH}^{(n,\ell)}\mathsf{Adv}(\mathcal{B}_1) \\ & +2 \cdot \mathsf{MultiDDH}^{(\ell)}\mathsf{Adv}(\mathcal{B}_2) + 2 \cdot \mathsf{MultiDDH}^{(\ell+1)}\mathsf{Adv}(\mathcal{B}_3),\end{aligned}$$

where $\ell = \lceil 3 \log_2 p \rceil$.

Proof. We consider a series of games, **Game** 0, …, **Game** 13 as shown below. For each game, define

$$p_i = \Pr(\mathcal{A} \text{ outputs } b' = 1 \text{ in } \textbf{Game } i).$$

Game 0. This is the same as the KDM$^{(n)}$ security game with $b = 0$. The challenger chooses $\mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_p^\ell$ and $\mathbf{sk}_i \xleftarrow{\$} \{0,1\}^\ell$ for $i = 1, \ldots, n$. He then sends n public keys $pk_1, \ldots, pk_n$ to the adversary $\mathcal{A}$ such that $pk_i = [\mathbf{u}_i, y_i]$, where $y_i = -\mathbf{u}_i \cdot \mathbf{sk}_i^T$ and $\mathbf{sk}_i$ is the secret key.
If the adversary $\mathcal{A}$ queries $(j, f_{a_0,\mathbf{a}_1,\ldots,\mathbf{a}_n})$ to the challenger, the challenger returns

$$C = E_{pk_j}([a_0 + \sum_i \mathbf{a}_i \cdot \mathbf{sk}_i^T]).$$

Game 1. This is the same as **Game** 0 except that for $i = 2, \ldots, n$, the challenger chooses $\mathbf{e}_i \xleftarrow{\$} \{0,1\}^\ell$ and sets $\mathbf{sk}_i = \mathbf{sk}_1 \oplus \mathbf{e}_i$. It is clear that $p_1 = p_0$.
From Lemma 1, it holds that

$$\mathbf{sk}_i = \mathbf{sk}_1 \oplus \mathbf{e}_i = \mathbf{sk}_1 \cdot \mathbf{T}(\mathbf{e}_i) + \mathbf{e}_i \tag{8}$$

Hence we have

$$pk_i = [\mathbf{u}_i, -\mathbf{u}_i \cdot (\mathbf{sk}_1 \cdot \mathbf{T}(\mathbf{e}_i) + \mathbf{e}_i)^T].$$

Also $a_0 + \sum_i \mathbf{a}_i \cdot \mathbf{sk}_i^T$ is written as an affine function on $\mathbf{sk}_1$ such that

$$a_0 + \sum_i \mathbf{a}_i \cdot \mathbf{sk}_i^T = b_0 + \mathbf{b} \cdot \mathbf{sk}_1$$

for some $b_0 \in \mathbb{Z}_p$ and $\mathbf{b} \in \mathbb{Z}_p^\ell$. Therefore for a query $(j, f_{a_0,\mathbf{a}_1,\ldots,\mathbf{a}_n})$, the challenger returns

$$\begin{aligned} C &= E_{pk_j}([b_0 + \mathbf{b} \cdot \mathbf{sk}_1^T]) \\ &= ([r\mathbf{u}_j, ry_j + b_0 + \mathbf{b} \cdot \mathbf{sk}_1^T]) \\ &= ([r\mathbf{u}_j, -r\mathbf{u}_j \cdot (\mathbf{T}(\mathbf{e}_j) \cdot \mathbf{sk}_1^T + \mathbf{e}_j^T) + b_0 + \mathbf{b} \cdot \mathbf{sk}_1^T]) \end{aligned}$$

Game 2. This is the same as **Game** 1 except that the challenger replaces $\mathbf{u}_j$ with $\mathbf{u}_j \cdot \mathbf{T}(\mathbf{e}_j)^{-1}$ for $j = 1, \ldots, n$. Then $p_2 = p_1$ because $\mathbf{u}_j$ is random. Since $\mathbf{T}(\mathbf{e}_j)^T = \mathbf{T}(\mathbf{e}_j)$, we have

$$\begin{aligned} pk_j &= [\mathbf{u}_j \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -\mathbf{u}_j \cdot (\mathbf{sk}_1^T + \cdot\mathbf{T}(\mathbf{e}_j)^{-1}\mathbf{e}_j^T)] \\ C &= [r\mathbf{u}_j \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -r\mathbf{u}_j \cdot (\mathbf{sk}_1^T + \cdot\mathbf{T}(\mathbf{e}_j)^{-1}\mathbf{e}_j^T) + b_0 + \mathbf{b} \cdot \mathbf{sk}_1^T] \end{aligned}$$

Game 3. This is the same as **Game** 1 except that the challenger replaces $r\mathbf{u}_j$ with $\mathbf{R} \xleftarrow{\$} \mathbb{Z}_p^\ell$ for $j = 1, \ldots, n$. Then we can construct a general multi-query DDH adversary $\mathcal{B}_1$ such that

$$|p_3 - p_2| \leq \mathsf{MultiDDH}^{(n,\ell)}\mathsf{Adv}(\mathcal{B}_1).$$

In this game, we have

$$\begin{aligned} C &= [\mathbf{R} \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -\mathbf{R} \cdot \mathbf{sk}_1^T - \mathbf{R} \cdot \mathbf{T}(\mathbf{e}_j)^{-1}\mathbf{e}_j^T + b_0 + \mathbf{b} \cdot \mathbf{sk}_1^T] \\ &= [\mathbf{R} \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, b_0 - (\mathbf{b} - \mathbf{R}) \cdot \mathbf{sk}_1^T - \mathbf{R} \cdot \mathbf{T}(\mathbf{e}_j)^{-1}\mathbf{e}_j^T] \end{aligned}$$

Game 4. This is the same as **Game** 3 except that we replace $\mathbf{R} - \mathbf{b}$ with $\mathbf{R}' \xleftarrow{\$} \mathbb{Z}_p^\ell$. Then $p_4 = p_3$ because $\mathbf{R}$ is random. In this game, we have

$$C = [(\mathbf{R}' + \mathbf{b}) \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, b_0 - \mathbf{R}' \cdot \mathbf{sk}_1^T - (\mathbf{R}' + \mathbf{b}) \cdot \mathbf{T}(\mathbf{e}_j)^{-1} \cdot \mathbf{e}_j^T]$$

Game 5. This is the same as **Game** 4 except that the challenger replaces $\mathbf{R}'$ with $r\mathbf{u}_1$, where $r \xleftarrow{\$} \mathbb{Z}_p$. At the same time, the challenger replaces each $\mathbf{u}_j$ with $\gamma_j\mathbf{u}_1$, where $\gamma_j \xleftarrow{\$} \mathbb{Z}_p$. Then we can construct a multi-query DDH adversary $\mathcal{B}_2$ such that

$$|p_5 - p_4| \leq \mathsf{MultiDDH}^{(\ell)}\mathsf{Adv}(\mathcal{B}_2).$$

In this game, we have

$$\begin{aligned} pk_j &= [\gamma_j\mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -\gamma_j\mathbf{u}_1 \cdot (\mathbf{sk}_1 + \mathbf{T}(\mathbf{e}_j)^{-1}\mathbf{e}_j)^T] \\ C &= [(r\mathbf{u}_1 + \mathbf{b}) \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, b_0 - r\mathbf{u}_1 \cdot \mathbf{sk}_1^T - (r\mathbf{u}_1 + \mathbf{b}) \cdot \mathbf{T}(\mathbf{e}_j)^{-1} \cdot \mathbf{e}_j^T] \end{aligned}$$

Game 6. This is the same as **Game** 5 except that the challenger replaces $\mathbf{u}_1 \cdot \mathbf{sk}_1$ with $s_0 \xleftarrow{\$} \mathbb{Z}_p$. Then we have $|p_6 - p_5| \leq 1/p$ from Proposition 2. In this game, we have

$$\begin{aligned} pk_j &= [\gamma_j\mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -\gamma_j(s_0 + \mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1}\mathbf{e}_j^T)] \\ C &= [(r\mathbf{u}_1 + \mathbf{b}) \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, b_0 - rs_0 - (r\mathbf{u}_1 + \mathbf{b}) \cdot \mathbf{T}(\mathbf{e}_j)^{-1} \cdot \mathbf{e}_j^T] \end{aligned}$$

Game 7. This is the same as **Game** 6 except that the challenger replaces $r(\mathbf{u}_1, s_0)$ with $(\mathbf{R}, r') \xleftarrow{\$} \mathbb{Z}_p^{\ell+1}$. Then we can construct a multi-query DDH adversary $\mathcal{B}_3$ such that

$$|p_7 - p_6| \leq \mathsf{MultiDDH}^{(\ell+1)}\mathsf{Adv}(\mathcal{B}_3).$$

In this game, we have

$$C = [(\mathbf{R} + \mathbf{b}) \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, b_0 - r' - (\mathbf{R} + \mathbf{b}) \cdot \mathbf{T}(\mathbf{e}_j)^{-1} \cdot \mathbf{e}_j^T]$$

Game 8. This is the same as **Game** 7 except that the challenger replaces $(\mathbf{R}+\mathbf{b}, r'-b_0)$ with $(\mathbf{R}', r) \xleftarrow{\$} \mathbb{Z}_p^{\ell+1}$. Then $p_8 = p_7$ because $\mathbf{R}$ and r' are random. In this game, we have

$$C = [\mathbf{R}' \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -r - \mathbf{R}' \cdot \mathbf{T}(\mathbf{e}_j)^{-1} \cdot \mathbf{e}_j^T]$$

Game 9. This is the same as **Game** 8 except that the challenger replaces $(\mathbf{R}', r)$ with $r(\mathbf{u}_1, s_0)$, where $r \xleftarrow{\$} \mathbb{Z}_p$. Then we can construct a multi-query DDH adversary $\mathcal{B}_4$ such that

$$|p_9 - p_8| \leq \mathsf{MultiDDH}^{(\ell+1)}\mathsf{Adv}(\mathcal{B}_4).$$

In this game, we have

$$C = [r\mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -rs_0 - r\mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1} \cdot \mathbf{e}_j^T]$$

Game 10. This is the same as **Game** 9 except that the challenger replaces s_0 with $\mathbf{u}_1 \cdot \mathbf{sk}_1^T$. Then we have $|p_{10} - p_9| \leq 1/p$ from Proposition 2. In this game, we have

$$\begin{aligned} pk_j &= [\gamma_j \mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -\gamma_j(\mathbf{u}_1 \cdot \mathbf{sk}_1^T + \mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1}\mathbf{e}_j^T)] \\ &= [\gamma_j \mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -\gamma_j \mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1}(\mathbf{sk}_1\mathbf{T}(\mathbf{e}_j) + \mathbf{e}_j)^T] \\ C &= [r\mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -r\mathbf{u}_1 \cdot \mathbf{sk}_1^T - r\mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1} \cdot \mathbf{e}_j^T] \\ &= [r\mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -r\mathbf{u}_1 \cdot \mathbf{T}(\mathbf{e}_j)^{-1}(\mathbf{sk}_1\mathbf{T}(\mathbf{e}_j) + \mathbf{e}_j)^T] \end{aligned}$$

Game 11. This is the same as **Game** 10 except that the challenger replaces $\gamma_j \mathbf{u}_1$ with $\mathbf{u}_j \xleftarrow{\$} \mathbb{Z}_p^{\ell}$ for $j = 1, \ldots, n$. Then we can construct a multi-query DDH adversary $\mathcal{B}_5$ such that

$$|p_{11} - p_{10}| \leq \mathsf{MultiDDH}^{(\ell)}\mathsf{Adv}(\mathcal{B}_5).$$

In this game, we have

$$\begin{aligned} pk_j &= [\mathbf{u}_j \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -\mathbf{u}_j \cdot \mathbf{T}(\mathbf{e}_j)^{-1}(\mathbf{sk}_1\mathbf{T}(\mathbf{e}_j) + \mathbf{e}_j)^T] \\ C &= [(r/\gamma_j)\mathbf{u}_j \cdot \mathbf{T}(\mathbf{e}_j)^{-1}, -(r/\gamma_j)\mathbf{u}_j \cdot \mathbf{T}(\mathbf{e}_j)^{-1}(\mathbf{sk}_1\mathbf{T}(\mathbf{e}_j) + \mathbf{e}_j)^T] \end{aligned}$$

Game 12. This is the same as **Game** 11 except that the challenger replaces $\mathbf{u}_j \cdot \mathbf{T}(\mathbf{e}_j)^{-1}$ with $\mathbf{u}_j$. Then $p_{12} = p_{11}$ because $\mathbf{u}_j$ is random. In this game, we have

$$\begin{aligned} pk_j &= [\mathbf{u}_j, -\mathbf{u}_j \cdot (\mathbf{sk}_1\mathbf{T}(\mathbf{e}_j) + \mathbf{e}_j)^T] \\ C &= [(r/\gamma_j)\mathbf{u}_j, -(r/\gamma_j)\mathbf{u}_j \cdot (\mathbf{sk}_1\mathbf{T}(\mathbf{e}_j) + \mathbf{e}_j)^T] \end{aligned}$$

Game 13. This is the same as **Game** 12 except that the challenger replaces $\mathbf{sk}_1\mathbf{T}(\mathbf{e}_j) + \mathbf{e}_j$ with $\mathbf{sk}_j$. Then $p_{13} = p_{12}$ from Lemma 1. In this game, we have

$$\begin{aligned} pk_j &= [\mathbf{u}_j, -\mathbf{u}_j \cdot \mathbf{sk}_j^T] \\ C &= [(r/\gamma_j)\mathbf{u}_j, -(r/\gamma_j)\mathbf{u}_j \cdot \mathbf{sk}_j^T] = E_{pk_j}([0]) \end{aligned}$$

Finally **Game** 13 is the same as the $\mathrm{KDM}^{(n)}$ security game with $b = 1$. Therefore by summing up each $|p_{i+1} - p_i|$, we obtain this theorem.

□

Corollary 4. *In BHHO encryption scheme, for any $\mathcal{F}_{\mathrm{affine}}$-$KDM^{(n)}$ adversary $\mathcal{A}$, there exist $\mathcal{B}_1$ and $\mathcal{B}_2$, both running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{KDM}^{(n)}\mathsf{Adv}(\mathcal{A}) \leq 3 \cdot \mathsf{MatrixAdv}^{(1,\ell;\ell,\ell)}(\mathcal{B}_1) + 2 \cdot \mathsf{MatrixAdv}^{(1,\ell+1;\ell+1,\ell+1)}(\mathcal{B}_2) + 2/p,$$

where $\ell = \lceil 3 \log_2 p \rceil$.

Proof. From Theorems 1, 2, and Corollary 2. □

Corollary 5. *In BHHO encryption scheme, for any $\mathcal{F}_{\mathrm{affine}}$-$KDM^{(n)}$ adversary $\mathcal{A}$, there exists a DDH adversary $\mathcal{B}$, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{KDM}^{(n)}\mathsf{Adv}(\mathcal{A}) \leq (5\ell - 3)\mathsf{DDHAdv}(\mathcal{B}) + 2/p.$$

Proof. From Proposition 1. □

7 Improvement

In this section, we derive a more tight reduction from the matrix DDH assumption to the DDH assumption than Proposition 1. The loss factor is reduced from $O(\ell)$ to $O(\log \ell)$, where ℓ is the size of the matrix. Then both the $\mathrm{KDM}^{(1)}$ security and the $\mathrm{KDM}^{(n)}$ security of BHHO encryption scheme are further improved.

7.1 More Tight Reduction from Matrix DDH to DDH

For an $a \times a$ matrix $\mathbf{A}$ and a $b \times b$ matrix $\mathbf{B}$, define $\mathbf{A} \oplus \mathbf{B}$ as a $(a+b) \times (a+b)$ matrix such that

$$\mathbf{A} \oplus \mathbf{B} = \left(\begin{array}{c|c} \mathbf{A} & 0 \\ \hline 0 & \mathbf{B} \end{array}\right).$$

(See Remark 1.)

