Ed Dawson (Ed.)

LNCS 7779

Topics in Cryptology – CT-RSA 2013

The Cryptographers' Track at the RSA Conference 2013
San Francisco, CA, USA, February/March 2013
Proceedings

RSACONFERENCE2013

Springer

Ed Dawson (Ed.)

Topics in Cryptology – CT-RSA 2013

The Cryptographers' Track at the RSA Conference 2013
San Francisco, CA, USA, February 25 - March 1, 2013
Proceedings

Springer

Volume Editor

Ed Dawson
Queensland University of Technology
Institute for Future Enviroments
Brisbane, QLD 4000, Australia
E-mail: e.dawson@qut.edu.au

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-36094-7 e-ISBN 978-3-642-36095-4
DOI 10.1007/978-3-642-36095-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012955763

CR Subject Classification (1998): E.3, K.6.5, D.4.6, K.4.4, J.1, C.2.0, E.4

LNCS Sublibrary: SL 4 – Security and Cryptology

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The RSA conference has been a major international event for information security experts since its introduction in 1991. It is an annual event that attracts hundreds of vendors and thousands of participants from industry, government, and academia. Since 2001, the RSA conference has included the Cryptographers' Track (CT-RSA), which provides a forum for current research in cryptography. CT-RSA has become a major publication venue for cryptographers.

This year's RSA conference was held in San Francisco, California, from February 25 to March 1, 2013. The CT-RSA conference servers were hosted by Queensland University of Technology in Australia. This year 89 submissions were received out of which 25 papers were selected. I would like to thank the authors of all submissions. The review process was thorough with each submission receiving at least three reviews (five if the submitted paper included a Program Committee member). I wish to thank all Program Committee members as well as the subreviewers who assisted them for their hard and dedicated work in selecting the papers for CT-RSA 2013.

Two invited talks were given. The first was by Nadia Heninger: "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices." The second was by John Kelsey: "SHA3 and the Future of Hashing."

I would especially like to thank my postdoctoral research fellow Farzad Salim, who assisted me throughout the organization of the program for CT-RSA. I would also like to thank Liz McVety from the RSA conference organization for assisting with arrangements for speakers and Orr Dunkelman for the advise. Finally, I would like to thank Thomas Baigneres and Matthieu Finiasz for the iChair software that helped facilitate a smooth review process.

November 2012 Ed Dawson

Organization

Program Chair

Ed Dawson	Queensland University of Technology, Australia

Steering Committee

Orr Dunkelman	University of Haifa, Israel
Marc Fischlin	Darmstadt University of Technology, Germany
Ari Juels	RSA Laboratories, USA
Aggelos Kiayias	University of Connecticut, USA
Josef Pieprzyk	Macquarie University, Australia
Ron Rivest	MIT, USA
Moti Yung	Google, USA

Program Committee

Michel Abdalla	École Normale Supérieure, France
Masayuki Abe	NTT Secure Platform Laboratories, Japan
Josh Benaloh	Microsoft Research, USA
John R. Black	University of Colorado at Boulder, USA
Ioana Cristina Boureanu	EPFL, Switzerland
Lily Chen	NIST, USA
Seung Geol Choi	University of Maryland, USA
Ed Dawson (Chair)	Queensland University of Technology, Australia
Yvo Desmedt	UT Dallas, USA
Pierre-Alain Fouque	École Normale Supérieure, France
Jovan Dj. Golic	Telecom Italia, Italy
Tor Helleseth	University of Bergen, Norway
Matt Henricksen	Institute for Infocomm Research (I2R), Singapore
Huseyin Hisil	Izmir Yasar University, Turkey
Kwangjo Kim	Korea Advanced Institute of Science and Technology (KAIST), Korea
Lars R. Knudsen	Technical University of Denmark, Denmark
Xuejia Lai	Shanghai Jiao Tong University, China
Tanja Lange	Technische Universiteit Eindhoven, The Netherlands

Arjen Lenstra	EPFL, Switzerland
Javier Lopez	University of Malaga, Spain
Stefan Lucks	Bauhaus-Universität Weimar, Germany
Mark Manulis	University of Surrey, UK
Tsutomu Matsumoto	Yokohama National University, Japan
Chris Mitchell	Royal Holloway, UK
Atsuko Miyaji	Japan Advanced Institute of Science and Technology (JAIST), Japan
Elisabeth Oswald	University of Bristol, UK
Josef Pieprzyk	Macquarie University, Australia
Rei Safavi-Naini	University of Calgary, Canada
Palash Sarkar	Indian Statistical Institute, India
Joerg Schwenk	Ruhr-Universität Bochum, Germany
Douglas Stebila	Queensland University of Technology, Australia
Andreas Steffen	University of Applied Sciences Rapperswil, Switzerland
Willy Susilo	University of Wollongong, Australia
Routo Terada	University of São Paulo, Brazil
Huaxiong Wang	Nanyang Technological University, Singapore
Chuan-Kun Wu	IIE, Chinese Academy of Sciences, China

External Reviewers

Asli Bay	Christian Forler	Jhawar Mahavir
Anja Becker	Eiichiro Fujisaki	Ho AuMan
Daniel J. Bernstein	Wei Gao	Dustin Moody
Sanjay Bhattacherjee	Zheng Gong	Sean Murphy
Andrey Bogdanov	Teng Guo	Pablo Najera
Julia Borghoff	Kishan Chand Gupta	Kris Narayan
Joppe Bos	Mitsuhiro Hattori	Ta KhoaNguyen
Colin Boyd	Nadia Heninger	Juanma Gonzalez Nieto
Xavier Boyen	Jialin Huang	Ryo Nishimaki
Tom Carlson	Qiong Huang	David Nuñez
Debrup Chakraborty	Tibor Jager	Katsuyuki Okeya
Jiageng Chen	Stanislaw Jarecki	Kazumasa Omote
Jie Chen	Jeremy Jean	Jose A. Onieva
Sherman Chow	Dimitar Jetchev	Matthew Parker
Clavier Christophe	Shaoquan Jiang	Rene Peralta
Craig Costello	Pierre Karpman	Krzysztof Pietrzak
Onete Cristina	Oleksandr Kazymyrov	Somindu Ramanna
Romar dela Cruz	John Kelsey	Jothi Rangasamy
Patrick Derbez	Kaleb Lee	Peter Schwabe
Alexandre Duc	Hoon Wei Lim	Shashank Singh
Gerardo Fernandez	Anders Smedstuen Lund	Daniel Smith

Le Su	Vincent Verneuil	Weijia Xue
Petr Susil	Huaxiong Wang	Guomin Yang
Stefan Tillich	Jakob Wenzel	Reza Z'aba
Michael Tunstall	Marcin Wojcik	Liangfeng Zhang
Meltem Sonmez Turan	Kenneth Wong	Rui Zhang
Serge Vaudenay	Keita Xagawa	Yun Zhang
Roel Verdult	Hong Xu	Jinmin Zhong
Damien Vergnaud	Jing Xu	Yongbin Zhou

Table of Contents

Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations

Aurélie Bauer, Eliane Jaulmes, Emmanuel Prouff, and Justine Wild

ANSSI, 51, Boulevard de la Tour-Maubourg, 75700 Paris 07 SP, France
`firstname.name@ssi.gouv.fr`

Abstract. Since the introduction of side-channel attacks in the nineties, RSA implementations have been a privileged target. A wide variety of countermeasures have been proposed and most of practical attacks are nowadays efficiently defeated by them. However, in a recent work published at ICICS 2010, Clavier *et al.* have pointed out that almost all the existing countermeasures were ineffective if the attacks are performed with a *modus operandi* called *Horizontal*. Such attacks, originally introduced by Colin Walter at CHES 2001, involve a single observation trace contrary to the classical attacks where several ones are required. To defeat Horizontal attacks, the authors of the ICICS paper have proposed a set of new countermeasures. In this paper, we introduce a general framework enabling to model both Horizontal and classical attacks (called *Vertical*) in a simple way. This framework enables to enlighten the similarities and the differences of those attack types. From this formalism, we show that even if Clavier *et al.*'s countermeasures thwart existing attacks, they do not fully solve the leakage issue. Actually, flaws are exhibited in this paper and efficient attacks are devised. We eventually propose a new countermeasure.

1 Introduction

Side-Channel Analysis (SCA) is a cryptanalytic technique that consisting in exploiting the *side channel leakage* (*e.g.* the power consumption, the electromagnetic emanations) produced during the execution of a cryptographic algorithm embedded on a physical device. It uses the fact that this leakage is statistically dependent on the intermediate variables that are processed. Some of these variables are *sensitive* in the sense that they are related to secret data, thus reaching information on them enables efficient key recovery attacks [3, 9, 15]. Since the publication of the first attacks, many papers describing either countermeasures or attack improvements have been published (see [3,4,16] for example). Among these improvements, *higher-order SCA* are of particular interest. They extend the initial concept by considering a set of several instructions instead of a single one and circumvent many countermeasures proposed in the literature (*e.g.* [4, 10]). Another significant improvement has been proposed initially by Walter [19] and then studied more deeply by Clavier *et al.* in [5]. Essentially, it consists in a new *modus operandi* called *Horizontal*, in which sensitive information is extracted from a single measurement split into several parts. It differs from the classical *Vertical* mode where information is obtained from different algorithm executions. Horizontal mode applies when the same guessable sub-part of a secret is involved in many internal operations during the overall algorithm processing. This is particularly the case for RSA implementations where the exponentiation is composed of small multiplications that all depend on the same secret exponent sub-part. As noticed in [5,19], a striking point is that classical countermeasures (*e.g.* the exponent or the message blinding techniques), efficient in the Vertical context, turn out to be almost ineffective in the Horizontal one. This makes the construction of new appropriate countermeasures a real issue for the security of embedded implementations of RSA and similar algorithms. An attempt to design such countermeasures thwarting both types of attacks has been done in [5]. Actually, we show in this paper that these countermeasures (Sections 4 and 5) do not completely remove the leakage, even if they thwart some attacks. To exhibit the flaws and, eventually, to propose new countermeasures (Section 6), we first introduce a framework enabling to formally study the resistance of an implementation against side channel attacks in both Horizontal and Vertical modes (Sections 2 and 3). This framework could be used to further analyse the security of other algorithms' implementations than RSA ones. On the other hand, the countermeasure proposed in this paper is a first step towards an efficient and effective security against Horizontal attacks. The definition of alternative countermeasures is a new open avenue for further research.

2 A Comprehensive Study of Side-Channel Analyses

In the following, a general framework is introduced which enables to describe most of the existing attacks in a similar way and to identify their core differences (actually the leakage pre-treatment, the leakage model and the statistical test).