Lemma 2. *Let $\ell_1, \ell_2 \geq 2, r \geq 2$ and $1 \leq m \leq \lfloor r/2 \rfloor$. Then for any $\mathcal{A}$, there exists $\mathcal{B}$, running in about the same time as $\mathcal{A}$ such that*

$$\mathsf{MatrixAdv}^{(r-m,r;\ell_1,\ell_2)}(\mathcal{A}) \leq \mathsf{DDHAdv}(\mathcal{B}).$$

Proof. $\mathcal{B}$ takes $[\mathbf{D}]$ as an input, where $\mathbf{D} \xleftarrow{\$} Rk_1(\mathbb{Z}_p^{2\times 2})$ or $\mathbf{D} \xleftarrow{\$} Rk_2(\mathbb{Z}_p^{2\times 2})$. $\mathcal{B}$ first builds an $r \times r$ matrix such that

$$\mathbf{M} = \underbrace{\mathbf{D} \oplus \cdots \oplus \mathbf{D}}_{m \text{ times}} \oplus \mathbf{I}_{r-2m}$$

$\mathcal{B}$ then chooses $\mathbf{L} \xleftarrow{\$} \mathrm{Rk}_r(\mathbb{Z}_p^{\ell_1 \times r})$, $\mathbf{R} \xleftarrow{\$} \mathrm{Rk}_r(\mathbb{Z}_p^{r \times \ell_2})$, and sends a matrix $[\mathbf{L} \cdot \mathbf{M} \cdot \mathbf{R}]$ to $\mathcal{A}$.

If $rank(\mathbf{D}) = 1$, then $\mathbf{L} \cdot \mathbf{M} \cdot \mathbf{R}$ is uniformly distributed over $\mathrm{Rk}_{r-m}(\mathbb{Z}_p^{\ell_1 \times \ell_2})$. If $rank(\mathbf{D}) = 2$, then it is uniformly distributed over $\mathrm{Rk}_r(\mathbb{Z}_p^{\ell_1 \times \ell_2})$. Hence the lemma follows. □

Theorem 5. *Consider r_1, r_2, ℓ_1, ℓ_2 such that $1 \le r_1 < r_2 \le \min(\ell_1, \ell_2)$. Then for any $\mathcal{A}$, there exists $\mathcal{B}$, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{MatrixAdv}^{(r_1,r_2;\ell_1,\ell_2)}(\mathcal{A}) \le \lceil \log_2 r_2 - \log_2 r_1 \rceil \mathsf{DDHAdv}(\mathcal{B}).$$

Proof. There exists unique k such that

$$2^{k-1} \cdot r_1 < r_2 \le 2^k \cdot r_1. \tag{9}$$

For $i = 0, \ldots, k$, let $t_i = 2^i r_1$. Then we have $t_i = 2t_{i-1}$ for $i = 1, \ldots, k-1$, and $t_{k-1} < r_2 \le t_k$.

For $i = 0, \ldots, k-1$, choose $\mathbf{M}_i \xleftarrow{\$} \mathrm{Rk}_{t_i}(\mathbb{Z}_p^{\ell_1 \times \ell_2})$. For k, choose $\mathbf{M}_k \xleftarrow{\$} \mathrm{Rk}_{r_2}(\mathbb{Z}_p^{\ell_1 \times \ell_2})$. Let

$$p_i = \Pr[\mathcal{A}([\mathbf{M}_i]) \textit{ outputs } 1].$$

Then from Lemma 2, we have

$$|p_{i+1} - p_i| = \mathsf{MatrixAdv}^{(t_i,t_{i+1};\ell_1,\ell_2)}(\mathcal{A}) \le \mathsf{DDHAdv}(\mathcal{B}_{i+1})$$

for $i = 0, \ldots, k-2$, and

$$|p_k - p_{k-1}| = \mathsf{MatrixAdv}^{(t_{k-1},r_2;\ell_1,\ell_2)}(\mathcal{A}) \le \mathsf{DDHAdv}(\mathcal{B}_k)$$

Now since $t_0 = r_1$, it holds that

$$\begin{aligned} \mathsf{MatrixAdv}^{(r_1,r_2;\ell_1,\ell_2)}(\mathcal{A}) &= |p_k - p_0| \\ &\le |p_1 - p_0| + \cdots + |p_k - p_{k-1}| \\ &\le k \cdot \mathsf{DDHAdv}(\mathcal{B}) \end{aligned}$$

for some $\mathcal{B}$. Finally from Eq. (9), we have

$$k = \lceil \log_2 r_2 - \log_2 r_1 \rceil$$

Therefore the theorem follows. □

7.2 Application to the KDM Security of BHHO

Corollary 6. *In BHHO encryption scheme, for any $\mathcal{A}$, there exists $\mathcal{B}$, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{KDM}^{(1)}\mathsf{Adv}(\mathcal{A}) \le 4 \cdot \lceil \log_2(\ell+1) \rceil \mathsf{DDHAdv}(\mathcal{B}) + 2/p.$$

Proof. From Corollary 2 and Theorem 5. □

Corollary 7. *In BHHO encryption scheme, for any $\mathcal{A}$, there exists $\mathcal{B}$, running in about the same time as $\mathcal{A}$, such that*

$$\mathsf{KDM}^{(n)}\mathsf{Adv}(\mathcal{A}) \le 5 \cdot \lceil \log_2(\ell+1) \rceil \mathsf{DDHAdv}(\mathcal{B}) + 2/p.$$

Proof. From Corollary 4 and Theorem 5. □

References

1. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_35
2. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: quadratic residuosity strikes back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_1
3. Brakerski, Z., Goldwasser, S., Kalai, Y.T.: Black-box circular-secure encryption beyond affine functions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 201–218. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_13
4. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_7
5. Galindo, D., Herranz, J., Villar, J.: Identity-based encryption with master key-dependent message security and leakage-resilience. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 627–642. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33167-1_36
6. Malkin, T., Teranishi, I., Yung, M.: Efficient circuit-size independent public key encryption with KDM security. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 507–526. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_28
7. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005, pp. 84–93 (2005)
8. Wee, H.: KDM-security via homomorphic smooth projective hashing. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9615, pp. 159–179. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49387-8_7

From Identification Using Rejection Sampling to Signatures via the Fiat-Shamir Transform: Application to the BLISS Signature

Pauline Bert(✉) and Adeline Roux-Langlois

Univ Rennes, CNRS, IRISA, Rennes, France
pauline.bert@irisa.fr

Abstract. In this paper, we present a reduction from non-lossy/lossy identification scheme using rejection sampling to signature in the Random Oracle Model (ROM). The rejection sampling is used to ensure that the last step in the identification scheme does not leak information about the secret key of the scheme. This last step may fail, and to hide these failures to an adversary we use a Fiat-Shamir transform where we rerun the identification protocol until we get a valid output. We also apply our result for non-lossy identification scheme to the well-known BLISS signature [DDLL13] and compare with the original proof.

Keywords: Signature schemes · Identification schemes
Fiat-Shamir transform · Rejection sampling · Lattices

1 Introduction

The Fiat-Shamir transform [FS86] is a well-studied transform from an identification scheme to a digital signature. In the lattice literature, Fiat-Shamir signatures are probably the most efficient ones [Lyu12, GLP12, DDLL13], compared to hash-and-sign, or even standard model signatures. In this paper, we propose a reduction where almost every Fiat-Shamir transform on lattices can fit into, and we apply our reduction to the BLISS signature [DDLL13].

From Identification to Signature. An identification scheme $\mathcal{ID}$ is a three-move protocol Commitment-Challenge-Response. The prover, using its secret key, sends a commitment Cmt to the verifier. The verifier responds a random challenge Ch. The prover finally sends a response Rsp. The verifier, having access to the corresponding public key, accepts or not the complete transcript Cmt||Ch||Rsp. The Fiat-Shamir (FS) transform [FS86] is a way to construct a digital signature scheme in the Random Oracle Model (ROM) from an identification scheme. The signer runs the identification scheme by itself by choosing the challenge via a hash function $\text{Ch} \leftarrow H(\text{Cmt}, m)$. The signature of a message m is $\sigma = (\text{Cmt}, \text{Rsp})$, and to

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 297–312, 2018.
https://doi.org/10.1007/978-3-319-97916-8_19

verify such a signature, we recompute the challenge $\textsc{Ch} \leftarrow H(\textsc{Cmt}, m)$ and check whether $\textsc{Cmt}||\textsc{Ch}||\textsc{Rsp}$ is a valid transcript or not.

Security. A basic security for an identification scheme is the security against passive impersonation. The adversary, often called the impersonator, has access to the public key of the scheme and to a transcript generation oracle [AABN02]. This oracle, depending on the identification scheme $\mathcal{ID}$ and on a key pair (pk, sk), outputs a random transcript of a honest execution. The goal of the adversary is to impersonate the prover. By interacting with an honest verifier, the adversary wants the verifier to accept at the end of the execution of the protocol. The signature scheme obtained by applying the FS transform is secure against chosen-message attack in the ROM if, and only if, the underlying identification scheme is secure against impersonation under passive attack (and non-trivial meaning that the challenge space is super-polynomial) [AABN02]. This transformation is not tight, it looses a factor at least q_H (number of queries to the random oracle H) in the advantage of the impersonator compared to the advantage of the forger.

Lossy/Non-Lossy Identification Scheme. Motivating by the work of Katz and Wang [KW03], Abdalla et al. in [AFLT12] introduced the idea of lossy identification scheme and give a tight analogous of the reduction of [AABN02], starting from a lossy identification scheme. A lossy identification scheme comes with an additional lossy key generation algorithm which outputs a lossy public key, which is computationally indistinguishable from a honestly generated one. Such scheme has also a property of simulatability, meaning that we can construct a simulated transcript generation oracle, with no access to the secret key of the identification scheme but still outputs transcripts whose distribution is statistically close to those from the original transcript generation oracle. The security of a lossy identification scheme is a notion of impersonation with respect to lossy keys, where the adversary has access to a lossy public key of the scheme and to the simulated variant of the transcript generation oracle. In the lattice literature, we can find some lossy identification schemes: the lattice instantiation at the end of [AFLT12], the underlying identification scheme of NIST submissions TESLA, and Dilithium. In [ABB+17], the authors showed that the TESLA signature is secure in the Quantum Random Oracle Model (QROM) and recently in [KLS17], the proof is generalized to Fiat-Shamir signature starting from lossy identification scheme with an application to Dilithium. There exist also non-lossy lattice-based identification schemes, for example the ones underlying the signatures [Lyu12, DDLL13]. To prove the security of such schemes, a solution is to use the Forking Lemma [PS00, BN06], resulting in a non-tight proof.

Rejection Sampling. The use of rejection sampling in lattice constructions is due to Lyubashevsky [Lyu08, Lyu12]. He first describes an abort technique, allowing the prover to abort the protocol instead of returning its response. The idea behind the abort technique is to shorten the response by allowing it to fall in a smaller space/interval. This will happen with small probability, and if it does the prover simply aborts the protocol. When we construct a signature via

the Fiat-Shamir transform from such identification scheme, the aborts can be hidden by simply rerun the protocol. The abort/rejection sampling technique is also used to ensure that the response of the prover is independent from its secret key. Rejection sampling is a method to sample from an arbitrary distribution f, given the access to a family of probability distributions g_v indexed by some v. A sample $x \xleftarrow{\$} g$ drawn from g is accepted with probability $\frac{f(x)}{M \cdot g_v(x)}$ where M is a constant satisfying $M \cdot g_v(x) \geq f(x)$ for all x drawn from f. This procedure succeeds with probability at least $\frac{1}{M}$.

Identification Scheme using Rejection Sampling. In an identification scheme using rejection sampling, the probability distribution f corresponds to the target output distribution of the prover responses. Unlike the distribution f, the family of probability distribution g_v will depend on the prover secret key; and will be indexed by a random value v, being a function of the prover secret key and a uniformly random challenge. For example in [Lyu12], the distribution of the responses f is a known discrete Gaussian distribution D_σ^m of parameter σ. However, the distribution $g_{\mathbf{v}}$ is a shifted discrete Gaussian distribution $D_{\mathbf{v},\sigma}^m$ with the same parameter σ but centered on a vector $\mathbf{v} = \mathbf{Sc}$ depending on the prover secret key $\mathbf{S}$ and on a particular uniformly random challenge $\mathbf{c}$.

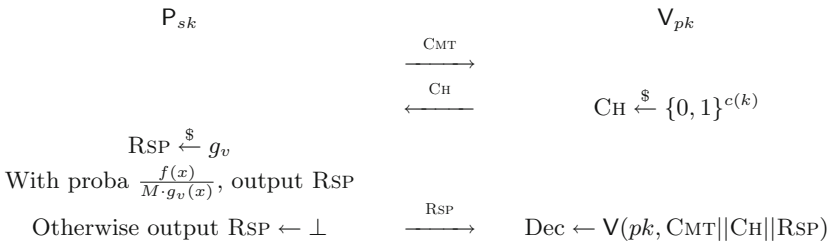


Fig. 1. Identification scheme using Rejection sampling

Our Contribution. In this paper, we give a definition of an identification scheme using rejection sampling starting from the definition of rejection sampling from [Lyu12]. This kind of identification scheme has two inherent and quite classical properties:

1. *Correctness Error:* The probability that a honestly generated transcript contains a non-valid response is negligible, here it corresponds to $\left(1 - \frac{1}{M}\right)$.
2. *Simulatability:* There exists a simulated transcript generation oracle, who does not have access to the secret key and is able to output transcripts statistically close to those from the original transcript generation oracle.

The security we consider for such identification scheme is the impersonation against passive attacks where the adversary has access to the real public key of the scheme and also to the simulated transcript oracle.

Our main result is a transformation from an identification scheme using rejection sampling to an existentially unforgeable signature in the ROM. This signature is obtained by applying the Fiat-Shamir transform on an identification scheme using rejection sampling. The only significant modification from existing Fiat-Shamir transform [AFLT12,KLS17] is that we repeat the execution of the identification protocol in the signing algorithm as long as the response of the prover is non-valid. We then discuss whether or not the identification scheme is also lossy. If the identification scheme is lossy, we get a tight proof as in [AFLT12] and if not, we get a non-tight proof loosing a factor of roughly q_H as in [AABN02]. To link the advantage of the impersonator and the advantage against the underlying search problem, we use a propriety of soundness, i.e. if we have access to two valid transcripts on a same commitment, we can extract a solution of a instance of this search problem. Next, by using the Reset Lemma [BP02] we can link the advantage of the impersonator playing the impersonation experiment to the advantage of an adversary playing twice this experiment with different randomness and getting two valid transcripts on a same commitment.

We give an example of this by applying our main result to the well-known BLISS signature [DDLL13] with its underlying non-lossy identification scheme. We choose the BLISS signature because the construction follows exactly our Fiat-Shamir transform (i.e. rerun the identification scheme in the signing algorithm to get a valid response) and we remark that the BLISS paper does not take into account this feature in the proof.

Overview of Our Main Result. The idea behind this proof is to use honest transcripts of the identification scheme to answer the signing queries of the forger. If we have a valid transcript $\textsc{Cmt}||\textsc{Ch}||\textsc{Rsp}$, we will set the random oracle $H(\textsc{Cmt}||m) \leftarrow \textsc{Ch}$ to ensure that $\sigma = (\textsc{Cmt}, \textsc{Rsp})$ is a valid signature for the message m. The first step of our proof is to limit the number of signing attempts to l, where we can take l greater than M. Doing this modification implies that the forger might see invalid signatures, when after l tries, the response of the prover is non-valid. This happens for each signing query, so the probability that a signature is non-valid is at most $q_S(1 - \frac{1}{M})^l$ where $\left(1 - \frac{1}{M}\right)$ is the probability that an honestly generated transcript contains a non-valid response.