2.1 A General Framework for Side-Channel Analyses

Notations. A realization of a random variable X is referred to as the corresponding lower-case letter x. A *sample* of observations of X is denoted by (x) or by $(x_i)_i$ when an indexation is needed. In this case, the global event is sumed up as $(x) \hookleftarrow X$. The i^{th} coordinate of a variable X (resp. x), viewed as a vector, is denoted by $X[i]$ (resp. $x[i]$). As usual, the notation $\mathbb{E}[X]$ refers to the mean of X. For clarity reasons we sometimes use the notation $\mathbb{E}_X[\cdot]$ to enlighten the variable over which the mean is computed.

All the attacks below target a variable $Z(k, X)$ defined as the output of a specific computation (*e.g.* a multiplication) performed by the device and parametrized by a secret sub-part k and a public variable X[1]. In the following, we shall use Z instead of $Z(k, X)$ if there is no ambiguity on k and X.

To recover information on k, the attacks are performed on a sample of observations related to the processing of Z by the device. Each of those observations,

[1] We shall sometimes need to consider the known value as a pair of variables: in this case we will use the notation (X, Y) instead of X.

such as power consumption, electromagnetic emanations, and so on, is usually composed of several physical measurements corresponding to leakages at different times t_i. It can hence be viewed as a realization of a multivariate random variable $\boldsymbol{L}$ whose coordinates $\boldsymbol{L}[i]$ satisfy:

$$\boldsymbol{L}[i] = \varphi_i\big(Z\big) + \beta_i \ , \tag{1}$$

where φ_i only depends on the device behaviour at time t_i during the processing of Z and β_i is an independent Gaussian noise with zero mean and standard deviation σ_i. The function φ_i is *a priori* unknown. The index i will be sometimes omitted. In this case, it is assumed that the same function is associated to all the time indices.

An SCA is based on the *Hypothesis Testing principle* [13]. To make this test, a set of *prediction values* $\boldsymbol{h}_j$ are deduced from each hypothesis $\hat{k}$ on k and from the sample of implementation inputs (x_j) corresponding to the observations. This step involves a *leakage model function* $\mathbf{m}$ that must have been priorly chosen by the attacker (for instance based on its knowledge on the attacked device architecture). With this model function, the prediction values $\boldsymbol{h}_j$ are built s.t. $\boldsymbol{h}_j = \mathbf{m}(z(\hat{k}, x_j))$. Eventually, the adversary uses a distinguisher $\varDelta$ to compare the $\boldsymbol{h}_j$ with the observations $\boldsymbol{l}_j \hookleftarrow \boldsymbol{L}|X = x_j$.

The overall set of SCA is usually split in two subclasses. The first one, called *simple Side-Channel Analysis*, contains all attacks where observations only need to be done on a single value of the public input parameter (this implies that all the x_j are equal to a same value, say x). This set contains S-PA [14], S-EMA [8,18] or S-TA (Timing Analysis) [14]. The second subclass, called *advanced Side-Channel Analysis*, includes attacks where observations of the targeted internal processing must be done for several public input parameters. In particular, it contains *univariate SCA* attacks such as DPA [15], CPA [3] or MIA [9] and *multivariate SCA* attacks such as HO-DPA [15, 17] or HO-MIA [1]. We give hereafter a more formal description of those two subclasses.

Simple SCA. The class of simple SCA includes all Vertical or Horizontal SCA where the adversary makes observations for a single algorithm input. Table 1 provides a description of a simple Side-Channel Analysis[2].

Remark 1. In theory, simple SCA may be conducted with a single observation. In practice however, it is often necessary to use several observations of the processing for the same variable x in order to reduce the noise impact. In this case, the statistical distinguisher $\varDelta$ may for instance involve a preliminary step consisting in averaging the observations sample.

Advanced SCA. All the attacks where observations must involve different inputs belong to the advanced SCA category. This kind of attacks follows the outlines given in Table 2.

[2] In contexts where the adversary is not allowed to choose the algorithm input but knows it, the first step just aims at fixing the input value for the rest of the attack.

Table 1. Simple Side-Channel Analysis

```
1. Choose a value $x$ for $X$.
2. Measure a sample $(\boldsymbol{l}_j)_j \hookleftarrow (\boldsymbol{L}|X = x)$ of $N$ leakages.
3. Select a distinguisher $\Delta$ and choose a model function $\mathbf{m}$.
4. For each hypothesis $\hat{k}$ on $k$, compute $\boldsymbol{h} = \mathbf{m}(z(\hat{k}, x))$.
5. For each $\hat{k}$, compute $\Delta[\hat{k}] = \Delta\big[(\boldsymbol{l}_j)_j, \boldsymbol{h}\big]$.
6. Deduce from $\Delta[\cdot]$ information on $k$.
```

Table 2. Advanced Side-Channel Analysis

```
1. Get $N$ measurements $(\boldsymbol{l}_j, x_j)_j \hookleftarrow (\boldsymbol{L}, X)$.
2. Select a distinguisher $\Delta$ and choose a model function $\mathbf{m}$.
3. For each hypothesis $\hat{k}$ on $k$ build a set of predictions $h_j$
   such that $\boldsymbol{h}_j = \mathbf{m}(z(\hat{k}, x_j))$.
4. For each $\hat{k}$, compute $\Delta[\hat{k}] = \Delta\Big[(\boldsymbol{h}_j)_j, (\boldsymbol{l}_j)_j\Big]$
5. Deduce from $\Delta[\cdot]$ information on $k$.
```

Remark 2. Depending on the statistical treatment processed by the distinguisher, the latter one may include a particular leakage post-processing $\mathcal{E}$. This post-treatment may be used to select some particular points in the leakage traces and, possibly, to combine them. For instance, in a second-order advanced SCA involving the mutual information as distinguisher, the function $\mathcal{E}$ can be defined such that $\mathcal{E}\big(\boldsymbol{L}\big) = \big(\boldsymbol{L}[p], \boldsymbol{L}[q]\big)$ for some constant indices (leakage times) p and q. In a second-order advanced SCA involving the correlation coefficient as distinguisher, $\mathcal{E}$ may be defined such that $\mathcal{E}\big(\boldsymbol{L}\big) = (\boldsymbol{L}[p] - \mathbb{E}(\boldsymbol{L}[p])) \cdot (\boldsymbol{L}[q] - \mathbb{E}(\boldsymbol{L}[q]))$. Moreover, the choice of the model function must be done in accordance with the distinguisher (see *e.g.* [17] and [9]).

2.2 Leakage Measurements and Observations

In the literature, two main ways have been defined to get the observations $\boldsymbol{l}_j$ during the first step of the attacks in Tables 1 and 2. The first method simply consists in executing the implementation several times (with the same input in simple SCA or with several ones in advanced SCA) and in defining $\boldsymbol{l}_j$ as the observation related to the j^{th} algorithm execution. Those attacks are called *Vertical*. The second method refers to attacks where a single execution is needed and where each $\boldsymbol{l}_j$ corresponds to the observation of a processing at a different time period during this execution. In this case, the index j refers to the time period. The underlying assumption is that all the observations rely on the same internal calculus of $Z(k, X)$, parametrized by a same secret k and different known values

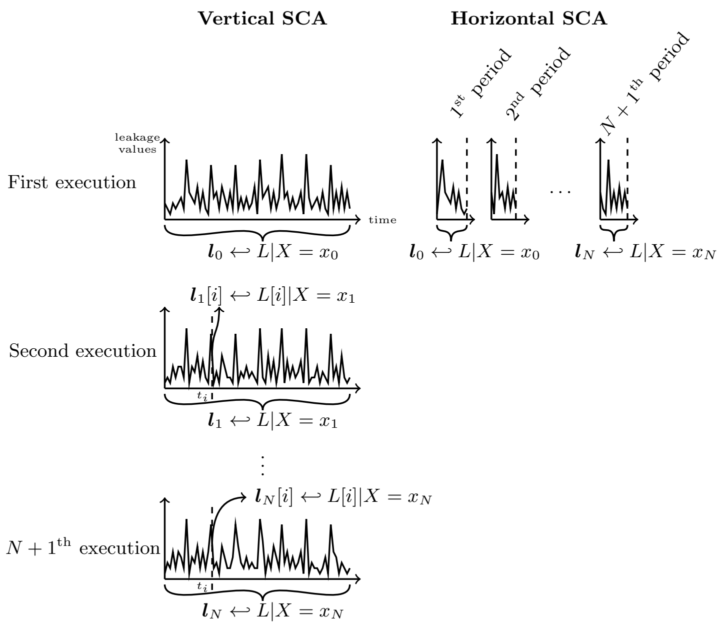

Vertical SCA
Horizontal SCA
leakage values
time
First execution
Second execution
$N+1^{\text{th}}$ execution
1^{st} period
2^{nd} period
$N+1^{\text{th}}$ period
$\boldsymbol{l}_0 \hookleftarrow L|X = x_0$
$\boldsymbol{l}_N \hookleftarrow L|X = x_N$
$\boldsymbol{l}_1[i] \hookleftarrow L[i]|X = x_1$
$\boldsymbol{l}_1 \hookleftarrow L|X = x_1$
$\boldsymbol{l}_N[i] \hookleftarrow L[i]|X = x_N$
t_i

Fig. 1. Vertical and Horizontal SCA

x_j in advanced SCA, or a constant one x in simple SCA. Attacks corresponding to this *modus operandi* are called *Horizontal*. Figure 1 illustrates the notations and the differences between the two *modus operandi*.

All the attacks discussed in Section 2.1 can be either Vertical or Horizontal[3]. Even if the Horizontal or Vertical characteristic of an SCA has no impact on the attack steps themselves (as described in Tables 1 and 2), it impacts the implementation security analysis. Indeed, we will see in Section 4 that a countermeasure may become ineffective when going from one category of attacks to another one. We illustrate this in the context of secure RSA implementations.

2.3 Taxonomy

Based on the discussions conducted in previous sections, we propose here a general taxonomy for simple and advanced side-channel attacks. To name an attack we propose to use the convention [XXX]-[YYY]-[ZZZ] where:

– XXX equals either S for simple SCA or is a reference to the statistical tool for advanced SCA (*e.g.* C for Correlation, MI for Mutual Information, ML for Maximum Likelihood, LR for Linear Regression, etc.). In case of multivariate SCA, we propose to pad the order/dimension followed by O at the left of the distinguisher letter.

[3] Possibly, the observations acquisition phase may mix horizontal and vertical techniques. In this case, the attack will be termed *Rectangle*.

- YYY is an acronym referring to the leakage type; PA for Power Analysis, EMA for Electromagnetic Analysis, TA for Timing Attacks, etc.
- ZZZ is optional and may be used to specify if the attack is profiled or not. In this case, ZZZ is replaced by P (for Profiling) or UnP (for UnProfiling). For instance, Template attack is a ML-PA-P attack.

Of course, all those attacks can be applied either on a Vertical or Horizontal mode. Figure 2 illustrates the taxonomy for some existing attacks.