On sign query m, we may overwrite the value $H(\textsc{Cmt}, m)$. Such collisions happen with probability at most $\frac{l(q_S+q_H+1)q_S}{2^\beta}$ where β corresponds to the min-entropy of commitments.

Then we apply a series of small changes to get a signing algorithm that no longer needs the secret key sk. To do that the major change is to switch from the transcript generation algorithm to its simulated counterpart thanks to the use of rejection sampling. If the statistical distance between the distribution of the transcripts outputted by this two oracles is at most ε_{rs}, the advantage of the forger changes by at most $q_S\varepsilon_{rs}$ due to the rejection sampling technique.

Non-Lossy Identification Scheme. If the identification is non-lossy, we can link the advantage of the forger to the advantage of the impersonator. For this step, we make a guess about which hash query will be used in the forgery. If our

guess is correct, we are able to break the underlying impersonation problem, that's why we loose at least a factor q_H in the advantage of the impersonator compared to the advantage of the forger, like [AABN02].

Lossy Identification Scheme. If the identification is lossy, we can add another step which involves switching the real public key of the scheme to a lossy one. The advantage of the adversary is modified by at most the advantage in distinguishing a real public key from a lossy one. The reduction is tight because the advantage of the forger is tightly related to the advantage of breaking the underlying search problem, for example the decision-LWE problem in [AFLT12]. Finally, the last step link the advantage of the forger to the advantage of the impersonator with respects to lossy keys as with non-lossy identification scheme.

2 Preliminaries

Notation. Let $\mathrm{A}(\cdot,\cdot,\cdots)$ be a randomized algorithm, then $x \leftarrow \mathrm{A}(a,b,\cdots;R)$ is the unique output on inputs $a,b,\cdots$ and coins R, while $x \xleftarrow{\$} \mathrm{A}(a,b,\cdots)$ means that we first pick a random $R \xleftarrow{\$} \mathrm{Coins}(k)$ and then assigned $x \leftarrow \mathrm{A}(a,b,\cdots;R)$.

2.1 Identification Scheme Using Rejection Sampling

To hide the secret key of a prover in a identification scheme, Lyubashevsky [Lyu12] proposed to use a rejection technique. Informally, the prover generates a candidate for its response and rejects it with a certain probability to ensure that the distribution of the response is independent from the prover secret key.

Lemma 1 (Rejection Sampling [Lyu12]). *Let V be an arbitrary set, $h : V \to \mathbb{R}$ and $f : \mathbb{Z}^m \to \mathbb{R}$ be probability distributions. If $g_v : \mathbb{Z}^m \to \mathbb{R}$ is a family of probability distributions indexed by all $v \in V$ with the property that there exists a constant $M \in \mathbb{R}$ such that $\forall v, Pr\left[M \cdot g_v(x) \geq f(x), x \xleftarrow{\$} f\right] \geq 1 - \varepsilon_{rs}$, then, the output distribution of*

$v \xleftarrow{\$} h$
$x \xleftarrow{\$} g_v$
return (x,v) *with probability* $\min\left(\frac{f(x)}{M \cdot g_v(x)}, 1\right)$

is within statistical distance ε_{rs}/M of the output distribution of

$v \xleftarrow{\$} h$
$x \xleftarrow{\$} f$
return (x,v) *with probability* $1/M$

Moreover, the probability p_{out} that the first algorithm output something is bounded by $(1-\varepsilon_{rs})/M \leq p_{out} \leq 1/M$.

ID. An identification scheme using rejection sampling (see Fig. 1) is a classical one using rejection sampling to ensure that responses follow a probability distribution f, by first generate then following a family of probability g_v.

Definition 1. *An identification scheme using rejection sampling $\mathcal{ID}$ is defined by $\mathcal{ID} = (\mathsf{KeyGen}, \mathsf{P}, \mathsf{V}, c, g_v, f)$ where:*

- $\mathsf{KeyGen}(1^k)$ *is the key generation algorithm, taking the security parameter $k \in \mathbb{N}$ of the scheme and outputting a pair of keys (pk, sk). The secret key sk is given to the prover algorithm P, and the public pk is given to the verifier algorithm V.*
- P *is the prover algorithm, which takes as input the secret key sk and the current conversation transcript and outputs the next message to be sent to the verifier.*
- V *is a deterministic algorithm which takes as input the public key pk and the complete transcript conversation* Cmt||Ch||Rsp *and outputs a boolean decision* Dec.
- $c(k)$ *is a function of the security parameter k, which corresponds to the length of the challenge.*
- g_v *is a family of probability distributions indexed by v, a function of the secret key sk, a particular challenge* Ch *and in some case v can also depend on a particular commitment* Cmt *or on the secret used to construct the commitment,*
- f *is the output distribution of the prover responses such that there exists a constant $M \in \mathbb{R}$ verifying $\forall\, \text{Ch} \in \{0,1\}^{c(k)}, \forall\, x, M \cdot g_v(x) \geq f(x)$.*

Transcript Generation Oracle. Like in [AABN02, AFLT12], we associate a transcript generation oracle $\text{Tr}^{\mathcal{ID}}_{pk,sk,k}$ to an identification scheme $\mathcal{ID}$. The transcript generation oracle $\text{Tr}^{\mathcal{ID}}_{pk,sk,k}$ returns a random transcript Cmt||Ch||Rsp of an honest execution of $\mathcal{ID}$ with key pair (pk, sk) and security parameter k. In an identification scheme using rejection sampling, the prover may output a response $\text{Rsp} = \perp$, in this case the transcript generation oracle will output $\perp||\perp||\perp$.

$\text{Tr}^{\mathcal{ID}}_{pk,sk,k}$:

Cmt $\xleftarrow{\$}$ $\mathsf{P}(sk)$,
Ch $\xleftarrow{\$}$ $\{0,1\}^{c(k)}$, Rsp $\xleftarrow{\$}$ g_v
return Cmt||Ch||Rsp with probability $\frac{f(x)}{M \cdot g_v(x)}$, otherwise $\perp||\perp||\perp$.

Inherent Properties. Thanks to the rejection sampling (Lemma 1) we can simulate the transcript generation oracle $\text{Tr}^{\mathcal{ID}}_{pk,sk,k}$ by an algorithm $\tilde{\text{Tr}}^{\mathcal{ID}}_{pk,k}$ with no access to the secret key sk. It proceeds by first generating Cmt $\xleftarrow{\$}$ $\mathsf{P}(sk)$ and Ch $\xleftarrow{\$}$ $\{0,1\}^{c(k)}$ and outputting Cmt||Ch||Rsp with Rsp $\xleftarrow{\$}$ f with probability $\frac{1}{M}$, and otherwise $\perp||\perp||\perp$.

This property is called Non-Abort Honest-Verifier Zero-Knowledge (naHVZK) in [KLS17].

Definition 2 (naHVZK). *$\mathcal{ID}$ is said to be ϵ-perfect naHVZK if there exists an algorithm $\tilde{\mathrm{Tr}}^{\mathcal{ID}}_{pk,k}$, given only the public key pk and the security parameter k, outputs* Cmt||Ch||Rsp *such that the following conditions hold:*

1. *The distribution of* Cmt||Ch||Rsp $\xleftarrow{\$} \tilde{\mathrm{Tr}}^{\mathcal{ID}}_{pk,k}$ *has statistical distance at most ϵ from* Cmt′||Ch′||Rsp′ $\xleftarrow{\$} \mathrm{Tr}^{\mathcal{ID}}_{pk,sk,k}$,
2. *The distribution of* Ch *from* Cmt||Ch||Rsp $\xleftarrow{\$} \tilde{\mathrm{Tr}}^{\mathcal{ID}}_{pk,k}$ *is uniform in the challenge set* $\{0,1\}^{c(k)}$.

Our identification scheme also satisfy the correctness property from [KLS17], with $\varepsilon_c = 1 - 1/M$.

Definition 3 (Correctness Error). *An identification scheme $\mathcal{ID}$ has correctness error ε if for all $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$ the following holds:*

1. *All possible transcripts* Cmt||Ch||Rsp *satisfying* Rsp $\neq \perp$ *are valid,*
2. *The probability that a honestly generated transcript* Cmt||Ch||Rsp *contains* Rsp $= \perp$ *is bounded by ε.*

Security. The security of the identification scheme we consider here is a security against passive impersonation where the goal of the adversary is to impersonate the prover without the knowledge of the secret key sk. This impersonator is modeled as a probabilistic algorithm $\mathcal{I}$ which is given as input the public key pk of the identification scheme and also has access to the simulation of the transcript oracle $\tilde{\mathrm{Tr}}^{\mathcal{ID}}_{pk,k}$ described above. After looking at these transcripts, the impersonator $\mathcal{I}$ interacts with an honest verifier in the three-move protocol and wants the verifier to accept at the end of this protocol.

$\mathbf{Exp}^{\text{imp-pa-sim}}_{\mathcal{ID},\mathcal{I}}(k)$:

$(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$, $st \| \text{Cmt} \xleftarrow{\$} \mathcal{I}^{\tilde{\mathrm{Tr}}^{\mathcal{ID}}_{pk,k}}(pk)$
Ch $\xleftarrow{\$} \{0,1\}^{c(k)}$, Rsp $\xleftarrow{\$} \mathcal{I}(st, \text{Ch})$, Dec $\leftarrow \mathsf{V}(pk, \text{Cmt}\|\text{Ch}\|\text{Rsp})$
return Dec

The advantage of $\mathcal{I}$ playing the game above is

$$\mathbf{Adv}^{\text{imp-pa-sim}}_{\mathcal{ID},\mathcal{I}}(k) = \Pr\left[\mathbf{Exp}^{\text{imp-pa-sim}}_{ID,\mathcal{I}}(k) = 1\right].$$

An $\mathcal{ID}$ is polynomially-secure against impersonation under passive attack if $\mathbf{Adv}^{\text{imp-pa-sim}}_{\mathcal{ID},\mathcal{I}}(\cdot)$ is negligible for every poly(k)-time impersonator $\mathcal{I}$.

2.2 Lossy Identification Scheme Using Rejection Sampling

A lossy identification scheme using rejection sampling is defined like a classical identification scheme plus an algorithm $\mathsf{LossyKeyGen}(1^k)$ which takes the security parameter $k \in \mathbb{N}$ and outputs a lossy public key pk. We will replace a truly

generated public key in $\mathbf{Exp}^{\text{imp-pa-sim}}_{\mathcal{ID},\mathcal{I}}(k)$ by a lossy one in the impersonation experiment with respect to lossy keys $\mathbf{Exp}^{\text{los-imp-pa}}_{\mathcal{ID},\mathcal{I}}(k)$ and we have no need of a secret key in this case. A lossy identification scheme satisfies two properties, a simulatability property like the naHVZK defined above and the following one:

Definition 4 (Indistinguishability of keys). *Consider the two experiments* $\boldsymbol{Exp}^{\textit{ind-keys-real}}_{\mathcal{ID},\mathcal{D}}(k)$ *and* $\boldsymbol{Exp}^{\textit{ind-keys-lossy}}_{\mathcal{ID},\mathcal{D}}(k)$ *in which we respectively generate pk via* $\mathsf{KeyGen}(1^k)$ *and via* $\mathsf{LossyKeyGen}(1^k)$*, and provide it as input to the distinguishing algorithm* $\mathcal{D}$*. We say that* $\mathcal{D}$ *can* (t,ε)*-solve the key-indistinguishability problem if* $\mathcal{D}$ *runs in time* t *and*

$$\left| Pr\left[\boldsymbol{Exp}^{\textit{ind-keys-real}}_{\mathcal{ID},\mathcal{D}}(k)=1\right] - Pr\left[\boldsymbol{Exp}^{\textit{ind-keys-lossy}}_{\mathcal{ID},\mathcal{D}}(k)\right]\right| \geq \varepsilon.$$

We say that $\mathcal{ID}$ *is* (t,ε)*-key-indistinguishable if no algorithm* (t,ε)*-solve the key-indistinguishability problem.*

Min-Entropy of Commitments. Let $\mathcal{C}(sk) = \{\mathsf{P}(sk;R) : R \in \text{Coins}(k)\}$ be the set of commitments associated to sk, where $\text{Coins}(k)$ is a set of binary string depending on the security parameter k. The maximum probability that a commitment takes a particular value is:

$$\alpha(sk) = \max_{\text{CMT}\in\mathcal{C}(sk)} \left\{ \Pr\left[\mathsf{P}(sk;R) = \text{CMT} : R \xleftarrow{\$} \text{Coins}(k)\right]\right\}.$$

Then, the min-entropy function associated to $\mathcal{ID}$ is $\beta(sk) = \min_{sk}\left\{\log_2 \frac{1}{\alpha(sk)}\right\}$, where the minimum is taken over all the (pk, sk) generated by $\mathsf{KeyGen}(1^k)$.

2.3 Reset Lemma

Here we recall the Reset Lemma from [BP02] which apply to identification scheme in the same way the Forking Lemma [PS00, BN06] applies to signature scheme.

Lemma 2 (Reset Lemma [BP02]). *Let* P *be a prover in a canonical identification scheme with verifier* V *and let* q, v *be inputs for the prover and verifier respectively. Let* $\text{acc}(p,v)$ *be the probability that* V *accepts after its interaction with* P*, i.e. the probability that the following experiment returns 1:*

$R \xleftarrow{\$} \text{Coins}(k)$, $st||\text{CMT} \leftarrow \mathsf{P}(p;R)$
$\text{CH} \xleftarrow{\$} \{0,1\}^{c(k)}$, $\text{RSP} \xleftarrow{\$} \mathsf{P}(st,\text{CH})$, $\text{Dec} \leftarrow \mathsf{V}(v,\text{CMT}||\text{CH}||\text{RSP})$
return Dec

Let $\text{res}(q,v)$ *be the probability that the following reset experiment outputs 1:*

$R \xleftarrow{\$} \text{Coins}(k)$, $st||\text{CMT} \leftarrow \mathsf{P}(p;R)$
$\text{CH}_1 \xleftarrow{\$} \{0,1\}^{c(k)}$, $\text{RSP}_1 \xleftarrow{\$} \mathsf{P}(st,\text{CH}_1)$, $\text{Dec}_1 \leftarrow \mathsf{V}(v,\text{CMT}||\text{CH}_1||\text{RSP}_1)$
$\text{CH}_2 \xleftarrow{\$} \{0,1\}^{c(k)}$, $\text{RSP}_2 \xleftarrow{\$} \mathsf{P}(st,\text{CH}_2)$, $\text{Dec}_2 \leftarrow \mathsf{V}(v,\text{CMT}||\text{CH}_2||\text{RSP}_2)$
return $\text{Dec}_1 \wedge \text{Dec}_2 \wedge \text{CH}_1 \neq \text{CH}_2$

Then $\text{acc}(q,v) \leq \frac{1}{2^{c(k)}} + \sqrt{\text{res}(q,v)}$.

2.4 Lattice Background

Lattices. An m-dimensional full-rank lattice Λ is a discrete additive subgroup of $\mathbb{R}^m$. A lattice is the set of all integer combinations of some linearly independent basis vectors, $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_m\} \in \mathbb{R}^{m \times m}$, $\Lambda(\mathbf{B}) = \{\sum_{i=1}^{m} z_i \mathbf{b}_i : z_i \in \mathbb{Z}\}$.

Gaussian Distribution. The continuous Gaussian distribution of center $\mathbf{c} \in \mathbb{R}^m$ and width parameter σ is defined as $\rho^m_{\mathbf{c},\sigma}(\mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{\|\mathbf{x}-\mathbf{c}\|^2}{2\sigma^2})$. The discrete Gaussian distribution over the lattice $\mathbb{Z}^m$ is defined as $D^m_{\mathbf{c},\sigma} = \frac{\rho^m_{\mathbf{c},\sigma}(\mathbf{x})}{\rho^m_{\sigma}(\mathbb{Z}^m)}$ where $\rho^m_\sigma(\mathbb{Z}^m) = \sum_{\mathbf{x} \in \mathbb{Z}^m} \rho^m_\sigma(\mathbf{x})$.