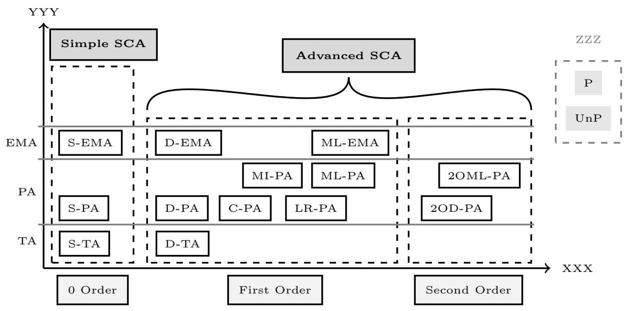


Fig. 2. Side-Channel Attacks

In the following sections, we focus on Horizontal SCA in the RSA context. We will recall the existing attacks and will discuss about the effectiveness of the Vertical SCA countermeasures against Horizontal SCA.

3 RSA Context

3.1 Operation Flows in RSA Exponentiations

The execution flow of an RSA implementation is usually viewed as the succession of only two different operations: a modular squaring and a modular multiplication respectively denoted by O_0 and O_1. For convenience, we will assume that both operations are bivariate and defined such that $\mathrm{O}_i(X,Y) = X^i Y^{\bar{i}} \cdot Y$. For instance, the left-to-right *Square and Multiply* algorithm parametrized by a d-bit long secret k (the most significant bit is assumed to be equal to 1) and a public modulus n, operating on a message X can be associated to the following sequence:

$$Y \leftarrow \mathrm{O}_{f(0)}(X,Y), Y \leftarrow \mathrm{O}_{f(1)}(X,Y), \cdots, Y \leftarrow \mathrm{O}_{f(N)}(X,Y) \ , \tag{2}$$

where Y is the updated intermediate result (initially set to 1), N denotes the value $d + \mathrm{HW}(k)$ and the binary function f is defined as:

$$f(j) = \begin{cases} j & \text{, if } j = 0,1 \\ \overline{f(j-1)} \cdot k\left[d-1-\sum_{i=0}^{j-2} \overline{f(i)}\right] & \text{, otherwise.} \end{cases} \tag{3}$$

The operations' flow in (2) is illustrated on Figure 3.

To defeat simple SCA against RSA implementations, a classical countermeasure is to insert dummy multiplications in order to have a *regular* algorithm. This leads to the definition of the so-called *Square and Multiply Always* algorithm which may be associated with the sequence below where each square is followed by a multiplication whatever the secret k:

$$Y_1 \leftarrow \mathrm{O}_0(X, Y_1), Y_{k[d-1]} \leftarrow \mathrm{O}_1(X, Y_1), Y_1 \leftarrow \mathrm{O}_0(X, Y_1), Y_{k[d-2]} \leftarrow \mathrm{O}_1(X, Y_1), \cdots ,$$
$$Y_1 \leftarrow \mathrm{O}_0(X, Y_1), Y_{k[0]} \leftarrow \mathrm{O}_1(X, Y_1) \ , \quad (4)$$

with Y_0 denoting a garbage variable and Y_1 a working register initially set to 1 (and playing the same role as Y in (2)).

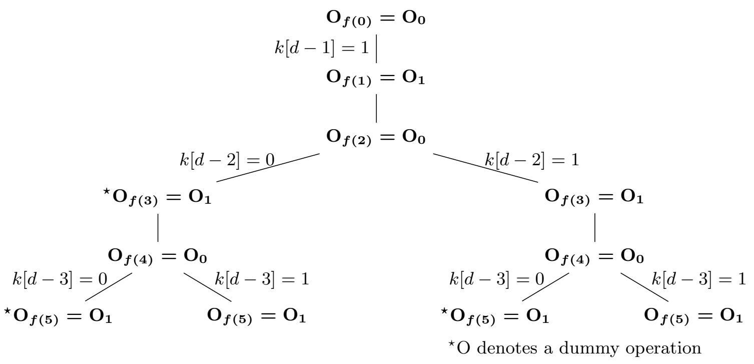

$\mathbf{O}_{f(0)} = \mathbf{O}_0$
$k[d-1]=1$
$\mathbf{O}_{f(1)} = \mathbf{O}_1$
$\mathbf{O}_{f(2)} = \mathbf{O}_0$
$k[d-2]=0$
$k[d-2]=1$
${}^\star\mathbf{O}_{f(3)} = \mathbf{O}_1$
$\mathbf{O}_{f(3)} = \mathbf{O}_1$
$\mathbf{O}_{f(4)} = \mathbf{O}_0$
$\mathbf{O}_{f(4)} = \mathbf{O}_0$
$k[d-3]=0$
$k[d-3]=1$
$k[d-3]=0$
$k[d-3]=1$
${}^\star\mathbf{O}_{f(5)} = \mathbf{O}_1$
$\mathbf{O}_{f(5)} = \mathbf{O}_1$
${}^\star\mathbf{O}_{f(5)} = \mathbf{O}_1$
$\mathbf{O}_{f(5)} = \mathbf{O}_1$
*O denotes a dummy operation

Fig. 3. First Loops of Square and Multiply Always algorithm

Remark 3. An improved version of the Square and Multiply Always algorithm, based on the *Montgomery Ladder* trick [11], is often preferred as it is more resistant to the so-called *Safe-Error* attacks [20]. In this version, there is no garbage variable and $Y_{k[i]}$ is used in the subsequent operation even if $k[i]$ is equal to 0. We point out here that this version and the Square and Multiply Always algorithm have exactly the same vulnerabilities with respect to advanced SCA. Indeed, in both cases, each loop iteration in the exponentiation processes the same operations and only the memory manipulation is different.

The granularity of the sequence descriptions in (2) and (4) is not fine enough to investigate advanced SCA. Those attacks indeed require the identification of intermediate results depending on small sub-parts of the input parameters. To enable such an identification, the execution flows must be rewritten as a succession of operations on ω-bit words[4]. Let us assume that modular squarings and multiplications are implemented with the schoolbook multiplication called

[4] The value ω typically depends on the device architecture and is usually equal to 8, 16 or 32.

Long Integer Multiplication (LIM for short) followed by a Barrett reduction (for self-contentedness we recall the LIM algorithm in Appendix A). The variables X and Y are then represented as base-2^ω vectors[5] $(X[a])_{0\leq a\leq t}$ and $(Y[b])_{0\leq b\leq t}$ with $t = \lfloor \frac{\log_2(X)}{\omega} \rfloor$. After this rewriting, we get the following decomposition of an operation O_i, where we only exhibited the intermediate base-2^ω multiplications $Z[a,b] \leftarrow X[a]^i Y[a]^{\bar{i}} \cdot Y[b]$:

$$\begin{array}{cccc}
Z[0,0]{\leftarrow}X[0]^iY[0]^{\bar{i}}\cdot Y[0], & Z[0,1]{\leftarrow}X[0]^iY[0]^{\bar{i}}\cdot Y[1], & \cdots, & Z[0,t]{\leftarrow}X[0]^iY[0]^{\bar{i}}\cdot Y[t] \\
Z[1,0]{\leftarrow}X[1]^iY[1]^{\bar{i}}\cdot Y[0], & \cdots & \cdots, & Z[1,t]{\leftarrow}X[1]^iY[1]^{\bar{i}}\cdot Y[t] \\
\vdots & \vdots & \vdots & \vdots \\
Z[t,0]{\leftarrow}X[t]^iY[t]^{\bar{i}}\cdot Y[0], & \cdots & \cdots, & Z[t,t]{\leftarrow}X[t]^iY[t]^{\bar{i}}\cdot Y[t]
\end{array} .$$

Fig. 4. Decomposition of the operation $\mathrm{O}_i(X,Y) = X^i \cdot Y^{\bar{i}} \cdot Y$

3.2 Attacks Targets

When applied against the operations' sequences (2) or (4), advanced SCA aim at recovering all the bits of k one after another from the left to the right. Here, we assume that the most significant bit of k is 1 and we show in this section and the next one how advanced SCA succeed in recovering the value of $k[d-2]$. The attacks may further be repeated to fully recover k. To simplify the notations, we denote the secret bit $k[d-2]$ by s. In a classical left-to-right Square and Multiply algorithm, s is involved for the first time in the operation $\mathrm{O}_{f(3)}$. In this case, one can develop the operands of $\mathrm{O}_{f(3)}$ in terms of s and X. Actually, according to (3) we have $f(3) = s$, which means that $\mathrm{O}_{f(3)}$ corresponds to the processing $X^2 \cdot X^{2-s}$ (*i.e.* $Y = X^2$ in Figure 4). In a left-to-right Square and Multiply Always algorithm, the value s impacts on the fifth operation. Indeed, depending on s, the result of the fourth operation has either been put into the working register or in the garbage register. As a consequence, the fifth operation (which is always a squaring O_0) corresponds to the processing $X^{2+s} \cdot X^{2+s}$ (*i.e.* $Y = X^{2+s}$ in Figure 4). Eventually, depending on the algorithm we deduce that the elementary base-2^ω multiplications $Z[a,b]$ satisfy[6]:

– Square and Multiply (operation $\mathrm{O}_{f(3)}$ in (2)):

$$Z[a,b] = X^2[a] \cdot X^{2-s}[b] \ . \qquad (5)$$

[5] Without loss of generality, we assume that X and Y have the same length t. This possibly implies that the binary representation of one of them has been left-padded with 0s.

[6] We alert the reader on the fact that, in this paper, we make a distinction between the notations $X^i[a]$ and $(X[a])^i$: the first one denotes the $(a+1)^{\text{th}}$ coordinate of the base-2^ω representation of the value X^i, whereas the second one denotes the rising at the power i of the $(a+1)^{\text{th}}$ coordinate of the base-2^ω representation of the value X.

– Square and Multiply Always (fifth operation in (4)):

$$Z[a,b] = X^{2+s}[a] \cdot X^{2+s}[b] \ . \tag{6}$$

Equations (5) and (6) show that each intermediate result $Z[a,b]$ depends on s. This implies that the observation $\boldsymbol{L}_{a,b}$ related to the manipulation of $Z[a,b]$ by the device leaks information on s. To exploit this leakage in a vertical advanced SCA, the pair of indices (a,b) is fixed and the observations are measured for **different** values $x \hookleftarrow X$ of the algorithm input. In an Horizontal advanced SCA, the observations are performed for a **single** value $x \hookleftarrow X$ but different pairs of indices $(a,b) \in [0;t] \times [0;t]$ (in the latter case, a and b are viewed as random variables and will be denoted by capital letters).

3.3 Horizontal Attacks

In this section, we are interested in Horizontal analyses such as the *Big Mac attack* [19] and the *Horizontal Correlation Analysis* [5]. For a fixed value $x \hookleftarrow X$ but various pairs (a,b), we assume that the adversary observes the device behavior $\boldsymbol{l}_{a,b}$ during the processing of the intermediate results $z[a,b]$.

Big Mac Attack. This attack is a *Collision Analysis*, designed in the case of the Square and Multiply algorithm when the adversary does not known the exponentiation input x. The principle consists in recovering the secret key k from the most significant bit to the least significant one. According to Equation (5), elementary operations involved in $\mathrm{O}_{f(3)}$ during the modular exponentiation can be either of the shape $x^2[a] \cdot x^2[b]$ when s equals 0, or of the form $x^2[a] \cdot x[b]$ when s equals 1. As a consequence, if the attacker is able to determine whether the leakage traces $\boldsymbol{l}_{a,b}$, involved in this operation, correspond to multiplications by $x^2[b]$ or by $x[b]$, then the value of s will easily be recovered. In order to make this distinction, the adversary performs a collision attack between the traces $\boldsymbol{l}_{a,b}$ corresponding to $\mathrm{O}_{f(3)}$ and the traces $\boldsymbol{l}'_{a,b}$ related to another multiplication involving x as operand (*e.g.* the operation $\mathrm{O}_{f(1)}$ which defines the multiplication of 1 by the input x in the Square and Multiply algorithm). To this purpose, the attacker uses for instance the average leakages $(\frac{1}{t+1}\sum_a \boldsymbol{l}_{a,b})_b$ and $(\frac{1}{t+1}\sum_a \boldsymbol{l}'_{a,b})_b$, and after selecting a distinguisher Δ, *e.g.* the *Euclidean Distance*, computes the value $\Delta((\frac{1}{t+1}\sum_a \boldsymbol{l}'_{a,b})_b, (\frac{1}{t+1}\sum_a \boldsymbol{l}_{a,b})_b)$ in order to validate or invalidate the hypothesis $s = 1$. As explained before, the Big-Mac attack has originally been described as a *Collision Analysis* for unknown exponentiation inputs and a non-regular Square and Multiply algorithm[7] (see bold notations on right-hand sided leaf in Figure 5).

Horizontal C-PA. Contrary to the previous attack, this one has been described in the context of *Atomic Square and Multiply* implementations, and also applies to the *Square and Multiply Always* algorithm (Sequence (4)) when the input x is

[7] This principle can be extended to *Sliding Windows* implementations, see Walter's original paper [19].

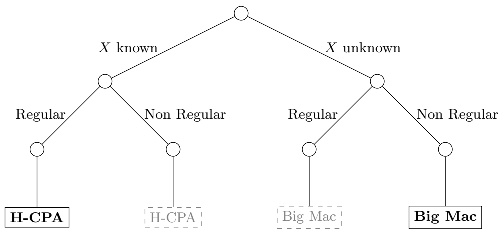


Fig. 5. Big-Mac attack and Horizontal CPA classification

known to the adversary. To recover the key-bit s corresponding to the variables $Z[a, b]$ defined in (6), the attacker involves a well-chosen model function (*e.g.* the Hamming weight) and for each key-bit hypothesis $\hat{s} \in \{0, 1\}$, computes the set of predictions $\boldsymbol{h}_{a,b} = \mathbf{m}(Z[a, b])$ (where s is replaced by $\hat{s}$ in (6)). Eventually, the Pearson coefficient ρ is chosen as distinguisher and the discrimination is done by processing $\rho[(\boldsymbol{h}_{a,b})_{a,b}, (\boldsymbol{l}_{a,b})_{a,b}]$. The applicability of this attack has been illustrated on Figure 5, see bold notations for the original description of the attack.

Extension of these Attacks. Even if the Big Mac Attack has been initially introduced for unknown exponentiation inputs, it can of course be adapted to known entries. Indeed, another way to proceed consists in using the model function $\mathbf{m} : a, \hat{s} \mapsto \frac{1}{t+1} \sum_a \hat{\varphi}(x^2[a] \cdot x^{1+\hat{s}}[b])$ where $\hat{\varphi}$ is chosen according to the device specificities (*e.g.* $\hat{\varphi}$ equals to the Hamming weight function). In the framework proposed in Section 2, the *Big Mac Attack* can thus become an *Horizontal ED-PA* (with ED standing for *Euclidean Distance*). This could be illustrated on Figure 5 by adding Big Mac in each leaf of the left hand-sided sub-tree. In addition, one can also use Big Mac attack to target regular implementations, such as modular exponentiations using the Square and Multiply always algorithm. Indeed, Equation (6) shows that operation $\mathrm{O}_{f(4)}$ is either $x^2[a] \cdot x^2[b]$ when s equals 0, or $x^3[a] \cdot x^3[b]$ when s equals 1. In that case, the attacker can average on the second multiplications operands instead of the first one (as done in the previous attacks). This leads to the computation of the values $(\frac{1}{t+1} \sum_b \boldsymbol{l}'_{a,b})_a$ coming from operation $\mathrm{O}_{f(3)}$ (which corresponds to $x^2[a] \cdot x[b]$) and $(\frac{1}{t+1} \sum_b \boldsymbol{l}_{a,b})_a$ issued from $\mathrm{O}_{f(4)}$. From that point and as before, the attacker evaluates a distinguisher Δ (*e.g.* the Euclidean Distance) to determine whether the multiplication has been performed with $x^2[a]$ or with $x^3[a]$, which leads to recover s. Eventually, the same process can be applied to guess the following remaining bits of the secret key k. This extension of the Big Mac attack is illustrated on Figure 5 by the grey-dotted box.

Applying Horizontal C-PA from the Square and Multiply always implementation to the non-regular one is obvious (see also the grey-dotted box on the scheme).

As seen in this section, the Big Mac attack and the Horizontal C-PA can both be applied not only in their original contexts but against Square and Multiply and Square and Multiply always implementations. Their success indeed does not depend on the structure of the exponentiation algorithm. They can moreover be applied in both known input and unknown input modes. In the sequel, we use those observations to argue that the countermeasures proposed to defeat Horizontal C-PA, are in fact sensible to Big Mac like attacks.

4 Existing Countermeasures

The most popular countermeasures against Vertical advanced SCA is the *exponent blinding* and the multiplicative/additive *message blinding* (*e.g.* [6,14]). The first countermeasure implies that all the observations in the adversary hands correspond to different secrets/exponents. The second countermeasure implies that no intermediate variable depends on the algorithm input. In the two cases, it becomes impossible to make predictions and the Vertical first-order advanced SCA fail.

The exponent blinding countermeasure perfectly illustrates that the effectiveness of a countermeasure may totally change when passing from Vertical to Horizontal contexts. Indeed, when the exponent randomization is applied, all the variables $Z[a, b]$ defined as in (5) or (6) will depend on the same masked bit $\tilde{s}$. As a consequence, the Horizontal advanced SCA described in Section 3 will succeed in recovering it. As the knowledge of the blinded exponent provides the adversary with the same capabilities as knowing s itself (*e.g.* it can produce the same signatures) the attack may be considered as successful. In [5], the authors also argue that message blinding thwart Horizontal attacks only if the bit-length λ of the random value R is greater than 32 bits. For smaller values of λ, an efficient attack is indeed exhibited. As a consequence of the exponent blinding ineffectiveness and of the message blinding inefficiency, there is a real lack of countermeasures against Horizontal attacks. This led Clavier *et al.* to propose the following three countermeasures [5]:

- *Blind Operands in LIM.* The first countermeasure proposed in [5] consists in applying a *full blinding* on the words $X[a]$ and $Y[b]$, *i.e.* to substitute in the LIM algorithm the operation $X[a] \cdot Y[b]$ by $(X[a] - R_1)(Y[b] - R_2) + R_1 \cdot Y[b] + R_2 \cdot X[a] - R_1 \cdot R_2$, where R_1 and R_2 are two ω-bit random values. For efficiency reasons, the authors propose to compute once the values $R_1 \cdot Y[b]$, $R_2 \cdot X[a]$ and $R_1 \cdot R_2$ and to store them. The complexity of this countermeasure is $(t+1)^2+2(t+1)+1$ ω-bit multiplications (the unprotected LIM requires $(t + 1)^2$ multiplications) and $4\log_2(n) + 2\omega$ bits of additional storage, where n is the RSA modulus.
- *Randomize One Loop in LIM and Blind.* The second countermeasure in [5] starts from the first one and mixes it with a randomization of the order in which the words $X[a]$ are involved in the LIM. This method consists in using a permutation vector applied to the words $X[a]$ and in masking the words

$Y[b]$. This countermeasure requires $(t+1)^2+t+1$ ω-bit multiplications and $2\log_2(n)$ bits of additional storage.
- *Randomize the Two Loops in LIM.* This countermeasure is a variant of the second one. In this case, the authors fully randomize the order of the processings of the $Z[a,b]$ variables. As an advantage no operand in the LIM needs to be blinded anymore. However, the drawback is that two random permutation vectors have to be stored. No extra ω-bit multiplication compared to the unprotected LIM is needed here.

In the next section, we argue that the three countermeasures below do not fully hide the first-order leakage and we exhibit efficient attacks.

5 Attacks against Horizontal SCA Countermeasures

The attacks presented in this section are described in the Square and Multiply Always setting but are straightly applicable in the classical Square and Multiply setting.

Blind Operands in LIM. In this case, the variable $Z[a,b]$ in (6) becomes:

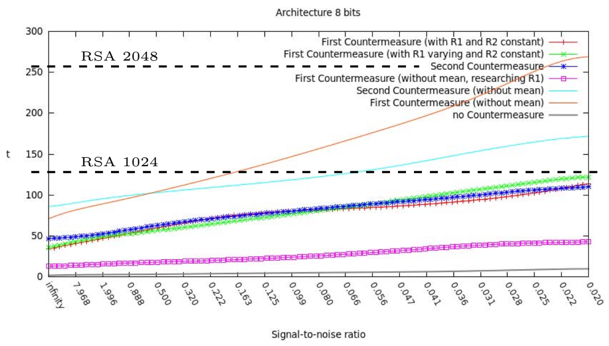
$$\tilde{Z}[a,b] = (X^{2+s}[a] - R_1)\cdot(X^{2+s}[b] - R_2)\ . \tag{7}$$

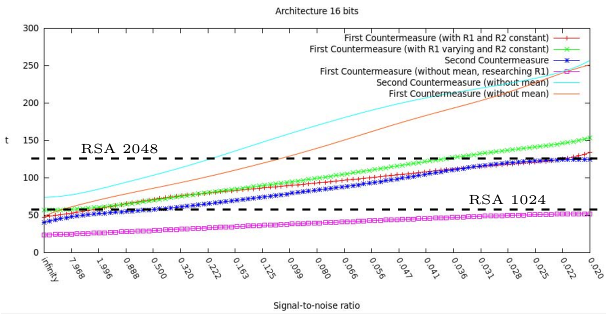
According to (1), the observation $\boldsymbol{l}_{a,b}$ of the $\tilde{Z}[a,b]$ processing satisfies $\boldsymbol{l}_{a,b} \hookleftarrow \varphi_{a,b}\big(\tilde{Z}[a,b]\big)+\beta_{a,b}$, where it can be checked that $\sum_a \tilde{Z}[a,b]$ depends on $X^{2+s}[b]$, and hence on s. This dependency can be exploited in a Horizontal C-PA by correlating the means $\overline{\boldsymbol{l}}_{\cdot,b} = \frac{1}{t+1}\sum_a \boldsymbol{l}_{a,b}$ with the predictions:

$$\boldsymbol{h}_b = \tfrac{1}{t+1}\textstyle\sum_a \mathbf{m}_{a,b}\big(X^{2+\hat{s}}[a]\cdot X^{2+\hat{s}}[b]\big)\quad , \tag{8}$$

for $\hat{s}=0$ and $\hat{s}=1$ and for $\mathbf{m}_{a,b}$ being an estimation of the unknown function $\varphi_{a,b}$. The rationale behind the definition of $\boldsymbol{h}_b$ in (8) may be found in the extended version of this paper [2]. As illustrated in Figure 6, it may be observed that the attack is still effective if the mask R_1 is different for all the words $X^{2+s}[a]$ since the leakages are averaging over indices a to compute the predictions $\boldsymbol{h}_b$.