Lemma 3 ([Lyu12]). *For any $\eta > 1$, $Pr_{\mathbf{z} \leftarrow D^m_\sigma}[\|\mathbf{z}\| > \eta\sigma\sqrt{m}] < \eta^m \exp^{\frac{m}{2}(1-\eta^2)}$.*

SIS. A classical hard problem in lattice based literature is the Short Integer Solution (SIS) problem, introduced by Ajtai [Ajt96] where he also gives a reduction from worst-case lattice problems to the average-case SIS problem.

Definition 5 ($\mathbf{SIS}_{q,n,m,\beta}$). *Given an uniformly random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, find a non-zero vector $\mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{A}\mathbf{x} = 0 \mod q$ and $0 < \|\mathbf{x}\| \leq \beta$.*

3 Signature Scheme Using Rejection Sampling

In this part, we will describe formally the Fiat-Shamir transform we use to construct a digital signature from our definition of an identification scheme using rejection sampling. Will we show that applying this Fiat-Shamir to such identification scheme gives us a secure digital signature in the ROM.

3.1 Fiat-Shamir Transform

Definition 6. *Let $\mathcal{ID} = (\mathsf{KeyGen}, \mathsf{P}, \mathsf{V}, c, g_v, f)$ be an identification scheme using rejection sampling, and $H : \{0,1\}^* \rightarrow \{0,1\}^{c(k)}$ be a hash function modeled as a random oracle, then we can construct a signature $\mathcal{DS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. The signature has the same key generation algorithm as the identification scheme, and the output length of the hash function equals the challenge length. The signing and verifying algorithms are defined as follows:*

$\mathsf{Sign}(sk, m)$:
while $\textsc{Rsp} = \perp$ ***do***
$\textsc{Cmt} \leftarrow \mathsf{P}(sk)$
$\textsc{Ch} \leftarrow H(\textsc{Cmt}, m)$
$\textsc{Rsp} \xleftarrow{\$} g_v$
return $\sigma = (\textsc{Cmt}, \textsc{Rsp})$ *with probability* $\frac{f(x)}{M \cdot g_v(x)}$, *otherwise*
$\textsc{Rsp} \leftarrow \perp$

$\mathsf{Verify}(pk, m, \sigma)$:
parse σ *as* $(\textsc{Cmt}, \textsc{Rsp})$
$\textsc{Ch} \leftarrow H(\textsc{Cmt}, m)$
return $\mathsf{V}(pk, \textsc{Cmt}||\textsc{Ch}||\textsc{Rsp})$

3.2 General Result from Non-Lossy Identification Scheme

Our first and main result gives us a reduction from non-lossy identification scheme using rejection sampling to an existential unforgeable secure signature in the ROM by supposing that the underlying ID scheme satisfies the two properties naHVZK and correctness error defined in Definitions 2 and 3.

Theorem 1. *Let* $\mathcal{ID} = (\mathsf{KeyGen}, \mathsf{P}, \mathsf{V}, c, g_v, f)$ *be an identification scheme using rejection sampling whose commitment space has min-entropy* $\beta(k)$*, let* $H : \{0,1\}^* \rightarrow \{0,1\}^{c(k)}$ *be a hash function modeled as a random oracle, and let* $\mathcal{DS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be the associated signature as in Definition 6. If* $\mathcal{ID}$ *is* ε_{rs}*-perfect naHVZK, has correctness error* ε_c *and is secure against impersonation under passive attacks then* $\mathcal{DS}$ *is existentially unforgeable secure against adaptive chosen-message attack in the random oracle model such that:*

$$\boldsymbol{Adv}^{uf\text{-}cma}_{\mathcal{DS},\mathcal{F}}(k) \leq (q_H + 1)\boldsymbol{Adv}^{imp\text{-}pa\text{-}sim}_{\mathcal{ID},\mathcal{I}}(k) + q_S\varepsilon_{rs} + \frac{l(q_S + q_H + 1)q_S}{2^\beta} + q_S\varepsilon_c^l.$$

Furthermore, if $\mathcal{I}$ *runs in at most time* t'*, then* $\mathcal{F}$ *runs in times* $t = t' - O(q_S t_{\text{Sign}})$*, where* t_{Sign} *designed the average time of the signing algorithm.*

Proof. This proof uses code-based game-playing à la [AABN02, AFLT12], by constructing a sequence of experiments $\mathbf{Exp}_0, \cdots, \mathbf{Exp}_7$ starting with the experiment catching the existentially unforgeability of the signature scheme. We defined δ_i as the event that experiment $\mathbf{Exp}_i$ returns 1, i.e. that the adversary $\mathcal{F}$ outputs a valid forgery. We will assume that before outputting a forgery $(m^*, \sigma^* = (\textsc{Cmt}^*, \textsc{Rsp}^*))$, $\mathcal{F}$ already queried the corresponding hash query on $\textsc{Cmt}^*, m^*$, which increase the number of hash query by one.

$\mathbf{Exp}_0$. In this first experiment, the challenger generates the pair of keys $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$, sets the hash counter hc and the sign counter sc to zero, and also initializes the set of queried messages $\mathcal{M}$ to empty in **Initialize** and returns the public key pk to $\mathcal{F}$. On hash query $\textsc{Cmt}, m$, the challenger checks if $H(\textsc{Cmt}, m)$ has already been set. If $H(\textsc{Cmt}, m) = \perp$, the counter hc is incremented by one, the challenger chooses a random challenge $\textsc{Ch} \xleftarrow{\$} \{0,1\}^{c(k)}$ and sets $H(\textsc{Cmt}, m) \leftarrow \textsc{Ch}$. The challenger finally outputs $H(\textsc{Cmt}, m)$. On sign query m, the counter sc is incremented by one, the queried message m is added to the set $\mathcal{M}$ and the challenger computes the signature $\sigma = (\textsc{Cmt}, \textsc{Ch})$ as in the signing algorithm. During the signing phase, the challenger also checks if $H(\textsc{Cmt}, m) = \perp$, if so it performs the same steps as for an hash query on $\textsc{Cmt}, m$. Finally, when $\mathcal{F}$ outputs a forgery (m^*, σ^*), the challenger runs $\text{Dec} \leftarrow \mathsf{V}(pk, \textsc{Cmt}^* || \textsc{Ch}^* || \textsc{Rsp}^*)$ and returns $\text{Dec} \wedge (m^* \notin \mathcal{M})$. By definition, $\Pr[\delta_0] = \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS},\mathcal{F}}(k)$.

$\mathbf{Exp}_1$. Let bad be a boolean variable initialize to false. In $\mathbf{Exp}_1$, we limit the number of signing attempts to l, with $l \geq M$ in practice. We also set bad to true if after l signing attempts, we do not output a valid signature. For each signature, the probability that $\textsc{Rsp} = \perp$ after at most l attempts is equal to ε_c^l

Initialize:
1: $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$
2: $hc \leftarrow 0$, $sc \leftarrow 0$, $\mathcal{M} \leftarrow \{\}$
3: **return** pk

On Hash-query $\textsc{Cmt}, m$:
1: **if** $H(\textsc{Cmt}, m) = \perp$ **then**
2: $\quad hc \leftarrow hc + 1$
3: $\quad \textsc{Ch} \xleftarrow{\$} \{0,1\}^{c(k)}$
4: $\quad H(\textsc{Cmt}, m) \leftarrow \textsc{Ch}$
5: **return** $H(\textsc{Cmt}, m)$

Finalize(m^*, σ^*):
1: Parse σ^* as $\textsc{Cmt}^*, \textsc{Rsp}^*$
2: $\textsc{Ch}^* \leftarrow H(\textsc{Cmt}^*, m^*)$
3: $\text{Dec} \leftarrow \mathsf{V}(pk, \textsc{Cmt}^* || \textsc{Ch}^* || \textsc{Rsp}^*)$
4: **return** $\text{Dec} \wedge (m^* \notin \mathcal{M})$

On Sign-query m:
1: $sc \leftarrow sc + 1$, $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
2: $\boxed{ctr \leftarrow 0}$
3: **while** $\textsc{Rsp} = \perp$ $\boxed{\text{and } ctr \leq l}$ **do**
4: $\quad \boxed{ctr \leftarrow ctr + 1}$
5: $\quad \textsc{Cmt} \leftarrow \mathsf{P}(sk)$
6: $\quad$ **if** $H(\textsc{Cmt}, m) = \perp$ **then**
7: $\quad\quad hc \leftarrow hc + 1$
8: $\quad\quad \textsc{Ch} \xleftarrow{\$} \{0,1\}^{c(k)}$
9: $\quad\quad H(\textsc{Cmt}, m) \leftarrow \textsc{Ch}$
10: $\quad \textsc{Ch} \leftarrow H(\textsc{Cmt}, m)$
11: $\quad \textsc{Rsp} \xleftarrow{\$} g_v$
12: $\quad$ **return** $\sigma = (\textsc{Cmt}, \textsc{Rsp})$ with probability $\frac{f(x)}{M \cdot g_v(x)}$, otherwise $\textsc{Rsp} \leftarrow \perp$
13: **if** $\boxed{\textsc{Rsp} = \perp}$ **then**
14: $\quad \boxed{\text{bad} \leftarrow \text{true}}$
15: $\quad$ **return** $\boxed{\sigma = (\perp, \perp)}$

Fig. 2. $\mathbf{Exp}_0$ and $\boxed{\mathbf{Exp}_1}$

where we recall that ε_c corresponds to the error correctness of the scheme. We get $|\Pr[\delta_1] - \Pr[\delta_0]| \leq \sum_{i=1}^{q_S} \Pr[\text{bad} = \text{true in the } i\text{-th sign query}] = q_S \varepsilon_c^l$ (Fig. 2).

$\mathbf{Exp}_2$. In this experiment, the challenger no longer sets bad to true due to a non-valid signature. This modification does not change the output of the experiment, we have $\Pr[\delta_2] = \Pr[\delta_1]$.

$\mathbf{Exp}_3$. In this experiment, on sign query m, the challenger sets bad to true if the value $H(\textsc{Cmt}, m)$ has already been defined. If bad is set, the challenger also chooses a new value $\textsc{Ch} \xleftarrow{\$} \{0,1\}^{c(k)}$ and overwrites the old value $H(\textsc{Cmt}, m) \leftarrow \textsc{Ch}$. To compute the probability of bad sets to true, we assume that all the hash queries have been already ask at the beginning of the experiment. The probability that during the i-th signing query bad sets to true is $(l(i-1) + q_H + 1)/2^\beta$ where we recall that l is the number of signing attempts we introduce in $\mathbf{Exp}_1$. So we get $|\Pr[\delta_3] - \Pr[\delta_2]| \leq \sum_{i=1}^{q_S} \Pr[\text{bad} = \text{true in the } i\text{-th sign query}] \leq \frac{l(q_S + q_H + 1)q_S}{2^\beta}$.

$\mathbf{Exp}_4$. In this experiment, the challenger no longer sets bad to true due to an overwriting of the value $H(\textsc{Cmt}, m)$. This modification does not change the output of the experiment, we have $\Pr[\delta_4] = \Pr[\delta_3]$.

$\mathbf{Exp}_5$. In the previous experiment, to answer signing query, the challenger generates a new uniform challenge $\textsc{Ch} \xleftarrow{\$} \{0,1\}^{c(k)}$ like in the transcript generation

oracle $\mathrm{Tr}^{\mathcal{ID}}_{pk,sk,k}$. So in this experiment, we change the simulation of the signing algorithm such that the value $\textsc{Cmt}||\textsc{Ch}||\textsc{Rsp}$ are generated thanks to $\mathrm{Tr}^{\mathcal{ID}}_{pk,sk,k}$. This change does not affect the output of the game, $\Pr[\delta_5] = \Pr[\delta_4]$.

Exp$_6$. We can go further, by generating all the transcripts needed to answer signing queries at the beginning of the experiment which does not change the output of the experiment, $\Pr[\delta_6] = \Pr[\delta_5]$.

Exp$_7$. In the last experiment, we replace the transcript generation oracle $\mathrm{Tr}^{\mathcal{ID}}_{pk,sk,k}$ by its simulated counterpart $\tilde{\mathrm{Tr}}^{\mathcal{ID}}_{pk,k}$. Since the statistical distance between the distribution outputted by $\mathrm{Tr}^{\mathcal{ID}}_{pk,sk,k}$ and by $\tilde{\mathrm{Tr}}^{\mathcal{ID}}_{pk,k}$ is at most ε_{rs}, we have $|\Pr[\delta_7] - \Pr[\delta_6]| \leq q_S \varepsilon_{rs}$ (Fig. 3).

Initialize:
1: $(pk, sk) \xleftarrow{\$} \mathsf{KeyGen}(1^k)$
2: $hc \leftarrow 0$, $sc \leftarrow 0$, $\mathcal{M} \leftarrow \{\}$
3: **for** $i = 1, \cdots, q_S$ **do**
4: $\quad ctr \leftarrow 0$
5: $\quad$ **while** $\textsc{Rsp}_i = \perp$ and $ctr \leq l$ **do**
6: $\quad\quad (\textsc{Cmt}_i, \textsc{Ch}_i, \textsc{Rsp}_i) \xleftarrow{\$} \begin{cases} \mathrm{Tr}^{\mathcal{ID}}_{pk,sk,k} \\ \boxed{\tilde{\mathrm{Tr}}^{\mathcal{ID}}_{pk,k}} \end{cases}$
7: **return** pk

On Sign-query m:
1: $sc \leftarrow sc + 1$, $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$
2: **return** $\sigma = (\textsc{Cmt}_{sc}, \textsc{Rsp}_{sc})$

On Hash-query $\textsc{Cmt}, m$:
1: **if** $H(\textsc{Cmt}, m) = \perp$ **then**
2: $\quad hc \leftarrow hc + 1$
3: $\quad \textsc{Ch} \xleftarrow{\$} \{0,1\}^{c(k)}$
4: $\quad H(\textsc{Cmt}, m) \leftarrow \textsc{Ch}$
5: **return** $H(\textsc{Cmt}, m)$

Finalize(m^*, σ^*):
1: Parse σ^* as $\textsc{Cmt}^*, \textsc{Rsp}^*$
2: $\textsc{Ch}^* \leftarrow H(\textsc{Cmt}^*, m^*)$
3: $\mathrm{Dec} \leftarrow \mathsf{V}(pk, \textsc{Cmt}^*||\textsc{Ch}^*||\textsc{Rsp}^*)$
4: **return** $\mathrm{Dec} \wedge (m^* \notin \mathcal{M})$

Fig. 3. **Exp**$_6$ and $\boxed{\textbf{Exp}_7}$

The last step of the proof is to show that we can use the forger from **Exp**$_7$ to construct an impersonator $\mathcal{I}$ playing the experiment $\mathbf{Exp}^{\text{imp-pa-sim}}_{\mathcal{ID},\mathcal{I}}(k)$. The impersonator receives a public key pk from an honest verifier V, chooses an index fp uniformly at random from $[1, q_H + 1]$ and sends pk to $\mathcal{F}$. It also generates the transcripts $(\textsc{Cmt}_i, \textsc{Ch}_i, \textsc{Rsp}_i)$ for $i = 1, \cdots, q_S$ thanks to $\tilde{\mathrm{Tr}}^{\mathcal{ID}}_{pk,k}$. On the j-th hash query $(\textsc{Cmt}_j, m_j)$ of $\mathcal{F}$, it first checks if $j \neq fp$. If so, $\mathcal{I}$ works like in **Exp**$_7$ and if not, $\mathcal{I}$ returns $\textsc{Cmt}_{fp}$ as the first interaction with the verifier V. In that case, the verifier outputs a challenge $\textsc{Ch}^*$, the impersonator sets $H(\textsc{Cmt}_{fp}, m_{fp}) \leftarrow \textsc{Ch}^*$ and returns this value to $\mathcal{F}$. On the i-th signing query, $\mathcal{I}$ returns $\sigma = (\textsc{Cmt}_i, \textsc{Rsp}_i)$ as in **Exp**$_7$. Eventually, the forger $\mathcal{F}$ outputs a forgery $(m^*, \sigma^* = (\textsc{Cmt}^*, \textsc{Rsp}^*))$ and $\mathcal{I}$ outputs $\textsc{Rsp}^*$ to the verifier as the last step of the identification protocol. If $(\textsc{Cmt}^*, m^*) = (\textsc{Cmt}_{fp}, m_{fp})$, then the probability that **Exp**$_7$ outputs one is the that $\mathbf{Exp}^{\text{imp-pa-sim}}_{\mathcal{ID},\mathcal{I}}(k)$ outputs one. We get $\Pr[\delta_7] \leq (q_H + 1)\mathbf{Adv}^{\text{imp-pa-sim}}_{\mathcal{ID},\mathcal{I}}(k)$. Putting everything together and we get the expected result. □