Simulation Results. Experiments have been performed to check that the predictions $\boldsymbol{h}_b$ in (8) are consistent with the means $\overline{\boldsymbol{l}}_{\cdot,b}$ of the observations $\boldsymbol{l}_{a,b}$ when the hypothesis $\hat{s}$ on s is valid. Indeed, in that case a correlation peak can be observed, making the adversary guess the right key-bit value s. Then, processing iteratively, the whole secret k can finally be recovered. The results that have been obtained are sumed up in Figure 6 for a model function $\mathbf{m}_{a,b}$ and φ defined as the Hamming weight. Each point on the curve corresponds to the smallest number t needed to achieve a 90% success rate, according to the noise standard deviation. Attacks are reported for an architecture size ω in $\{8,16,32\}$.





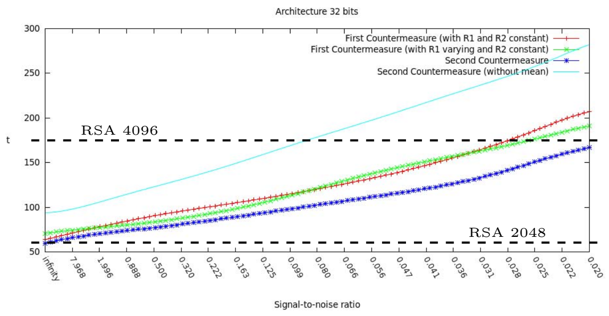


Fig. 6. Evolution of the ω-bit length t of X and Y (y-axis log-scaled) to achieve a 90% success rate depending on the noise standard deviation σ (x-axis log-scaled).

Randomize One Loop in LIM and Blind. In this case, a random permutation α over $[0; t]$ is generated before each new exponentiation and the variable $Z[a, b]$ in (6) becomes:

$$\tilde{Z}[a, b] = X^{2+s}[\alpha(a)] \cdot (X^{2+s}[b] - R) \ . \tag{9}$$

The randomization of the manipulations of the words $X^{2+s}[a]$ does not modify the value of the sum $\sum_a \tilde{Z}[a, b]$. As a consequence, it does not change the fact that - as for countermeasure 1 - this sum depends on $X^{2+s}[b]$ and hence on s.

The previous attack against Countermeasure 1 can hence be still applied and its efficiency is the same.

Randomize the Two Loops in LIM. In this case, two random permutations α and β defined over $[0;t]$ are generated and we have:

$$\tilde{Z}[a,b] = X^{2+s}[\alpha(a)] \cdot X^{2+s}[\beta(b)] \ . \tag{10}$$

To attack this countermeasure, a solution consists in performing the same attack as against the first and the second countermeasure exhaustively for all the possible permutations β. This attack stays efficient as long as t is reasonably small (lower than 16).

The first attack presented in this section shows that protection strategies based on masking are not promising. A possibility could be to use a different pair of masks for each internal multiplication but the cost of such a countermeasure would be prohibitive. The randomization of operations order seems to be more interesting but the attacks exhibited in this section point out that they must be carefully specified. In the following section we propose such a specification where the operations order randomization is done globally over all the indices.

6 New Countermeasure

Here, we propose to randomize the two loops (over a and b) in the LIM *simultaneously*. For such a purpose, we first need to efficiently[8] generate a random permutation over $\{(a,b); a,b \in [0;t]\}$. To achieve good efficiency, we propose to generate such a random permutation using Algorithm 1, following the same idea as in [7]. Even if there is no formal proof that this method enables to generate random permutations that are indistinguishable from perfectly random ones, we are confident about this in practice. The number of permutations that can be generated thanks to Algorithm 1 is around $(t+1)^{2\ell}$ where ℓ is the number of random values in the input of Algorithm 1. This number is sufficiently high to prevent attacks involving exhaustive search.

Algorithm 1. Generation of Random Permutation (GRP)

```
**Input**: Two integers $t$ and $\ell$, a permutation $\alpha_0$ over $[0,(t+1)^2-1]$.
**Output**: A vector in $[0,(t+1)^2-1]$ (elements are represented in base $t+1$).
$(r_0, r_1, \ldots, r_{\ell-1}) \leftarrow$ random elements in $\mathbb{Z}_{(t+1)^2}$
for $i$ from 0 to $\ell-1$ do
    for $j$ from 0 to $(t+1)^2-1$ do
        $\alpha_{i+1}[j] \leftarrow \alpha_0[(\alpha_i[j] + r_i) \bmod (t+1)^2]$
return $\alpha_\ell$
```

[8] This step may be very costly even for small values of the parameter t [12].

The random permutation returned by Algorithm 1 can be used to randomize the manipulation of the words $X[a]$ and $Y[b]$ simultaneaously. A second random permutation P over the set of integers $1, 2, \ldots, 2t+1$ must be used to avoid attacks in the carry propagation treatment. Eventually, we get the proposal of Algorithm 2 leading to a secure Long Integer Multiplication algorithm, where the carry registers $C[h]$ must be of bit-length $\omega + \log_2(t+1)$.

Algorithm 2. Long Integer Multiplication with randomization of the two loops together.

```
Input: $X = (X[t], X[t-1], \ldots, X[0])_{2^\omega}$, $Y = (Y[t], Y[t-1], \ldots, Y[0])_{2^\omega}$, $p$.
Output: $\mathrm{LIM}(X, Y)$.
$\alpha_\ell = (\alpha, \beta) \leftarrow \mathrm{GRP}(t, p, \alpha_0)$
$P \leftarrow$ random permutation of $1, 2, \ldots, 2t+1$.
for $a$ from 0 to $2t+1$ do
    $R[a] = C[a] = 0$
for $h$ from 0 to $(t+1)^2 - 1$ do
    $a \leftarrow \alpha[h]; b \leftarrow \beta[h]$
    $(U, V)_{2^\omega} \leftarrow R[a+b] + X[a] \cdot Y[b]$
    $R[a+b] \leftarrow V$
    $C[a+b+1] \leftarrow C[a+b+1] + U$
for $i$ from 1 to $2t+1$ do
    for $j$ from 1 to $2t+1$ do
        $s \leftarrow P[j]$
        if $s \geq i$ then
            $(U, V)_{2^\omega} \leftarrow R[s] + C[s]$
            $R[s] \leftarrow V$
            $C[s+1] \leftarrow C[s+1] + U$
            $C[s] \leftarrow 0$
return $R$
```

References

1. Batina, L., Gierlichs, B., Prouff, E., Rivain, M., Standaert, F.-X., Veyrat-Charvillon, N.: Mutual Information Analysis: a Comprehensive Study. J. Cryptology 24(2), 269–291 (2011)
2. Bauer, A., Jaulmes, É., Prouff, E., Wild, J.: Horizontal and Vertical Side-Channel Attacks Against Secure RSA Implementations – Extended Version. To appear on the Cryptology ePrint Archive (2013)
3. 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)
4. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)

5. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal Correlation Analysis on Exponentiation. In: Soriano, M., Qing, S., López, J. (eds.) ICICS 2010. LNCS, vol. 6476, pp. 46–61. Springer, Heidelberg (2010)
6. Coron, J.-S.: Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 725–725. Springer, Heidelberg (1999)
7. Coron, J.-S.: A New DPA Countermeasure Based on Permutation Tables. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 278–292. Springer, Heidelberg (2008)
8. 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)
9. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
10. Goubin, L., Patarin, J.: DES and Differential Power Analysis – The Duplication Method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
11. Joye, M., Yen, S.-M.: The Montgomery Powering Ladder. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 291–302. Springer, Heidelberg (2003)
12. Knuth, D.E.: The Art of Computer Programming, Volume III: Sorting and Searching. Addison-Wesley (1973)
13. Koc, C.K.: Cryptographic Engineering. Springer (2008)
14. 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)
15. Kocher, P.C., Jaffe, J., Jun, B., Rohatgi, P.: Introduction to Differential Power Analysis. J. Cryptographic Engineering 1(1), 5–27 (1998)
16. Messerges, T.S.: Using Second-Order Power Analysis to Attack DPA Resistant Software. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 238–251. Springer, Heidelberg (2000)
17. Prouff, E., Rivain, M., Bevan, R.: Statistical Analysis of Second-Order Differential Power Analysis. IEEE Trans. Computers 58(6), 799–811 (2009)
18. Quisquater, J.-J., Samylde, D.: A new Tool for non-Intrusive Analysis of Smart Cards based on Electro-Magnetic Emissions, the SEMA and DEMA Methods. Presented at the rump Session of Eurocrypt (2000)
19. Walter, C.D.: Sliding Windows Succumbs to Big Mac Attack. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 286–299. Springer, Heidelberg (2001)
20. Yen, S.-M., Joye, M.: Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis. IEEE Transactions on Computers 49(9), 967–970 (2000)

A Long Integer Multiplication

Let $X = (X[t], X[t-1], \dots, X[0])_{2^\omega}$ denote the decomposition of an integer X in ω-bit words. The Long Integer Multiplication algorithm is:

Algorithm 3. Long Integer Multiplication

```
Input: $X = (X[t], X[t-1], \dots, X[0])_{2^\omega}$, $Y = (Y[t], Y[t-1], \dots, Y[0])_{2^\omega}$.
Output: $\mathrm{LIM}(X, Y)$.
for $a$ from 0 to $2t + 1$ do
    $R[a] \leftarrow 0$
for $a$ from 0 to $t$ do
    $C \leftarrow 0$
    for $b$ from 0 to $t$ do
        $(U, V)_{2^\omega} \leftarrow Z[a, b] = X[a] \cdot Y[b]$
        $(U, V)_{2^\omega} \leftarrow (U, V)_{2^\omega} + C$
        $(U, V)_{2^\omega} \leftarrow (U, V)_{2^\omega} + R[a + b]$
        $R[a + b] \leftarrow V$
        $C \leftarrow U$
    $R[a + t + 1] \leftarrow C$
return $R$
```

Timing Attack against Protected RSA-CRT Implementation Used in PolarSSL

Cyril Arnaud[1] and Pierre-Alain Fouque[2]

[1] École de l'Air
cy.arnaud@orange.fr
[2] Université Rennes 1
pierre-alain.fouque@ens.fr

Abstract. In this paper, we present a timing attack against the RSA-CRT algorithm used in the current version 1.1.4 of PolarSSL, an open-source cryptographic library for embedded systems. This implementation uses a classical countermeasure to avoid two previous attacks of Schindler and another one due to Boneh and Brumley. However, a careful analysis reveals a bias in the implementation of Montgomery multiplication. We theoretically analyse the distribution of output values for Montgomery multiplication when the output is greater than the Montgomery constant, R. In this case, we show that an extra bit is set in the top most significant word of the output and a time variance can be observed. Then we present some proofs with reasonable assumptions to explain this bias due to an extra bit. Moreover, we show it can be used to mount an attack that reveals the factorisation. We also study another countermeasure and show its resistance against attacked library.

Keywords: Side-Channel Attack, Timing Attack, Montgomery Multiplication, Practical Attack, PolarSSL.

1 Introduction

The implementation of light-weight open-source cryptographic libraries is an important security issue. In this paper, we study timing attacks on the current version, 1.1.4, of the open source library PolarSSL [1] which is a well-known library, widely used in embedded systems. We look at the security of the exponentiation used in PolarSSL which is similar to the one used in OpenSSL [2] but presents some differences.

There are mainly three timing attacks [[3,4,5], on the RSA with Chinese Remainder Theorem (CRT) using Montgomery multiplication (MM). In the latter algorithm, a conditional branch leaks some information about the modulus and if RSA-CRT is used, the modulus is a prime factor of the RSA modulus N. These attacks were performed without blinding and can be applied to the version of Montgomery multiplication which contains a final conditional subtraction, called extra-reduction, for reducing the output. A classical and efficient countermeasure consists in using dummy operations to make the time computation constant in the Montgomery implementation. Such countermeasure is used by PolarSSL while OpenSSL prefers to use the blinding technique.

In this work, we study the dummy operation countermeasure used to avoid these three attacks and we propose an efficient practical chosen-ciphertext attack on this countermeasure used in PolarSSL. We show that even though the extra-reduction is not visible, since the dummy operations mask them, when the ouput of Montgomery multiplication before any conditional subtraction is greater than R, an *extra-bit* is set and the computation takes more time when the output is less than R. In this paper, we study the probability of an extra bit in Montgomery multiplication. When $q \simeq R$, we prove that the probability of an extra-bit for a squaring operation, for a multiplication with two random numbers and for a multiplication with a particular value $c \in \mathbb{Z}_q$ tends to $1/3$, $1/4$ and $c/2R$ respectively.

Our paper is organized as follows : §2 describes the different algorithms used to compute efficiently the exponentiation operations. We also recall how the previous attacks operate and the countermeasures that were proposed to defeat these attacks. In §3, we describe the specifications of the PolarSSL exponentiation implementations, we also show that some bias is still present in the time computation and we explain the probability of an extra bit. In §4, our attack details are addressed, and §5 presents our experimental results. We investigate possible countermeasures in §6. We conclude our paper in §7.

2 Background

2.1 Montgomery Multiplication

First proposed by Montgomery in [6], the MM algorithm provides an efficient method for computing modular multiplications (given q an odd integer and two integers $a, b \in \mathbb{Z}_q$, we compute $ab \ mod \ q$) and squaring ($a^2 \bmod q$).

In this subsection, we describe a generic multi-precision variant of MM. Long integers are represented as a sequence of words, the size of words is denoted by w. Let $r = 2^w$ and $s = \left\lceil \frac{|Q|}{w} \right\rceil$ represents the number of required words of size w. Large integers are written in basis r, and we note digits in lowercase. Thus, those numbers have the forms $A = \sum_{i=0}^{i=s-1} a_i r^i = (a_{s-1}, ..., a_0)_r$, $B = (b_{s-1}, ..., b_0)_r$, $Q = (q_{s-1}, ..., q_0)_r$, where $0 \leq a_i, b_i, q_i < r$. Let $R = r^s$ and $\mu_0 = -\frac{1}{q_0} \bmod r$. Big number library implement a w-bit multiplication used during the multiplication of a word with a large integer, which is denoted by $\otimes_w$. Multi-precision variant of MM is described in figure 1.

In order to use MM all variables must first be converted in Montgomery representation $\bar{A}(= AR \bmod Q)$ by computing MULTIMONTMUL$(A, R^2, Q) = \bar{A}$. At the end, the result of the algorithm can be converted to classical representation by performing MULTIMONTMUL$(\overline{A}, 1, Q) = A \bmod Q$.

In line 8 of MULTIMONTMUL, the conditional subtraction, $Z = Z - Q$, is called an *extra-reduction.* It is occasionally carried out to ensure that the result Z is in the range $[0, Q)$ and causes a timing difference. Timing attacks [3,4,5]

```
1: function MULTIMONTMUL($A, B, Q$)
2:     $Z = (z_s, ..., z_0)_r \leftarrow 0$
3:     for $i = 0$ to $s - 1$ do
4:         $u \leftarrow ((z_0 + a_i \times b_0) \times \mu_0) \bmod r$
5:         $Z \leftarrow (Z + a_i \otimes_w B)$
6:         $Z \leftarrow (Z + u \otimes_w Q)\ div\ r$
7:     if $Z \geq Q$ then
8:         $Z \leftarrow Z - Q$
9:     return $Z$ ($= ABR^{-1} \bmod Q$)
```

Fig. 1. MM Multi-precision variant. $A, B < Q$.

```
1: function MONTMUL($A, B, Q$)
2:     $Z = (z_{2s}, ..., z_0)_r \leftarrow 0$
3:     for $i = 0$ to $s - 1$ do
4:         $u \leftarrow (z_i + a_i \times b_0) \times \mu_0$
5:         $(z_{2s}, ..., z_i) \leftarrow ((z_{2s}, ..., z_i) + a_i \otimes_w B)$
6:         $(z_{2s}, ..., z_i) \leftarrow ((z_{2s}, ..., z_i) + u \otimes_w Q)$
7:         $z_i \leftarrow a_i$
8:     $G \leftarrow (z_{2s}, ..., z_s)$
9:     if $G \geq Q$ then
10:        SUB($G, Q, s$)                ▷ Extra-reduction
11:    else
12:        SUB($Z, G, s$)                ▷ defence
13:    return $G$ ($= ABR^{-1} \bmod Q$)
```

Fig. 2. POLARSSL's Montgomery Multiplication multi-precision with countermeasure. $A, B < Q$.

detect the timing difference according to the extra-reduction is executed or not. In fact, Schindler [3] showed that the probability of an extra-reduction occurs in MULTIMONTMUL($\bar{X}, B, W$), where B is uniformly distributed in $\mathbb{Z}_q$, is:

$$P\ (\text{extra-reduction in } \textsc{mont}(\bar{X}, B, Q)) = \frac{\bar{X} \bmod Q}{2R} \tag{1}$$

According to (1), when $\bar{X}$ rises and approaches a multiple of Q, the probability of an extra-reduction increases. At exact multiples of Q, the probability of an extra-reduction is null.

2.2 Modular Exponentiation Algorithm: Sliding Window

MM is particularly interesting when it is combined with the modular exponentiation algorithm to compute $m = c^d \bmod q$. OpenSSL[2] and PolarSSL[1] use an optimization of the square-&-multiply algorithm. This algorithm, called Sliding Windows Exponentiation (SWE), considers block of bits of the exponent rather than bits. The exponent d is split into windows of size *wsize* (depending on the size of d), where the windows are not always contiguous. Different ways are available for choosing windows. As shown in figure 3, SWE requires a precomputed table which is computed before the exponentiation process. During the modular exponentiation phase (computing $\bar{M}$), this table is used to process *wsize* bits of d at each iteration. In both phases of SWE (precomputation phase and exponentiation), the MM is used.

2.3 Decryption of RSA with Chinese Remainder Theorem

Let $N = PQ$ be a n-bit RSA modulus, where P and Q are prime numbers. The public key is denoted by (N, e) and the associated private key by (D, P, Q). RSA decryption consists in computing a modular exponentiation $M = C^D \bmod N$,

```
1: function EXPONENT($C, D, Q$)
2:     if $C \geq Q$ then
3:         $C \leftarrow C \bmod Q$
4:     $\bar{C}_1 \leftarrow$ MONT($C, R^2, Q$)
5:     Precomputing table phase
6:     Modular exponentiation phase
7:     $M \leftarrow$ MONT($\bar{M}, 1, Q$)
8:     return $M$ ($= C^D \bmod Q$)
```

Fig. 3. Sliding window exponentiation

```
1: function RSA-CRT($C, D, P, Q$)
2:     $C_p \leftarrow C^{D_p} \bmod P$
3:     $C_q \leftarrow C^{D_q} \bmod Q$
4:     $T \leftarrow (C_q - C_p)\pi \bmod Q$
5:     $M \leftarrow TP + C_p$
6:     return $M$ ($= C^D \bmod N$)
```

Fig. 4. RSA-CRT decryption. $D_p = D \bmod (P-1)$, $D_q = D \bmod (Q-1)$ and $\pi = P^{-1} \bmod Q$ are precomputed.

where C is the ciphertext to decrypt. A well-known optimization of this operation is the RSA-CRT which takes advantage of the decomposition in prime factor of N. Then, RSA-CRT reduces the computation time by a factor of about 75%. The RSA-CRT,with Garner's recombination, is shown in figure 4. The attacked implementations compute lines (2) and (3) using the SWE exponentiation algorithm.

General Idea of Timing Attacks on RSA-CRT. According to ([1]), during the modular exponentiation, if two chosen ciphertext X and Y are decrypted, when $\bar{X} < \bar{Y} < Q$ the total number of extra-reductions is greater than the case of $\bar{X} < Q < \bar{Y}$. So, by detecting time difference to perform the decryption of $\bar{X}$ and $\bar{Y}$ with RSA-CRT, we can reduce the search space of Q.

Overview of Known Timing Attacks on RSA-CRT. We focus on two attacks [5,4] against OpenSSL 0.9.7 without using any blinding countermeasure. OpenSSL implements four optimizations for RSA decryption : CRT, SWE, MM and two multiplication procedures, normal and Karatsuba's algorithm. OpenSSL performs Karatsuba's multiplication when multiplying two integers of the same number of words, otherwise uses normal routine. Karatsuba is faster than normal multiplication. The two attacks exploit the factorization of the RSA modulus by exploiting the time variance of RSA-CRT decryption in OpenSSL which depends on the number of extra-reductions in MM and the choice of multiplication procedure. The attacks perform a binary search to find the bits of Q bit-by-bit.

In [5], Boneh and Brumley show that the effect of extra-reductions and Karatsuba depends on the position of the bit being recovered. Thus, there is a prevailing parameter which has an influence on the time variance. Two ways are explored for recovering a bit of Q. In [5], timing attack exploits MM which are carried out during the modular exponentiation phase of SWE while [4] exploits those are carried out during the precomputation phase.