3.3 Result from Lossy Identification Scheme

To prove the security of the underlying identification scheme we can either use a decision hard problem, or a search hard problem. By using a decision hard problem, we can replace the public key pk by a "lossy" version in the impersonation experiment like in [AFLT12]. Then if the identification is lossy (see definition in part 2.2), we can add another step to the proof of Theorem 1 and we get the following result:

Theorem 2. *Let $\mathcal{ID} = (\mathsf{KeyGen}, \mathsf{LossyKeyGen}, \mathsf{P}, \mathsf{V}, c, g_v, f)$ be a lossy identification scheme using rejection sampling whose commitment space has min-entropy $\beta(k)$, let $H : \{0,1\}^* \to \{0,1\}^{c(k)}$ be a hash function modeled as a random oracle, and let $\mathcal{DS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be the associated signature as in Definition 6. If $\mathcal{ID}$ is ε_{rs}-perfect naHVZK, has correctness error ε_c, is (t', ε_k)-key-indistinguishable, and is secure against impersonation under passive attacks with respect to lossy keys then $\mathcal{DS}$ is existentially unforgeable secure against adaptive chosen-message attack in the random oracle model such that:*

$$\boldsymbol{Adv}^{uf\text{-}cma}_{\mathcal{DS},\mathcal{F}}(k) \leq (q_H+1)\boldsymbol{Adv}^{los\text{-}imp\text{-}pa}_{\mathcal{ID},\mathcal{I}}(k) + \varepsilon_k + q_S\varepsilon_{rs} + \frac{l(q_S+q_H+1)q_S}{2^\beta} + q_S\varepsilon_c^l.$$

Furthermore, $\mathcal{F}$ runs in times $t = t' - O(q_S t_{\mathrm{Sign}})$.

Proof. The beginning of the proof is the same as for Theorem 1, we use the same sequence of experiments $\mathbf{Exp}_0, \cdots, \mathbf{Exp}_7$ plus another experiment $\mathbf{Exp}_8$.

$\mathbf{Exp}_8$. In this experiment, the challenger generates the public key thanks to the $\mathsf{LossyKeyGen}(1^k)$ instead of $\mathsf{KeyGen}(1^k)$. Distinguishing these two experiments corresponds to the key-indistinguishability property of $\mathcal{ID}$, we get $|\Pr[\delta_8] - \Pr[\delta_7]| \leq \varepsilon_k$. To conclude the proof, as in the proof of Theorem 1, we show that we can use the forger from $\mathbf{Exp}_8$ to construct an impersonator $\mathcal{I}$ playing the experiment $\mathbf{Exp}^{\text{los-imp-pa}}_{\mathcal{ID},\mathcal{I}}(k)$. □

4 Application to the BLISS Signature

In this part, we apply our result from Theorem 1 to the non-lossy identification scheme of the BLISS signature. We describe this identification scheme, its properties and we compare our result to the original BLISS proof.

4.1 Description of the BLISS Identification and Signature Schemes

The BLISS signature was introduced by Ducas et al. [DDLL13] follows directly from the work of [Lyu12], where the authors improved the rejection sampling by taking a bimodal Gaussian instead of a shifted Gaussian. The secret key sk is a short matrix $\mathbf{S} \in \mathbb{Z}_{2q}^{m \times n}$ and the public key is a matrix $\mathbf{A} \in \mathbb{Z}_{2q}^{n \times m}$ such that $\mathbf{AS} = q\mathbf{I}_n \mod 2q$. The challenge set is the set of binary vectors of length n and weight κ, $\mathcal{C} = \{\mathbf{v} : \mathbf{v} \in \{0,1\}^n, \|\mathbf{v}\|_1 \leq \kappa\}$. The hash function $H : \{0,1\}^* \to \mathcal{C}$ outputs uniform elements in the challenge set $\mathcal{C}$. The underlying identification scheme of the BLISS signature works as follows:

$$
\begin{array}{ccc}
\mathsf{P}_{sk=(\mathbf{A},\mathbf{S})} & & \mathsf{V}_{pk=\mathbf{A}} \\
\mathbf{y} \xleftarrow{\$} D_\sigma^m,\ \mathbf{u} \leftarrow \mathbf{A}\mathbf{y} \mod 2q & \xrightarrow{\mathbf{u}} & \\
b \xleftarrow{\$} \{0,1\},\ \mathbf{z} \leftarrow (-1)^b\mathbf{S}\mathbf{c} + \mathbf{y} & \xleftarrow{\mathbf{c}} & \mathbf{c} \xleftarrow{\$} \mathcal{C} \\
\textbf{Output } \mathbf{z} \text{ with probability} & \xrightarrow{\mathbf{z}} & \textbf{Output } 1 \text{ iff } \|\mathbf{z}\|_\infty < q/4, \\
1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{S}\mathbf{c}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{S}\mathbf{c}\rangle}{\sigma^2}\right)\right) & & \|\mathbf{z}\| \leq \eta\sigma\sqrt{m}, \text{ and} \\
\text{otherwise } \textbf{output } \mathbf{z} \leftarrow \perp & & \mathbf{A}\mathbf{z} + q\mathbf{c} = \mathbf{u} \mod 2q.
\end{array}
$$

4.2 Properties of the Identification Scheme

We give an high level overview of the properties achieve by the BLISS identification scheme (for more details, see [DDLL13]).

Perfect Rejection Sampling. The target output distribution of the prover responses is $f(\mathbf{z}) = D_\sigma^m$. The responses in the above identification scheme follow the family of distribution $g_{\mathbf{S}\mathbf{c}}(\mathbf{z}) = \frac{1}{2}D^m_{\mathbf{S}\mathbf{c},\sigma}(\mathbf{z}) + \frac{1}{2}D^m_{-\mathbf{S}\mathbf{c},\sigma}(\mathbf{z}) = f(\mathbf{z}) \exp\left(-\frac{\|\mathbf{S}\mathbf{c}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{S}\mathbf{c}, \mathbf{z}\rangle}{\sigma^2}\right)$.

To ensure that $M \cdot g_{\mathbf{S}\mathbf{c}}(\mathbf{z}) \geq f(\mathbf{z})$ for all z, the authors of [DDLL13] choose $M = \exp\left(\frac{1}{2\alpha^2}\right)$ where α is such that $\sigma \geq \alpha\|\mathbf{S}\mathbf{c}\|$.

naHVZK. We describe the transcript generation oracle $\mathrm{Tr}^{\mathcal{ID}}_{pk,sk,k}$, and the simulated one $\tilde{\mathrm{Tr}}^{\mathcal{ID}}_{sk,k}$. Thanks to Lemma 1 the outputs of these to algorithm are statistically closed.

$\mathrm{Tr}^{\mathcal{ID}}_{pk,sk,k}$:
$\mathbf{y} \leftarrow D_\sigma^m,\ \mathbf{u} \leftarrow \mathbf{A}\mathbf{y} \mod 2q$
$\mathbf{c} \xleftarrow{\$} \mathcal{C},\ b \xleftarrow{\$} \{0,1\},\ \mathbf{z} \leftarrow (-1)^b\mathbf{S}\mathbf{c} + \mathbf{y}$
return $(\mathbf{u}, \mathbf{c}, \mathbf{z})$ with probability $1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{S}\mathbf{c}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{S}\mathbf{c}\rangle}{\sigma^2}\right)\right)$, otherwise **return** $(\perp, \perp, \perp)$.

$\tilde{\mathrm{Tr}}^{\mathcal{ID}}_{pk,k}$:
$\mathbf{z} \leftarrow D_\sigma^m$
$\mathbf{c} \xleftarrow{\$} \mathcal{C}$
$\mathbf{u} \leftarrow \mathbf{A}\mathbf{z} + q\mathbf{c} \mod 2q$
return $(\mathbf{u}, \mathbf{c}, \mathbf{z})$ with probability $\frac{1}{M}$, otherwise **return** $(\perp, \perp, \perp)$

Correctness Error. The prover uses a rejection sampling technique to ensure that its response $\mathbf{z}$ is independent from its secret key, and by Lemma 1, we know that the prover outputs $\mathbf{z} \neq \perp$ with probability at least $1 - 1/M$. If the prover outputs a valid response, we have $\mathbf{A}\mathbf{z} - q\mathbf{c} = \mathbf{A}((-1)^b\mathbf{S}\mathbf{c} + \mathbf{y}) + q\mathbf{c} = \mathbf{u}$ and by Lemma 1, $\mathbf{z}$ is distributed according to D_σ^m and hence has norm $\|\mathbf{z}\| \leq \eta\sigma\sqrt{m}$ with high probability.

Impersonation/Soundness. We want here to have an idea of the advantage of an impersonator $\mathcal{I}$ playing the experiment $\mathbf{Exp}^{\text{imp-pa-sim}}_{\mathcal{ID},\mathcal{I}}(k)$, where the impersonator $\mathcal{I}$ has access to the real public key of the scheme $pk = \mathbf{A}$ and to the simulated transcript generation oracle $\tilde{\mathrm{Tr}}^{\mathcal{ID}}_{pk,k}$ described above. If we apply the Reset Lemma 2, the advantage of $\mathcal{I}$ corresponds to the probability acc and we get two valid transcripts on a same commitment $(\mathbf{u}, \mathbf{c}, \mathbf{z})$ and $(\mathbf{u}, \mathbf{c}', \mathbf{z}')$ with

probability frk. With these two transcripts, we get $\mathbf{Az}+q\mathbf{c} = \mathbf{Az}'+q\mathbf{c}' \mod 2q$ which gives $\mathbf{A}(\mathbf{z}-\mathbf{z}') = 0 \mod q$.

Then $\mathbf{z}-\mathbf{z}'$ is a solution of norm at most $\leq 2\eta\sigma\sqrt{m}$ of a SIS instance of parameters n, m, q, and $\beta = 2\eta\sigma\sqrt{m}$. If $\mathbf{Adv}_{\mathrm{SIS}}$ denotes the advantage against such SIS instance, we finally get $\mathbf{Adv}^{\text{imp-pa-sim}}_{\mathcal{ID},\mathcal{I}} \leq \frac{1}{|\mathcal{C}|} + \sqrt{\mathbf{Adv}_{\mathrm{SIS}}}$.

Min-Entropy of commitments. To get an idea of the min-entropy, we consider the probability that a commitment takes a particular value,

$$\Pr\left[\mathbf{Ay} = \mathbf{u};\, \mathbf{y} \leftarrow D^m_\sigma\right] \leq 2^{-n}.$$

Conclusion. Applying our result from Theorem 1 to the BLISS signature, we get

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS},\mathcal{F}} \leq (q_H+1)\left(\frac{1}{|\mathcal{C}|} + \sqrt{\mathbf{Adv}_{\mathrm{SIS}}}\right) + l(q_S+q_H+1)q_S 2^{-n} + q_S(1-1/M)^l.$$

4.3 Original BLISS Proof

The original BLISS proof is summarized in [DDLL13, Theorem 3.3] but proved through two lemmas. The first lemma [DDLL13, Lemma 3.4], states that the advantage in distinguishing the actual signing algorithm from an hybrid one constructed using the rejection sampling is at most $q_S(q_S+q_H)2^{-n}$. And the second lemma [DDLL13, Lemma 3.5] is a direct application of the General Forking Lemma of [BN06], which says that the advantage against the SIS problem with parameters n, m, q, and $\beta = 2\eta\sigma\sqrt{m}$ is at least $\mathbf{Adv}_{\mathrm{SIS}} \geq acc \cdot \left(\frac{acc}{q_S+q_H} - \frac{1}{|\mathcal{C}|}\right)$ where $acc = \mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS},\mathcal{F}} - \frac{1}{|\mathcal{C}|}$. Thanks to the General Forking Lemma, we can rewrite this equation as $acc \leq \frac{q_H+q_S}{|\mathcal{C}|} + \sqrt{(q_H+q_S)\mathbf{Adv}_{\mathrm{SIS}}}$. Putting the two lemmas together and we get

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS},\mathcal{F}} \leq \frac{q_S+q_H+1}{|\mathcal{C}|} + \sqrt{(q_H+q_S)\mathbf{Adv}_{\mathrm{SIS}}} + q_S(q_S+q_H)2^{-n}.$$

Comparison. In the original BLISS paper, the proof does the identification scheme only once during the signing algorithm instead of repeating the identification scheme until the response is non-valid. So we need to add a factor l to the term $q_S(q_S+q_H)2^{-n}$ and add the term $q_S(1-1/M)^l$ in the previous equation to better fit the Fiat-Shamir transform from Definition 6. For concrete parameters, we would have $q_H \gg q_S$, for example in the NIST submission $q_H = 2^{128}$ and $q_S = 2^{64}$, and our reduction looses a factor roughly $\sqrt{q_H}$ (we looses at least q_H in the reduction but gains roughly $\sqrt{q_H}$ by applying the Reset Lemma instead of the Forking Lemma).

Acknowledgments. Pauline Bert is funded by the Direction Générale de l'Armement (Pôle de Recherche CYBER). This work has received a French government support granted to the CominLabs excellence laboratory and managed by the National Research Agency in the "Investing for the Future" program under reference ANR-10-LABX-07-01.

References

[AABN02] Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the Fiat-Shamir transform: minimizing assumptions for security and forward-security. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 418–433. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_28

[ABB+17] Alkim, E., et al.: Revisiting TESLA in the quantum random oracle model. In: Lange, T., Takagi, T. (eds.) PQCrypto 2017. LNCS, vol. 10346, pp. 143–162. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59879-6_9

[AFLT12] Abdalla, M., Fouque, P.-A., Lyubashevsky, V., Tibouchi, M.: Tightly-secure signatures from lossy identification schemes. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 572–590. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_34

[Ajt96] Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: STOC, pp. 99–108. ACM (1996)

[BN06] Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: ACM Conference on Computer and Communications Security, pp. 390–399. ACM (2006)

[BP02] Bellare, M., Palacio, A.: GQ and Schnorr identification schemes: proofs of security against impersonation under active and concurrent attacks. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 162–177. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_11

[DDLL13] Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_3

[FS86] 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). https://doi.org/10.1007/3-540-47721-7_12

[GLP12] Güneysu, T., Lyubashevsky, V., Pöppelmann, T.: Practical lattice-based cryptography: a signature scheme for embedded systems. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 530–547. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33027-8_31

[KLS17] Kiltz, E., Lyubashevsky, V., Schaffner, C.: A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. IACR Cryptology ePrint Archive 2017, p. 916 (2017)

[KW03] Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: ACM Conference on Computer and Communications Security, pp. 155–164. ACM (2003)

[Lyu08] Lyubashevsky, V.: Lattice-based identification schemes secure under active attacks. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 162–179. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78440-1_10

[Lyu12] Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_43

[PS00] Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptology **13**(3), 361–396 (2000)

Universal Witness Signatures

Chen Qian[1], Mehdi Tibouchi[2(✉)], and Rémi Géraud[3]

[1] Univ Rennes, Rennes, France
chen.qian@irisa.fr
[2] NTT Secure Platform Laboratories, Tokyo, Japan
tibouchi.mehdi@lab.ntt.co.jp
[3] École normale supérieure, Paris, France
remi.geraud@ens.fr

Abstract. A lot of research has been devoted to the problem of defining and constructing signature schemes with various delegation properties. They mostly fit into two families. On the one hand, there are signature schemes that allow the delegation of *signing rights*, such as (hierarchical) identity-based signatures, attribute-based signatures, functional signatures, etc. On the other hand, there are *malleable* signature schemes, which make it possible to derive, from a signature on some message, new signatures on related messages without owning the secret key. This includes redactable signatures, set-homomorphic signatures, signatures on formulas of propositional logic, etc.