Boneh and Brumley [5] propose two countermeasures to make RSA-CRT decryption time independent on the input ciphertext. The first one is to use only one multiplication procedure. The other one is to carry out a dummy subtraction if an extra-reduction is not needed. The result of dummy operation is not used. This approach is also suggested by Schindler [3].

3 PolarSSL's Implementation of RSA-CRT Decryption

PolarSSL [1] is a light-weight open source cryptographic and SSL/TLS library written in C. PolarSSL is implemented for embedded systems and has also been ported to Windows and Linux (32 and 64 bit). The PolarSSL implementation of RSA decryption uses, as optimization, the CRT, SWE and MM with defence. We describe the last two algorithms below.

3.1 Montgomery Multiplication Multi-precision

PolarSSL implements only one procedure for computing the MM and accepts multiplication for $w = 8,\ 16,\ 32$, or 64. PolasSSL's MM multi-precision is given in figure 2, where function $\textsc{sub}(X, Y, s)$ returns $X \leftarrow X - Y$ for the s least significant words. In this case, the division of Z by r is performed by bumping a counter on Z. After s iterations, the $s+1$ most significant words of Z are equals to $ABR^{-1} \bmod Q$ and the s least significant words to A.

In the following, we assume that for fixed parameters Q and r the running times to perform lines 8 to 13 are identical for all inputs. It is worth noticing that if an extra-reduction is not needed, a dummy subtraction is carried out (line 12). Those two operations (lines 10 and 12), subtractions on s words, take the same time to be performed. Moreover, we check that compiler optimizations do not remove the conditional branch. In all cases, the study of the assembly code reveals that the dummy subtraction is still present for all compiler optimizations.

In this paper we perform a cryptanalysis based on the conditions under which G is greater than R before any conditional subtraction is performed.

3.2 Timing Variation in PolarSSL's Montgomery Multiplication Multi-precision

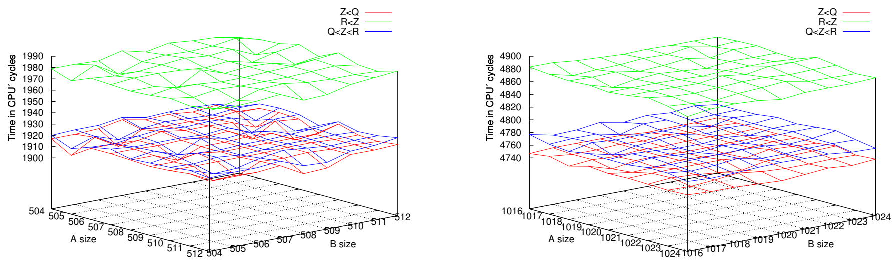
The dummy subtraction is used in $\textsc{montmul}(A, B, Q)$ to make the time required to perform the multiplication independent on the A and B operands. For more clarity, we study the behaviour of the generic Montgomery multiplication multi-precision figure 1.

Theorem 1. *[10] For inputs* $0 \leq A, B < Q$*,* $\textsc{multimontmul}$*(A,B,Q) returns the ouput* $Z \equiv ABR^{-1} \bmod Q$ *satisfying* $ABR^{-1} \leq Z < Q + ABR^{-1}$ *before any conditional subtraction.*

We suppose that $\frac{R}{2} < Q < R$. This assumption is true for standard RSA key lengths, such as 512, 1024 or 2048 bits, but also when these lengths are a multiple of world size. In this case, theorem 1 implied that $Z < 2R$. Moreover, if the value of Z is greater than R then the value of the top most word of Z is 1, i.e. $z_s = 1$, this bit is called extra bit.

In the source code of the multiplier, a while loop propagates a carry until no further carry is generated. Then, if the output Z is greater than R, during the s^{th} computation of line 6 of figure 1, the while loop is used to carry propagation up to the top most word of Z. Thus, a timing difference, whether the extra bit is carried out or not, could allow an attacker to mount a timing attack.

We performed experiments to show if an attacker could observe a timing difference in MONTMUL. We generated two random numbers A, B with known size, converted in Montgomery representation, and a prime number Q where $\frac{R}{2} < Q < R$. We sort the time in CPU's clock ticks to perform MONTMUL(A, B, Q) according to the size numbers and if an extra bit, an extra-reduction without extra bit or neither of them is carried out. The delay observed, Figure 5, to carry out Z before any conditional subtraction confirmed explanation above about timing difference. For the whole of MONTMUL(A, B, Q) we observed the same delay between curves. Thus, extra-reduction, if $Q \leq Z < R$, was actually masked by the dummy subtraction. However, the delay was smaller than the bias observed in previous attacks and it was proportional to the bitsize of Q, noted $|Q|$. In addition, compiler optimisation did not affect delay between curves.



(a) $|Q| = 512$ **(b)** $|Q| = 1024$

Fig. 5. MONTMUL(A, B, Q) without any conditional subtraction ($w = 64$ bits)

3.3 The Probability of an Extra Bit

Here we study the distribution of extra bit in MM algorithm inside the modular exponentiation algorithm. We investigate the distribution for some cases and we show that the probability for an extra bit is different for a squaring operation (P_{square}), for a multiplication with two random A and B (P_{mul}) and for a multiplication with a particular value $C \in \mathbb{Z}_Q$ (P_C).

First of all, to establish the probability distribution for the MM output Z after s iterations (figure 1), three reasonable assumptions are made :

1. Q is large and so we can switch from discrete to continuous method,

2. Colin D. Walter [11] showed that during an exponentiation, inputs to MM are uniformly distributed mod Q and independent,
3. for inputs A and B to MM during an exponentiation, the output Z before the conditional subtraction is uniformly distributed over the interval $[ABR^{-1}, Q + ABR^{-1})$ [11].

Lemma 1. *During the exponentiation, for input $0 \leq A, B < Q$, the probability of an extra bit is not null if and only if* $Q > \frac{\sqrt{5}-1}{2} \times R$.

Proof (of lemma 1). According to assumptions (2) and theorem 1, we obtain $ABR^{-1} \leq Z < Q + Q^2R^{-1}$. An extra bit is set if and only if $Z \geq R$ after s iterations. Then, if an extra bit is set, $R \leq Z < Q + Q^2R^{-1}$. This inequality is true if and only if $Q^2R^{-1} + Q - R > 0$, solving for Q we obtain $Q > \frac{\sqrt{5}-1}{2} \times R$.

□

Lemma 2. *During the exponentiation, for input $0 \leq A, B < Q$ and $C \in \mathbb{Z}_Q$ fixed, the probability of an extra bit is :*

1. $P_{mul} = \begin{cases} \frac{Q}{4R} + \frac{(R-Q)^2R}{Q^3} \times (\frac{3}{4} + \frac{1}{2}\log(\frac{Q^2}{(R-Q)R})) - \frac{(R-Q)}{R} & \text{if } Q > \frac{\sqrt{5}-1}{2} \times R \\ 0 & \text{otherwise} \end{cases}$

2. $P_{square} = \begin{cases} \frac{Q}{3R} + \frac{2(R-Q)\sqrt{(R-Q)R}}{3Q^2} - \frac{(R-Q)}{R} & \text{if } Q > \frac{\sqrt{5}-1}{2} \times R \\ 0 & \text{otherwise} \end{cases}$

3. $P_C = \begin{cases} \frac{C}{2R} + \frac{(R-Q)^2R}{2CQ^2} - \frac{(R-Q)}{R} & \text{if } C > \frac{(R-Q)R}{Q} \text{ and } Q > \frac{\sqrt{5}-1}{2} \times R \\ 0 & \text{otherwise} \end{cases}$

The proof (of lemma 2) is in appendix A.

If $Q \to R$, then the probability of an extra bit for a squaring, for a multiplication with random numbers and for a multiplication with a particular value in MONTMUL tends to $\frac{1}{3}$, $\frac{1}{4}$ and $\frac{C}{2R}$ respectively. On the other hand, for $Q \to \frac{\sqrt{5}-1}{2} \times R$, these probabilities tend to 0.

Corollary 1. *For $Q > \frac{\sqrt{5}-1}{2} \times R$ and $X, Y \in \left(\frac{(R-Q)R}{Q}, Q\right)$, if $X > Y$ then $P_X > P_Y$* .

Proof (of corollary 1). Let f a function denoted by :

$$\begin{aligned} f : \left(\frac{(R-Q)R}{Q}, Q\right) &\to (0,1) \\ C &\mapsto \frac{C}{2R} + \frac{(R-Q)^2R}{2CQ^2} - \frac{(R-Q)}{R} \end{aligned}$$

$\forall\, C > \frac{(R-Q)R}{Q}$, $f'(C) = \frac{1}{2R} - \frac{1}{C^2} \times \frac{(R-Q)^2R}{2Q^2} > 0$.

Then f is strictly increased in interval $\left(\frac{(R-Q)R}{Q}, Q\right)$. Thus if X, Y within this interval with $X > Y$ then $P_X > P_Y$.

□

3.4 Sliding Window Exponentiation

In this subsection we treat a variant of SWE, a fixed window exponentiation (FIXEDEXPONENT) which is used in PolarSSL. It differs from the SWE in OpenSSL. In PolarSSL, the precomputation table phase computes $\bar{C}_1$ which is the ciphertext in Montgomery representation, $\bar{C}_1^{2^{wsize-1}}$ with successive $wsize-1$ squares of $\bar{C}_1$, and stores in the table $\bar{C}_i = \text{MONTMUL}(\bar{C}_1, \bar{C}_{i-1}, Q)$, for $i = (2^{wsize-1}+1)$ to $(2^{wsize}-1)$. In the modular exponentiation, a block of bits (only one or $wsize$) of D are processed at each iteration. The secret exponent D is split into windows of fixed size $wsize$ (depending on the size of D) where the most significant bit is 1. We denote by w_d the value of the window. This window is used to carry out $\bar{M} = \text{MONTMUL}(\bar{M}, \bar{C}_{w_d}, Q)$ during an iteration of the modular exponentiation phase.

For fixed Q and D and according to lemma 2, the number of extra bits in modular exponentiation phase of FIXEDEXPONENT(C, D, Q) is constant and independent of ciphertext input C. However, using lemma 2, numbers of extra bits depend on $\bar{C}_1$ which is equal to $CR \bmod Q$. According to corollary 1, the secret modulus Q can be found by using a chosen ciphertext.

It is worth noticing that Schindler's attack fails while Boneh and Brumley timing attack [5] should work against PorlarSSL's RSA-CRT.

4 A Timing Attack on PolarSSL

In this section, we will suppose the size of RSA modulus is equal to 512, 1024, or 2048 bits. The precomputing table phase of SWE in modulo Q requires $2^{wsize-1}-1$ (from 1 to 32) MM with $\bar{C}_1$, i.e. ciphertext in Montgomery representation. Therefore, we exploit these operations in our attack. Because of the bias in PolarSSL's Montgomery multiplication is very small, we also use efficient statistical hypothesis tests such as F-test and T-test in our attack model.