In this paper, we set out to unify those various delegatable signatures in a new primitive called *universal witness signatures* (UWS), which subsumes previous schemes into a simple and easy to use definition, and captures in some sense the most general notion of (unary) delegation. We also give several constructions based on a range of cryptographic assumptions (from one-way functions alone to SNARKs and obfuscation) and achieving various levels of security, privacy and succinctness.

Keywords: Digital signatures · Delegation · Malleable signatures
Functional signatures · Identity-based cryptography · Obfuscation
SNARKs

1 Introduction

Many signature schemes in the literature have delegatability properties either for keys or for messages.

The first family includes notions like (hierarchical) identity-based [14,18,21,28] and attribute-based [22,23,25] signatures, in which a master key authority grants signing rights to users, who may in turn be able to delegate those rights to lower-level signers. It also includes a slightly different class of schemes where the owner of the master secret key can sign all messages in the message space, and can delegate signing rights on restricted families of messages (again, possibly with

A. Inomata and K. Yasuda (Eds.): IWSEC 2018, LNCS 11049, pp. 313–329, 2018.
https://doi.org/10.1007/978-3-319-97916-8_20

recursive delegation): functional signatures [4,11] and policy-based signatures [6] are examples of such schemes.

The second family of schemes is that of malleable signature schemes, as defined e.g. by Ahn et al. [1], Attrapadung et al. [3] and Chase et al. [12]: in those schemes, it is possible, given a signature on some message, to publicly derive signatures on certainly related messages. Specific examples include content extraction signatures [29], redactable and sanitizable signatures [2,19], some variants of set-homomorphic and network coding signatures [10,19] (where users sign sets, resp. vector spaces, and those signatures can be delegated to subsets or subspaces) and more. A different example and one of the original motivations of this work is Naccache's notion of signatures on formulas of propositional logic [24], which makes it possible to derive a signature on a propositional formula Q from a signature on P whenever $P \Rightarrow Q$.

Main Idea of This Work. The goal of this paper is to unify all of the schemes above into a single very general and versatile primitive, which we call *universal witness signatures* (UWS), and to propose concrete instantiations of that primitive achieving good security and privacy properties.

Our first observation is that delegation of keys and delegation of messages are really two sides of the same coin, which can be unified by regarding a *signature* on a message m by an identity A as really the same object as a *key* associated with the sub-identity (A, m) of A. The operation of signing messages simply becomes a special case of key delegation. Then, we can obtain in some sense the most general notion of signature scheme with (unary) delegation by saying that the set of identities is endowed with an essentially arbitrary pre-order relation $\leq$, and that given a key SK_A on an identity A, we are able to derive another key SK_B on any identity B such that $B \leq A$. Unforgeability is then defined in the obvious way: roughly speaking, after obtaining keys SK_{A_i} on various identities A_i (possibly derived from higher-level identities), an adversary is unable to construct a valid key SK_B for an identity B that doesn't satisfy $B \leq A_i$ for any i.

The type of delegation functionality achieved by such a scheme is simply determined by the pre-order relation $\leq$. For example, regular (non-delegatable) signatures are obtained by choosing a set of identities equal to the message space together with a special identity $*$, such that $m \leq *$ for any message m, and no other relationship exists. Then, SK_* is the secret key in the traditional sense, and it can be used to derive keys SK_m playing the role of signatures. If non-trivial relationships exist between the messages m themselves, we get a malleable signature scheme instead. For instance, we get redactable signatures if messages are ordered in such a way that $m' \leq m$ if and only if m' is obtained from m by replacing some of the text in m by blanks. And we obtain identity-based signatures with a larger set of identities, still containing a special identity $*$ associated with the master key authority, but also identities id_i associated with the various users of the system, and identities (id_i, m) for each pair of a user and a message. The order relation is then given by $\mathsf{id}_i \leq *$ for all i, and $(\mathsf{id}_i, m) \leq \mathsf{id}_i$

for all i and all messages. If additional nontrivial relationships exist between the id_i's, we essentially get hierarchical identity-based signatures.

We would like to support a really general class of pre-order relations $\leq$, to support for example the signatures on propositional formulas mentioned earlier, which are malleable signatures on messages (formulas) ordered by logical implication. However, the delegation algorithm that lets us publicly derive SK_B from SK_A when $B \leq A$ should certainly be able to efficiently test whether the relation $B \leq A$ actually holds. This is not possible directly for a relation like logical implication. For general NP relations, however, it does become possible if the delegation algorithm also receives as input a witness w of $B \leq A$ (for logical implication, for example, it would be a proof that $A \Rightarrow B$).

Our Contributions. Along the lines sketched above, our first contribution is to define the notion of universal witness signature (UWS) scheme with respect to an arbitrary NP pre-order relation $\leq$ with a greatest element $*$. Such a scheme consists of only two algorithms:

$\mathsf{Setup}(1^\lambda)$: returns a key SK_* on the special identity $*$, as well as some public parameters PP;

$\mathsf{Delegate}(\mathsf{PP}, \mathsf{SK}_A, A, B, w)$: checks that SK_A is a valid key for the identity A and that w is a valid witness of $B \leq A$. If so, returns a fresh key SK_B for the identity B. Otherwise, returns $\perp$.

We do not actually need a separate algorithm for signature verification: to check whether SK_A is a valid key on A, we can simply try to delegate A to itself (since $\leq$ is a pre-order, there is a trivial witness $w_{A \leq A}$ for $A \leq A$), and test whether the $\mathsf{Delegate}$ algorithm returns something or just $\perp$.

Security for a UWS scheme is defined as the unforgeability notion described in the previous paragraph. As usual, we distinguish between selective security (in which the adversary has to choose in advance the identity on which it will try to forge) and adaptive security. We also identify two other desirable properties of a UWS scheme: the privacy notion of context-hiding UWS, which says that the delegation path used to obtain a given key is computationally hidden, and the notion of succinctness, which says that the size of a key SK_A is bounded only in terms of the size of A and the security parameter, independently of the delegation path.

We show that universal witness signatures are sufficient to obtain many earlier schemes appearing the literature, including HIBS, redactable signatures, functional signatures and Naccache's propositional signatures (for which our concrete constructions provide the first complete instantiations, to the best of our knowledge).

And finally, we give several constructions of UWS based on a range of assumptions, and achieving various subsets of our desirable properties. First, we show that one-way functions alone are enough to obtain adaptively secure UWS for arbitrary NP pre-order relations. The approach is similar to the one-way function-based construction of functional signatures [11]. As in the work of

Boyle et al., however, the resulting scheme is neither context-hiding nor succinct. Then, we prove that virtual black-box obfuscation [5] (for a well-defined program depending on the pre-order relation under consideration) provides a very simple construction of secure, succinct, context-hiding UWS. This construction ticks all of our boxes, but it is of course based on a very strong assumption: in fact, Barak et al. showed that VBB is unachievable for general circuits. There *could* exist a virtual black-box obfuscator for our specific program of interest (and in fact, candidate constructions of indistinguishability obfuscation conjecturally satisfy that property), but this is rather speculative. We therefore try to achieve similarly strong properties based on somewhat more reasonable assumptions. We give two such constructions: one based on SNARKs [9,17,20] (actually, proof-carrying data [7,8,13]), which is secure and succinct but achieves a somewhat weaker form of privacy than the context-hiding property; and another based on indistinguishability obfuscation [5,15,16,27], which is succinct and context-hiding but which we only prove selectively secure. Both of those constructions suffer from a limitation on delegation depth: it can be an arbitrary polynomial in the security parameter but fixed at Setup time. Moreover, our iO-based construction only applies to order relations rather than general pre-orders.

2 Universal Witness Signatures

Our notion of universal witness signature generalizes many existing signature schemes with delegation or malleability, such as homomorphic signatures [19], functional signatures [11] and hierarchical identity-based signatures [14]. It supports delegation hierarchies defined by arbitrary NP pre-order relations, which, to the best of our knowledge, makes it more general than all previously proposed such schemes. And yet, its syntax is quite simple, consisting of only two algorithms.

Formal Definition. Now we give a formal definition of our scheme. Note that the terms *identity* and *message* are synonymous in our setting, and can be used interchangeably. Similarly, the *signing key* for an identity can equivalently be seen as a *signature* on the corresponding message.

Definition 1 (Universal Witness Signatures). *A* universal witness signature scheme *(*UWS *for short) for an* NP *pre-order relation* $\leq$ *with a greatest element* $*$ *consists of two probabilistic polynomial-time algorithms:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{PP}, \mathsf{SK}_*)$*: This algorithm takes as input a security parameter* λ*, produces a master signing key* SK_**, keeps it secret and outputs public parameters* PP*.*
- $\mathsf{Delegate}(\mathsf{PP}, \mathsf{SK}_A, A, B, w) \to \mathsf{SK}_B$*: This function takes as input the public parameters, a signing key for identity* A*, an identity* B*, and a witness* w*. If* w *is a valid witness of the* NP *statement* $B \leq A$ *and* SK_A *is a valid signing key for identity* A*, then the algorithm outputs a signing key for identity* B*, otherwise it outputs* $\perp$*.*

We note that there exists a trivial witness for $A \leq A$. For simplicity's sake, we also define a third algorithm Verify*, which is merely a specialization of* Delegate*:*

– $\mathsf{Verify}(\mathsf{PP}, \mathsf{SK}_A, A) \rightarrow \{\mathsf{True}, \mathsf{False}\}$*: This algorithm takes a public parameter, an identity A and a signing key SK_A. We have* $\mathsf{Verify}(\mathsf{PP}, \mathsf{SK}_A, A) = \mathsf{False}$ *if and and only if* $\mathsf{Delegate}(\mathsf{PP}, \mathsf{SK}_A, A, A, w_{A \leq A})$ *returns $\perp$, where $w_{A \leq A}$ is the trivial witness for the reflexive property $A \leq A$.*

Additional Properties of UWS**.** We now propose three desirable properties for a UWS scheme. Intuitively, we will say that a UWS scheme is *secure* (i.e. unforgeable) if a polynomially-bounded adversary allowed to make arbitrary delegation queries cannot come up with a valid signing key on an identity A^* that isn't reachable by delegation from any of the signing keys she obtained.

We will say that the scheme is *context-hiding* if a signing key does not reveal the delegation path used to derive it. For example, a signing key SK_A on A obtained from a long delegation path is indistinguishable from a signing key for the same identity delegated directly from the master key SK_*.

Finally, we will say that the scheme is *succinct* when the size of a signing key is bounded depending only on the size of the associated identity (and not on the length of the delegation path used to derive it).

These properties can be captured formally as follows.

Definition 2 (Correctness). Correctness *of an* UWS *scheme states that for all SK_A, if*

$$\mathsf{UWS.Verify}(\mathsf{PP}, \mathsf{SK}_A, A) \text{ outputs } \mathsf{True} \text{ and } w_{B \leq A} \text{ is a witness of } B \leq A,$$

then the signing key $\mathsf{SK}_B \leftarrow \mathsf{UWS.Delegate}(\mathsf{PP}, A, B, w_{B \leq A})$ is not equal to $\perp$, and $\mathsf{UWS.Verify}(\mathsf{PP}, \mathsf{SK}_B, B)$ outputs True*.*

Definition 3 (Selective security of UWS schemes). *The* selective security *of an* UWS *scheme is captured by the advantage of an adversary $\mathcal{A}$ in the following security game against a challenger $\mathcal{C}$:*

1. *$\mathcal{A}$ chooses a target identity A^*.*
2. *$\mathcal{C}$ runs the algorithm $\mathsf{Setup}(1^\lambda)$ to get SK_* and PP. Then it keeps the master signing key SK_* secret and sends the public parameter PP to $\mathcal{A}$, together with a tag t_{SK_*} referring to SK_*.*
3. *$\mathcal{C}$ initializes an associative list $\mathcal{H}$ in which the challenger maintains a set of tuples (t, A, SK) where t is a tag for the signing key generated by the challenger, A is an identity, and SK is a signing key for the identity A. The associative initially consists of just the tuple $(t_{\mathsf{SK}_*}, *, \mathsf{SK}_*)$.*
4. *$\mathcal{A}$ can submit two types of queries:*
   Delegate queries: *Assume that $\mathcal{C}$ generated the signing key SK_A on an identity A, and that $\mathcal{A}$ received the corresponding tag t_{SK_A}. For any identity B for which $\mathcal{A}$ knows a witness $w_{B \leq A}$ of $B \leq A$, $\mathcal{A}$ can instruct $\mathcal{C}$ to execute $\mathsf{Delegate}(\mathsf{PP}, \mathsf{SK}_A, A, B, w_{B \leq A})$. The challenger will then generate a new unique tag t_{SK_B} on the resulting signing key SK_B, and add the tuple $(t_{\mathsf{SK}_B}, B, \mathsf{SK}_B)$ to its associative list $\mathcal{H}$, sending back the tag t_{SK_B} to $\mathcal{A}$.*

Reveal queries: *Using the corresponding tag, $\mathcal{A}$ can ask $\mathcal{C}$ to reveal any previously generated signing key SK_A provided that the associated identity A does not satisfy $A^* \leq A$.*

5. *After polynomially many queries of the type above, $\mathcal{A}$ outputs a candidate forgery SK_{A^*}, and wins if and only if*

$$\mathsf{Verify}(\mathsf{PP}, \mathsf{SK}_{A^*}, A^*) = \mathsf{True}.$$

A UWS *scheme is* selectively secure *if and only if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the previous game is negligible.*

We also consider the *adaptive security* of UWS schemes, which is similarly defined, with the notable exception that the adversary does not announce in advance the identity A^* on which she will forge, but can choose it adaptively instead (with the condition that it does not satisfy $A^* \leq A$ for any of the identities A associated with revealed signatures).

Definition 4 (Context-hiding). *A* UWS *scheme is* context-hiding *if, for every tuple $(\mathsf{SK}_1, \mathsf{SK}_2, A_1, A_2, w_1, w_2)$ and every identity B such that*

$$\mathsf{Delegate}(\mathsf{PP}, \mathsf{SK}_1, A_1, B, w_1) \to \mathsf{SK}_1' \neq \bot$$
$$\textit{and}$$
$$\mathsf{Delegate}(\mathsf{PP}, \mathsf{SK}_2, A_2, B, w_2) \to \mathsf{SK}_2' \neq \bot,$$

the distributions of $(\mathsf{PP}, B, \mathsf{SK}_1')$ and $(\mathsf{PP}, B, \mathsf{SK}_2')$ are statistically close.

Definition 5 (Succinctness). *An* UWS *scheme is* succinct *if there exists a polynomial p such that the size of any signature SK_A is bounded by $p(\lambda, |A|)$.*

3 Applications: From UWS to Other Primitives

Our universal witness signature scheme can be considered as a generalization of many existing malleable signature schemes. To showcase this, we use our UWS scheme to instantiate several well-known signature schemes, and some more original ones, namely: functional signatures and propositional signatures. A construction of hierarchical identity-based signatures and redactable signatures is also given in the full version of this paper [26]. To the best of our knowledge, this is the first time that a construction for propositional signatures appears in the literature.

Functional Signatures. Functional signatures, introduced by Boyle et al. [11], are a particularly wide-ranging generalization of identity-based signatures in which the key authority can generate signing keys sk_f associated to functions f, such that the owner of sk_f can sign exactly those messages that are in the image of f. Moreover, to sign m in the image of f, the owner of sk_f needs a witness to this fact, namely a preimage of m under f. In this section, we show how we can easily obtain functional signatures based on universal witness signatures.

Definition 6 (Functional Signature). *The* functional signature *for a message space $\mathcal{M}$ and a function family $\mathcal{F} = \{f : \mathcal{D}_f \to \mathcal{M}\}$ is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *which is specified as follows:*

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{msk}, \mathsf{mvk})$*: the setup algorithm takes a security parameter λ and it returns a master signing key* msk *and a master verification key* mvk*. Then it keeps the master signing key secret* msk *and publishes the master verification key* mvk*.*
- $\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$*: the key generation algorithm takes a master signing key* msk *and a function f to specify which one will be allowed to sign the messages, then it outputs a corresponding signing key.*
- $\mathsf{Sign}(f, \mathsf{sk}_f, m) \to (f(m), \sigma_{f(m)})$*: the signing algorithm takes a function f and the corresponding signing key sk_f as input and produces $f(m)$ and the signature $\sigma_{f(m)}$ of $f(m)$*
- $\mathsf{Verify}(\mathsf{mvk}, \sigma_m, m) \to \{\mathsf{True}, \mathsf{False}\}$*: the verification algorithm takes a signature-message pair (m, σ_m) and the master verification key* mvk*. The algorithm outputs* True *if σ_m is a valid signature of m, otherwise outputs* False*.*

Functional Signatures From UWS. Consider the order $\mathcal{O}_\mathcal{F}$ corresponding to the function family $\mathcal{F}$:

- Let $\mathcal{M}$ be the message space. The order $\mathcal{O}_\mathcal{F}$ is an order on the set $\{*\} \cup \mathcal{F} \cup \mathcal{M}$
- $*$ is the greatest identity, bigger than all other messages.
- $m \leq f$ when $\exists m' \in \mathcal{M} \wedge m = f(m')$
- There does not exist any other non-trivial order

We note $\leq$ be the previously defined order. Then consider the UWS scheme corresponding to this order. The construction of the functional signature is described in figure Fig. 1.

Security of the Functional Signature Scheme. A functional signature scheme is typically expected to verify the following properties:

- *Correctness:* This property expresses the fact that a properly generated signature is verified to be correct. Formally: $\forall f \in \mathcal{F}, \forall m \in \mathcal{D}_f$, $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f), (f(m), \sigma_{f(m)}) \leftarrow \mathsf{Sign}(f, \mathsf{sk}_f, m)$,

$$\mathsf{Verify}(\mathsf{mvk}, \sigma_{f(m)}, f(m)) = \mathsf{True}.$$

- *Unforgeability:* The unforgeability of the functional signature scheme is defined by the the following security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:
  - $\mathcal{C}$ generates a pair of keys $(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{Setup}(1^\lambda)$, then publishes the master verification key mvk byt keeps the master signing key msk secret.
  - $\mathcal{C}$ constructs an initially empty associative list $\mathcal{H}$ indexed by $f \in \mathcal{F}$, a key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$ and a signing oracle $\mathcal{O}_{\mathsf{Sign}}$ as follows:

$*$ $\mathcal{O}_{\mathsf{KeyGen}}(f)$: If there exists already a value associated to f in the associative list $\mathcal{H}$, then outputs $\mathcal{H}(f)$ directly. Otherwise the oracle uses the KeyGen algorithm to generate the signing key $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ associated to f, then adds the function-signing key pair (f, sk_f) to the associative list.
$*$ $\mathcal{O}_{\mathsf{Sign}}(f, m)$: If there exists a value associated to f in the associative list $\mathcal{H}$, then uses the signing algorithm to generate a signature $\sigma_{f(m)}$of $f(m)$

- $\mathcal{A}$ can query the two oracles $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{KeyGen}}$. $\mathcal{A}$ can also make requests of corresponding value in the associative list $\mathcal{H}$. $\mathcal{A}$ win against the security game if it can produce a message-signature pair (m, σ) such that:
  $*$ $\mathsf{Verify}(\mathsf{mvk}, m, \sigma) = 1$
  $*$ There does not exist m' and f such that $m = f(m')$ and f was sent as a query to the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$.
  $*$ There does not exist a function-message pair (f, m') was a query to the signing oracle $\mathcal{O}_{\mathsf{Sign}}$ and $m = f(m')$

Let $\mathcal{A}$ be an adversary against the functional signature constructed using UWS scheme with non-negligible advantage. Then by the definition it can produce a message-signature pair (m, σ) such that:

1. $\mathsf{Verify}(\mathsf{mvk}, m, \sigma) = \mathsf{True}$
2. There does not exist m' and f such that $m = f(m')$ and f was sent as a query to the key generation oracle $\mathcal{O}_{\mathsf{KeyGen}}$.
3. The message m was not sent as a query to the signing oracle $\mathcal{O}_{\mathsf{Sign}}$.

Condition 1 implies that $\mathsf{Verify}(\mathsf{PP}, \sigma, m) = \mathsf{True}$. Condition 2 implies that for all (f, m') which verifies that $f(m') = m$, sk_f has never been revealed. Then the third condition implies that σ_m has not been revealed. As in the specific order corresponding to the functional signature, the only elements bigger than m are m itself, $*$, and $\{f \mid \exists m' \in \mathcal{M}.f(m') = m\}$. With the conditions 2 and 3, the signatures of these identities have never been revealed, but $\mathcal{A}$ can produce a valid signature for m which break the existential unforgeability of the underlying UWS scheme.

$\underline{\mathsf{Verify}(\mathsf{mvk}, \sigma_m, m)}$:
return $\mathsf{UWS.Verify}(\mathsf{PP}, \mathsf{SK}_{f(m)}, f(m))$

$\underline{\mathsf{Setup}(1^\lambda)}$:
$(\mathsf{PP}, \mathsf{SK}_*) \leftarrow \mathsf{UWS.Setup}(1^\lambda)$
return $(\mathsf{SK}_*, \mathsf{PP})$

$\underline{\mathsf{KeyGen}(\mathsf{msk}, f)}$:
return $\mathsf{UWS.Delegate}(\mathsf{PP} = \mathsf{msk}, \mathsf{SK}_*, *, f, \text{“}f \leq *\text{”})$

$\underline{\mathsf{Sign}(f, \mathsf{sk}_f, m)}$:
$\sigma_{f(m)} = \mathsf{UWS.Delegate}(\mathsf{PP}, \mathsf{sk}_f, f, f(m), m, \text{“}f(m) \leq f\text{”})$
return $(f(m), \sigma_{f(m)})$

Fig. 1. Functional signatures from UWS.

Propositional Signatures. In an invited talk at CRYPTO and CHES 2010, Naccache [24] introduced the new notion of propositional signatures, for which he suggested a number of real-world applications such as contract-signing. Propositional signatures are signatures on formulas of propositional calculus, which are homomorphic with respect to logical implication. In other words, given a signature on a propositional formula P, one should be able to publicly derive a signature on any Q such that $P \Rightarrow Q$. Since the satisfiablity of propositional formulas cannot be decided efficiently without auxiliary information, the derivation algorithm should also take as input a witness of $P \Rightarrow Q$, i.e. a proof of Q assuming P.

To the best of our knowledge, no construction of propositional signatures has been proposed so far. However, it is easy to see that they are, again, easily obtained from UWS.

Security of Propositional Signatures. Formally, a propositional signature scheme is a triple $(\mathcal{G}, \mathcal{D}, \mathcal{V})$ of efficient algorithms for key generation: $\mathcal{G}(1^\lambda) \rightarrow (\mathsf{mvk}, \sigma_{\mathsf{False}})$, signature derivation: $\mathcal{D}(\mathsf{mvk}, \sigma_P, P, Q, \pi) \rightarrow \sigma_Q$, and verification: $\mathcal{V}(\mathsf{mvk}, \sigma_P, P) \rightarrow \mathsf{True}/\mathsf{False}$. The signature σ_{False} on the false proposition plays the role of master secret key, due to *ex falso quodlibet*. Correctness states that if σ_P is a valid signature on proposition P (in the sense that $\mathcal{V}(\mathsf{mvk}, \sigma_P, P)$ evaluates to True) and π is a valid proof of $P \Rightarrow Q$, then $\mathcal{D}(\mathsf{mvk}, \sigma_P, P, Q, \pi)$ a valid signature σ_Q on Q. Unforgeability says that after obtaining signatures on propositions P_i of his choice, an efficient adversary cannot produce a valid signature on a position Q such that none of the P_i's implies Q.

Propositional Signatures from UWS. Clearly, the UWS scheme associated with the corresponding set of propositional formulas endowed with the NP preorder relation given by logical implication (where witnesses are proofs) exactly gives a propositional signature scheme.

In fact, our definition of security captures for UWS captures a slightly stronger security model, where unforgeability still holds when the adversary can make unrestricted delegation queries on messages he cannot see, so as to control delegation paths.

4 Construction of UWS

We now explain how to realise UWS. The first construction requires only the existence of one-way function, but is neither context-hiding nor succinct. These properties can be obtained at the cost of introducing new assumptions such as the existence of proof-carrying data (for succinctness) or obfuscators (for context-hiding).

4.1 Construction from One-Way Functions

Firstly, we propose a construction of the UWS scheme based only on the existence of one-way functions. The existence of one-way function is a very weak and basic

assumption of cryptography. But this very basic construction does not verify many other properties than adaptive security like succinctness or context-hiding.

In this construction, we use a "certificate of computation" approach: Every signature is provided with a certificate of the delegation path. And an identity A can provide a certificate of the identity B if and only if $B \leq A$. For example if for the generation of the signature of the identity A, we have passed the identities $*, id_1, id_2, \ldots, id_n, A$. Each identity produced a signing-verification key pair $(\mathsf{sk}_{id_i}, \mathsf{vk}_{id_i})$ using the key generation of a signature scheme, and $\sigma_{\mathsf{vk}_{id_i}}$ is a signature of $(id_i, w_{id_i \leq id_{i-1}}, \mathsf{vk}_{id_i})$[1] produced using the signing key sk_{i-1} The signature of the identity A is represented by

$$\mathsf{SK}_A = \left(\mathsf{sk}_A, \left[\left(\mathsf{vk}_A, A, w_{A \leq n}, \sigma_{\mathsf{vk}_A}\right), \left(\mathsf{vk}_{id_n}, id_n, w_{id_n \leq id_{n-1}}, \sigma_{\mathsf{vk}_{id_n}}\right), \ldots, \right.\right.$$
$$\left.\left. \ldots, \left(\mathsf{vk}_*, *, w_{*\leq *}, \sigma_{\mathsf{vk}_*}\right) \right]\right)$$

For simplicity we will note "w is a valid witness of $x \leq y$" by "$x \leq_w y$" in the following sections of this paper and we propose the following construction of our UWS scheme using a classical existential unforgeable signature scheme $\mathsf{Sig} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$.

$\mathsf{Setup}(1^\lambda)$:

- The setup algorithm first takes two pairs of keys $(\mathsf{msk}, \mathsf{mvk}), (\mathsf{vk}_*, \mathsf{sk}_*)$ from the signature's parameters generation algorithm.
- Then it generates a signature σ_{vk_*} of $(*, w_{*\leq *}, \mathsf{vk}_*)$ using the signing key msk, this can be considered as a certificate of the verification key vk_* delivered by the master authority.
- The certificate c_* of the identity $*$ is $[(\mathsf{vk}_*, *, w_{*\leq *}, \sigma_{\mathsf{vk}_*})]$. The signing key (signature) SK_* of the identity $*$ in our UWS scheme is (sk_*, c_*) and the public parameter PP will be mvk.

$\mathsf{Delegate}(\mathsf{mvk}, \mathsf{SK}_A, A, B, w)$:

- The signing key SK_A has the form (sk_A, c_A), where the certificate c_A is a list

$$[(\mathsf{vk}_A, A, w_A, \sigma_A), \ldots, (\mathsf{vk}_*, *, w_{*\leq *}, \sigma_{\mathsf{vk}_*})].$$

- The delegation algorithm first verifies that the certificate c_A is valid, by checking that each σ_j is a valid signature on $(id_j, w_j, \mathsf{vk}_j)$ with respect to the verification key vk_{j-1}. It also checks that the witnesses are valid: $B \leq_w A \leq_{w_A} id_n \ldots$ and that vk_A is a correct public key for sk_A by signing a random message and verifying it.
- If all these verification steps succeed, a fresh key pair $(\mathsf{sk}_B, \mathsf{vk}_B)$ for Sig is generated, together with a signature σ_B on $(B, w_{B \leq A}, \mathsf{vk}_B)$ using the singing key sk_A The algorithm then computes an extended certificate c_B by prepending $(\mathsf{vk}_B, B, w_B, \sigma_B)$ to c_A, and returns the signature SK_B as (sk_B, c_B).

This construction is summarized in Fig. 2.

[1] This is represented as a message $id_i || w_{id_i \leq id_{i-1}} || \mathsf{vk}_{id_i}$.

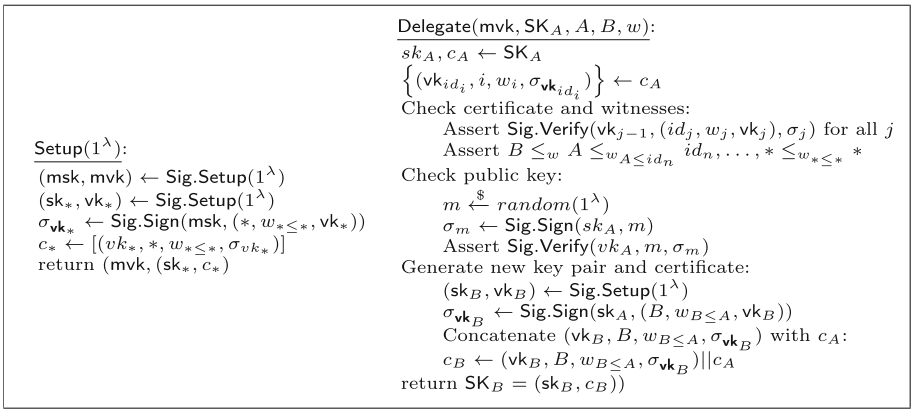


Fig. 2. Construction of UWS from one way functions.

Theorem 1. *The construction of* UWS *based on the one-way function is correct.*

Proof. If we have $\mathsf{Verify}(\mathsf{mvk}, \mathsf{SK}_A, A) = \mathsf{True}$ and $B \leq_w A$, then $\mathsf{SK}_B = (\mathsf{sk}_B, c_B)$ with $c_B = [(\mathsf{vk}_B, B, w, \sigma_B), c_A]$, and by construction, we have that $(\mathsf{vk}_B, \mathsf{sk}_B)$ is a valid signature key pair, and σ_B is a valid signature of $(B, w_{B\leq A}, \mathsf{vk}_B)$. By the verification above and hypothesis of SK_A is valid, c_B is also a valid certificat and $\mathsf{Verify}(\mathsf{mvk}, \mathsf{SK}_B, B)$ outputs True. □

The proof that this construction is adaptively secure is given in the full version of this paper [26].

4.2 Succinct Construction from Proof-Carrying Data

The previous construction based on one-way functions is clearly neither context-hiding nor succinct. In this section, we give a construction based on SNARKs, which is adaptively secure and succinct; we also provide arguments suggesting that it should satisfy some form of context-hiding property at least when the SNARK is zero-knowledge. As a downside, we are limited to a polynomial number of delegation steps: the maximum delegation depth must be fixed at setup time, as a polynomial in the security parameter. Our construction use the following theorems proposed by Bitansky et al. [8].

Theorem 2 (SNARK Recursive Composition Theorem). *There exists an efficient transformation* RecComp *algorithm such that, for every publicly-verifiable SNARK* $(\mathcal{G}_{SNARK}, \mathcal{P}_{SNARK}, \mathcal{V}_{SNARK})$*, the 3-tuple algorithms*

$$(\mathcal{G}, \mathcal{P}, \mathcal{V}) = \mathsf{RecComp}(\mathcal{G}_{SNARK}, \mathcal{P}_{SNARK}, \mathcal{V}_{SNARK})$$

is a publicly-verifiable PCD system for every constant-depth compliance predicate.

Theorem 3 (PCD Depth-Reduction Theorem). *Let $\mathcal{H} = \{\mathcal{H}_K\}_{k\in\mathcal{N}}$ be a collision-resistant hash-function family. There exists an efficient transformation* $\mathrm{DepthRed}_{\mathcal{H}}$ *with the following properties:*

- *Correctness: If $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a PCD system for constant-depth compliance predicates, then $(\mathcal{G}', \mathcal{P}', \mathcal{V}') = \mathrm{DepthRed}_{\mathcal{H}}(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a path PCD for polynomial-depth compliance predicates.*
- *Verifiability Properties: If $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is publicly verifiable then so is $(\mathcal{G}', \mathcal{P}', \mathcal{V}')$*
- *Efficiency: There exists a polynomial p such that the (time and space) efficiency of $(\mathcal{G}', \mathcal{P}', \mathcal{V}')$ is the same as that of $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ up to the multiplicative factor $p(k)$*

For the construction, we follow the same line of thinking as in Sect. 4.1. We consider an existential unforgeable signature scheme $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, with master signing key msk and master verification key mvk. Let us first define the distributed computation transcript $T = (G, \mathsf{linp}, data)$ and the corresponding $\mathcal{C}$-compliance $\mathcal{C}(z_{\text{out}}, \mathsf{linp}, z_{\text{in}})$ as follows:

- $G = (V, E)$: The graph of the distributed computation transcript, with V labeled by the identity and $(A, B) \in E$ labeled by the tuple $(\mathsf{VK}_B, \mathsf{SK}_B, w_{B\leq A}, \sigma_{\mathsf{VK}_B})$.
- linp: The local input of the vertices will be the identity corresponding to this vertex.
- z_{out}: The data of the edges are the labels of the edges, specifically $z_{\text{out}} = (\mathsf{VK}_B, \mathsf{SK}_B, w_{B\leq A}, \sigma_{\mathsf{VK}_B})$.
- $\mathcal{C}(z_{\text{out}}, \mathsf{linp}, z_{\text{in}})$: parse z_{out} as $(\mathsf{VK}_B, \mathsf{SK}_B, w_{B\leq A}, \sigma_{\mathsf{VK}_B})$, and suppose $z_{\text{in}} = (\mathsf{VK}_A, \mathsf{SK}_A, w_{A\leq C}, \sigma_{\mathsf{VK}_A})$ with C the predecessor of A. The algorithm $\mathcal{C}(z_{\text{out}}, \mathsf{linp}, z_{\text{in}})$ outputs True if
  - $\mathsf{Sig.Verify}(\mathsf{VK}_A, (\mathsf{linp}(B), w_{B\leq A}, \mathsf{VK}_B), \sigma_{\mathsf{VK}_A}) = \mathsf{True}$
  - $B \leq_{w_{B\leq A}} A$
  - For a random message m, we have $\mathsf{Sig.Verify}(\mathsf{VK}_B, m, \mathsf{Sig.Sign}(\mathsf{SK}_B, m)) = \mathsf{True}$.

Let us consider the PCD scheme $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ corresponding to the distributed computation transcript described as above. Then we have all the building blocks for our universal witness signature.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{PP}, \mathsf{SK}_*)$: Let $(\mathsf{mvk}, \mathsf{msk}) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$, then use the generation algorithm of the proof-carrying data to get $\mathsf{PP} = \mathsf{crs} \leftarrow \mathcal{G}(1^\lambda)$. Let $z_* = (\mathsf{mvk}, \mathsf{msk}, \text{“}* \leq *\text{”}, \sigma_{\mathsf{VK}_*})$, we compute $\pi_* \leftarrow \mathcal{P}(\mathsf{crs}, \perp, \perp, z_*, *)$ and $\sigma_{\mathsf{VK}_*} \leftarrow \mathsf{Sig.Sign}(\mathsf{msk}, (*, \text{“}* \leq *\text{”}, \mathsf{mvk}))$. Then output $\mathsf{SK}_* = (z_*, \pi_*)$ and $\mathsf{VK}_* = \mathsf{mvk}$.
- $\mathsf{Delegate}(\mathsf{PP}, \mathsf{SK}_A, A, B, w_{B\leq A}) \to \mathsf{SK}_B$:
  - Parse SK_A as (z_A, π_A) with $z_A = (\mathsf{VK}_A, \mathsf{SK}_A, w_{A\leq C}, \sigma_{\mathsf{VK}_A})$.
  - We first check whether π_A is valid proof of the fact that z_A is consistent with the $\mathcal{C}$-compliance transcript. If the test fails then the algorithm ouputs $\perp$

- If the check succeeds, use the underlying signature scheme's generation algorithm to get $(\mathsf{SK}_B, \mathsf{VK}_B) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$, and use the sign algorithm to get $\sigma_{\mathsf{VK}_B} = \mathsf{Sig.Sign}(\mathsf{SK}_A, (B, w_{A \leq B}, \mathsf{VK}_B))$. Let $z_B = (\mathsf{VK}_B, \mathsf{SK}_B, w_{B \leq A}, \sigma_{\mathsf{VK}_B})$.
- Finally use the proof algorithm $\mathcal{P}$ to generate a proof of z_B: $\pi_B \leftarrow \mathcal{V}(\mathsf{crs}, z_A, \pi_A, z_B, B)$ which is a proof of the fact that z_B is consistent with the $\mathcal{C}$-compliance transcript. The algorithm outputs $\mathsf{SK}_B = (z_B, \pi_B)$.

This construction is summarized in Fig. 3.

Theorem 4. *The construction of* UWS *scheme based on the Proof-Carrying Data is correct.*

Proof. If SK_B is produced by the Delegate algorithm on SK_A, and SK_A is a valid signature of A, by the correctness of the PCD scheme, we have π_B is a valid proof of the fact that z_B is consistent with the $\mathcal{C}$-compliance transcript. This assures that the new signature SK_B will pass the verification algorithm which means $\mathsf{Verify}(\mathsf{PP}, \mathsf{SK}_B, B)$ will outputs True. □

We give also proof of the selective security and succinctness of this construction in the full version of this paper [26].

4.3 Construction from Indistinguishability Obfuscation

The SNARK based construction from the previous section has some limitations: a polynomial bound on delegation depth, and no rigorously proved context-hiding property. A very simple construction without either of these shortcomings (thus achieving all the desired properties of universal witness signatures) can be obtained from virtual black-box obfuscation: we describe that construction in the full version of this paper [26]. Admittedly, however, virtual black-box obfuscation is an onerous assumption: it is known to be unachievable for general circuits in the standard model [5]. In this section, we provide a satisfactory construction using a somewhat more realistic flavor of obfuscation (indistinguishability obfuscation) together with puncturable pseudorandom functions (as is usual for iO-based constructions). Both notions are formally recalled in the full version of this paper [26].

$\underline{\mathsf{Setup}(1^\lambda)}$:
$(\mathsf{msk}, \mathsf{mvk}) \leftarrow \mathsf{Sig.Setup}(1^\lambda)$
$\mathsf{crs} \leftarrow \mathcal{G}(1^\lambda)$
$\sigma_{\mathbf{VK}_*} \leftarrow \mathsf{Sig.Sign}(\mathsf{msk}, (*, \text{“}* \leq *\text{”}, \mathsf{VK}_*))$
$\pi_* \leftarrow \mathcal{P}(\mathsf{crs}, \bot, \bot, z_*, *)$
$z_* = (\mathsf{VK}_*, \mathsf{SK}_*, \text{“}* \leq *\text{”}, \sigma_{\mathbf{VK}_*})$
return $((z_*, \pi_*), \mathsf{crs})$

$\underline{\mathsf{Delegate}(\mathsf{mvk}, \mathsf{SK}_A, A, B, w)}$:
$(z_A, \pi_A) \leftarrow \mathsf{SK}_A$
$(\mathsf{VK}_A, \mathsf{SK}_A, w_{A \leq C}, \sigma_{\mathbf{VK}_A}) \leftarrow z_A$
Assert that π_A is valid proof of z_A / $\mathcal{C}$-compliance
$(\mathsf{SK}_B, \mathsf{VK}_B) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$
$\sigma_{\mathbf{VK}_B} \leftarrow \mathsf{Sig.Sign}(\mathsf{SK}_A, (B, w_{A \leq B}, \mathsf{VK}_B))$
$z_B \leftarrow (\mathsf{VK}_B, \mathsf{SK}_B, w_{B \leq A}, \sigma_{\mathbf{VK}_B})$
$\pi_B \leftarrow \mathcal{V}(\mathsf{crs}, z_A, \pi_A, z_B, B)$
return (z_B, π_B)

Fig. 3. Construction of UWS from proof-carrying data.

$\underline{\mathsf{CheckSign}[K](\mathsf{SK}_A, A, B, w)}$:
if $f(\mathcal{F}(K, A)) == f(\mathsf{SK}_A) \wedge A >_w B$ then
  return $\mathsf{SK}_B = \mathcal{F}(K, B)$
else if $f(\mathcal{F}(K, A)) == f(\mathsf{SK}_A) \wedge A =_w B$ then
  return $\mathsf{SK}_B := \mathsf{SK}_A$
else return $\perp$

Fig. 4. The CheckSign algorithm.

Our construction of UWS from punctured PRFs and iO achieves correctness, context-hiding, and succinctness. However, besides the reliance on iO, which is a strong assumption, the maximum delegation level must again be chosen at setup time, and there is a limitation of the type of preorder relations we support. More precisely, our construction applies to *order* relations ($\leq$ is anti-symmetric) and moreover, any given element has at most polynomially many elements greater than itself. We will use the following three primitives in our construction:

1. Let $C^\flat = i\mathcal{O}(C)$ be the indistinguishability obfuscation of the circuit C;
2. Let $(\mathcal{G}, \mathcal{F})$ be a punctured PRF scheme in which $\mathcal{G}$ generates the system parameters and $\mathcal{F}$ is the keyed PRF;
3. Let f be an injective length-doubling PRG.

Here is our construction:

- $\mathsf{Setup}(1^\lambda)$:
  - We generate the PRF key K from $\mathcal{G}(1^\lambda)$ and compute the indistinguishability obfuscation $\mathsf{CheckSign}^\flat$ of the algorithm CheckSign.
  - The master signing key (signature of $*$) SK_* is the PRF value $\mathcal{F}(K, *)$ of $*$ and the public parameter PP is the obfuscated circuit $\mathsf{CheckSign}^\flat$.
- $\mathsf{Delegate}(\mathsf{PP}, \mathsf{SK}_A, A, B, w_{B \leq A})$:
  - $\mathsf{CheckSign}^\flat(\mathsf{SK}_A, A, B, w_{B \leq A})$ which is an indistinguishability obfuscation of the program CheckSign described Fig. 4.

Theorem 5. *The $i\mathcal{O}$-based construction is correct, succinct, and context-hiding.*

Proof. All three properties rely on the underlying indistinguishability obfuscation's properties; we denote this obfuscator $i\mathcal{O}$. By construction, a valid signature SK_A of identity A is equivalent to the fact that $\mathsf{SK}_A = \mathcal{F}(K, A)$. Very roughly, it follows that:

- **Correctness** If SK_B is produced by the algorithm Delegate, then $\mathsf{SK}_B = \mathcal{F}(K, B)$. We have $\mathsf{Verify}(\mathsf{PP}, \mathsf{SK}_B, B) = \mathsf{True}$.
- **Succinctness and Context-hiding** $\mathsf{SK}_A = \mathcal{F}(K, A)$ is independent of the delegation path, and its output is that of a pseudorandom function. Hence the UWS scheme is succinct and context-hiding.

A security proof and extended proofs of succinctness and the context-hiding property are provided in the full version of this paper [26]. □

5 Conclusion

In this paper, we have introduced a very general notion of delegatable signature scheme: universal witness signatures (UWS). We have formally defined the security properties that UWS should ideally satisfy, and provided four different constructions based on a range of assumptions from the existence of one-way functions to virtual black-box obfuscation, and achieving some or all of these security properties. Those constructions can be used to instantiate a number of other primitives, and provide, in particular, the first instantiations of propositional signatures, a notion with numerous interesting applications.

Each of our constructions has some limitations, however. Constructing secure, succinct and context-hiding UWS with unbouded delegation depth based on relatively weak assumptions is left as a challenging open problem. In addition, our work did not tackle the problem of anonymity for UWS-like schemes, and we only considered *unary* delegation. Those problems are also worth investigating in future work.

References

1. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. J. Cryptol. **28**(2), 351–395 (2015)
2. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: di Vimercati, S.C., Syverson, P., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005). https://doi.org/10.1007/11555827_10
3. Attrapadung, N., Libert, B., Peters, T.: Computing on authenticated data: new privacy definitions and constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 367–385. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_23
4. Backes, M., Meiser, S., Schröder, D.: Delegatable functional signatures. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016 (1). LNCS, vol. 9614, pp. 357–386. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_14
5. Barak, B., et al.: On the (im)possibility of obfuscating programs. J. ACM **59**(2), 6:1–6:48 (2012)
6. Bellare, M., Fuchsbauer, G.: Policy-based signatures. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 520–537. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_30
7. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable zero knowledge via cycles of elliptic curves. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014 (2). LNCS, vol. 8617, pp. 276–294. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_16
8. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In: STOC, pp. 111–120. ACM (2013)
9. Bitansky, N., Chiesa, A., Ishai, Y., Paneth, O., Ostrovsky, R.: Succinct non-interactive arguments via linear interactive proofs. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 315–333. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_18

10. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a linear subspace: signature schemes for network coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00468-1_5
11. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_29
12. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable signatures: new definitions and delegatable anonymous credentials. In: CSF, pp. 199–213. IEEE Computer Society (2014)
13. Chiesa, A., Tromer, E.: Proof-carrying data and hearsay arguments from signature cards. In: ICS, pp. 310–331. Tsinghua University Press (2010)
14. Chow, S.S.M., Hui, L.C.K., Yiu, S.M., Chow, K.P.: Secure hierarchical identity based signature and its application. In: Lopez, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 480–494. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30191-2_37
15. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS, pp. 40–49. IEEE Computer Society (2013)
16. Garg, S., Miles, E., Mukherjee, P., Sahai, A., Srinivasan, A., Zhandry, M.: Secure obfuscation in a weak multilinear map model. In: Hirt, M., Smith, A. (eds.) TCC 2016-B (2). LNCS, vol. 9986, pp. 241–268. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_10
17. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_37
18. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_34
19. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45760-7_17
20. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: STOC, pp. 723–732. ACM (1992)
21. Kiltz, E., Neven, G.: Identity-based signatures. In: Joye, M., Neven, G. (eds.) Identity-Based Cryptography, Cryptology and Information Security Series, vol. 2, pp. 31–44. IOS Press, Amsterdam (2008)
22. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: AsiaCCS, pp. 60–69. ACM (2010)
23. Maji, H.K., Prabhakaran, M., Rosulek, M.: Attribute-based signatures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 376–392. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19074-2_24
24. Naccache, D.: Is theoretical cryptography any good in practice? CRYPTO & CHES 2010 invited talk (2010)
25. Okamoto, T., Takashima, K.: Efficient attribute-based signatures for non-monotone predicates in the standard model. IEEE Trans. Cloud Comput. **2**(4), 409–421 (2014)
26. Qian, C., Tibouchi, M., Géraud, R.: Universal witness signatures. HAL Open Archive (2018). Full version of this paper, https://hal.archives-ouvertes.fr/hal-01814279v1

27. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC, pp. 475–484. ACM (2014)
28. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5
29. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45861-1_22

Author Index