4.1 Two-Sample Hypothesis Testing

Two-sample hypothesis testing is a method estimating two independent samples parameters which are extracted from two populations.

An F-test is used to test equal variance of the two populations. The value generated by F-test, F_{observed}, is used to verify the timing sampling correctness. In some case, we invalidate some erroneous measurements due to the noise of other processes running on the machine. The F-test is :

$$F_{\text{observed}} = \frac{\frac{n_1}{n_1-1}{s_1}^2}{\frac{n_2}{n_2-1}{s_2}^2}, \tag{2}$$

where n_1, n_2 are sample sizes and ${s_1}^2$, ${s_2}^2$ are the sample variances. Let F_α is the critical value of the F-distribution with $(n_1-1,\, n_2-1)$ degrees of freedom and a significance level α. If $F_{\text{observed}} > F_\alpha$ then variances of the population are different.

In two-sample t-test, we compare two independent sample means from two populations with a same variance. The value generated by T-test, T_{observed}, is used in the decision strategy for recovering bit. The two-sample t-test is :

$$T_{\text{observed}} = \frac{(\bar{x}_1 - \bar{x}_2) - d_0}{S_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}, \tag{3}$$
$$\text{where } {S_p}^2 = \frac{(n_1 - 1){s_1}^2 + (n_2 - 1){s_2}^2}{n_1 + n_2 - 2},$$

where $\bar{x}_1$ and $\bar{x}_2$ are the sample means. Let t_α is the critical value of the Student's t distribution with $(n_1 + n_2 - 2)$ degrees of freedom and a significance level α. If the means of the two samples are right, T_{observed} is less than T_α, otherwise greater than the critical value. Furthermore, we define a parameter T_β to increase the correctness of guess.

4.2 Our Attack Method

Let N be an RSA modulus with $N = pq$, $q < p$, $|N| = n$ and $|q| = |p| = n/2$. To factorize N, it is enough to recover the half most significant bits of p or q, since Coppersmith's [7] algorithm allows one to recover the complete factorisation of N. However, our method allows us to recover around the $n/2 - \log_2(NS) - 1$ most significant bits of modulus, where the parameter NS is called the neighbourhood size and used to increase statically the bias (for more details about NS refer to [5]).Thus, our attack ensures the recovery of bits of p or q one at a time, from most significant to least. Let $q = (1, q_1, ..., q_{\frac{n}{2}-1})$ is the binary coded of q and assume that the attacker knows the k most significant bits of q. We recover q_k as follows :

- **Step 1 :** Generate g and g_h where $g = (1, q_1, , ..q_{k-1}, 0, \ldots, 0)$ and $g_h = (1, q_1, , ..q_{k-1}, 1, ..., 0)$. If $q_k = 1$ then $g < g_h < q$, otherwise $g < q < g_h$.
- **Step 2 :** For $j = 0$ to $NS - 1$ compute $u_j = (g + j)R^{-1} \bmod N$ and $u_{hj} = (g_h + j)R^{-1} \bmod N$. The parameter neighbourhood size, i.e. NS, depend on experiment parameters.
- **Step 3 :** For $j = 0$ to $NS - 1$ measure the time to decrypt both u_j and u_{hj} . Let $T_{u_j} = Time(RSA{-}CRT(u_j, d, p, q))$ and $T_{u_{hj}} = Time(RSA{-}CRT(u_{hj}, d, p, q))$. We obtain two groups.
- **Step 4 :** Take a time sampling of the two groups, noted ζ_u and ζ_{u_h} .
- **Step 5 :** Compute $F_{observed}$ of ζ_u and ζ_{u_h} using (2). If $F_{\text{observed}} > F_\alpha$, we assume that timing samples are invalidated. So, we replay step 3.
- **Step 6 :** Find the largest interval `t` in ζ_u and ζ_{u_h} where $F_{observed} \approx 1$.
- **Step 7 :** Compute $T_{observed}$ of ζ_u and ζ_{u_h}, using 3, in interval `t`. If $T_{\text{observed}} < T_\alpha$, the attacker assumes that $q_k = 1$. Otherwise, if $T_{\text{observed}} > T_\beta$, we fixed empirically $T_\beta = 10$, then $q_k = 0$. If $T_\alpha < T_{\text{observed}} < T_\beta$, we assume that timing samples are invalidated and we replay step 3.

Unlike Boneh and Brumley [5], our attack does not need a particular phase to determine the first few bits: the attack begins with $g = (1, 0, ..., 0)$ and recovers p or q.

An optimization of step 3 was to carry out only one of the two samples ζ_u and ζ_{u_h}. Indeed, one of sample of the previous recovering bit could be reused during the recovering process. If we guessed that $q_k = 0$ (resp. $q_k = 1$) then we reused ζ_u (resp. ζ_{u_h}) during the decision process of the next bit. Thus, the number of chosen ciphertext was divided by two. However, we carried out the two samples ζ_u and ζ_{u_h} when these timing samples are invalidated.

5 Experimental Results

We performed our attack against the latest version 1.1.4 of PolarSSL with countermeasure (dummy subtraction). All of the experiments presented were run under Ubuntu 12.04 LTS 64 bits on an Intel Core i7. We compiled, using GCC 4.6.3, PolarSSL with its default value : `-D_FILE_OFFSET_BITS=64 -O`. We generated randomly all keys with PolarSSL's key generation routine. To get an accurate time measurement of RSA-CRT decryption, we used Time Stamp Counter (TSC) and Performance Monitor Counters (PMC).

5.1 Time Measurement

A well known method for measuring time is the TSC[8], a 64-bit register, which counts front side bus ticks and multiplies it by the CPU's frequency. The TSC is available on Intel CPU since introduction of the Pentium. This counter is read using the `rdtsc` assembly instruction which is not a privilege operation.

Our experiments were performed on a multi-core platform which compromised the use of the TSC registers. Thus, we implemented another way too.

PMC[8] provides the capability to monitor performance events and measure ticks for each core. Every PMC has a number which allows it to be referenced. PMC are supported by Model Specific Register (MSR). MSR are specific to a particular CPU which store data and set information for the CPU. PMC can be read using `rdmsr` while `wrmsr` writes to an MSR. We used these instructions to store ticks values for a chosen core. These instructions must be executed using privilege ring 0. Then, we need to use a kernel driver.

5.2 Experimental Results

In this subsection, we present the result of four experiments with the three sizes of modulus : 512, 1024 and 2048 bits. In each case, we showed that recovering half of the most significant bits of a prime factor is possible with a success probability around 100%. The first two experiments were performed in the same computing process. We measured directly the time to perform RSA-CRT for a chosen ciphertext. These experiments ensured we check the effectiveness of our statistical choices and the accuracy of using PMC. The last two experiments were obtained in inter-process in nature, and via TCP socket. We implemented a simple TCP server and client with the two processes ran on the same machine (server and client). The TCP server read a binary string sent by the client which is in PolarSSL's multi-precision representation. When it completed the decryption of the RSA-CRT, it returned 0 to the client. The TCP client measured the time between sending a message and receiving a response.

We attacked several random keys to determinate the efficiency of our method. The size of neighbourhoods was gained empirically. We denote by $\Delta_0 = mean(\zeta_u) - mean(\zeta_{u_h})$ when q_i is 0 and Δ_1 when it is 1. Due to Karatsuba's multiplication, in [5,4] the decryption time during RSA-CRT is variable (depending on the weight of recovering bit) when in PolarSSL is quite stable.

These different experiments demonstrated that our statistical tests are reliable and suitable for timing attack. Moreover, measuring time with PMC made our attack more efficient.

Experiment 1 - Same Process with `RTDSC` Instruction. In this experiment, we measured the time to perform RSA-CRT with `RDTSC` instruction. Table 1 shows when the key size increases, the ratio of replay becomes higher. The decryption time computing rises with the key size because of noise since other processes penalize our attack.

Table 1 also shows that the Δ_0 for $q_i = 0$ depends on the modulus size. The delay between Montgomery multiplication with and without extra bit is increased by a factor of 2 for q's size 512 and 1024 bits. When we perform RSA-CRT with a 1024-bit (resp. 2048-bit) modulus, the size of the window in SWE is 5 (resp. 6). Then, the precomputation table phase of SWE requires 15 (resp 31) MM. In the experiments, a factor of four between Δ_0 for 1024 and 2048 bits modulus is expected.

Table 1. Results of our attack with `RDTSC`. We measure timing computation in the same process.

Modulus size	NS	ratio of replay	Δ_0	Δ_1	Number of query
512 bits	600	2%	2614	-90	78600
1024 bits	800	18%	5553	-134	241600
2048 bits	1000	50%	21855	-1743	768000

Experiment 2 - Same Process with `RDMSR` Instruction. In this experiment we show, in table 2, that for the same size of neighbourhood, the noise due to other processes running in the computer do not interfere too much with our attack. When we use PMC for measuring ticks, we read time for our process core. Thus, processes running on other cores do not penalize the timing attack.

Table 2. Results of our attack with the `RDMSR`. We measure timing computation in the same process.

Modulus size	NS	ratio of replay	Δ_0	Δ_1	Number of query
512 bits	600	2%	2452	67	78600
1024 bits	800	10%	4554	−216	225600
2048 bits	1000	15%	22434	−992	589000

Experiment 3 - Inter-Process with `RTDSC` Instruction. Table 3 shows that communication via inter-process does not reduce the effectiveness of our attack. The noise from inter-process is eliminated by increasing the size of neighbourhood, given similar Δ_0 and ratio of replay.

Table 3. Results of our attack with `RDTSC`. We measure timing computation in inter-process via TCP.

Modulus size	NS	ratio of replay	Δ_0	Δ_1	Number of query
512 bits	1000	2%	2537	-144	131000
1024 bits	1100	21%	5557	-1489	341000
2048 bits	1200	55%	28157	1042	952800

Experiment 4 - Inter-Process with `RDMSR` Instruction. In this experiment we use processor affinity which allows us to choose the core/CPU where a process is running. Thus, the TCP client and server are carried out on different cores. Table 4 shows that the ratio of replay in inter-process attack is very low when we used `RDMSR` instruction.

Table 4. Results of our attack with `RDMSR`. We measure timing computation in inter-process via TCP.

Modulus size	NS	ratio of replay	Δ_0	Δ_1	Number of query
512 bits	1000	0%	2217	-1	128000
1024 bits	1100	5%	5067	-675	295900
2048 bits	1200	10%	17033	872	675600

When we used `RDMSR` in inter-process we were able to recover a 1024 bits key with an average of 215200 queries. These results are obtained with 800 neighbourhoods and a ratio of replay is around 5%.

5.3 Network Attacks

Two ways are performed to measure timing decryption of RSA-CRT during inter-process attack. In [5], Boneh and Brumley show that it is enough to increase neighbourhood to convert an inter-process into a network timing attack. Thus using `RTDSC` instruction, we should be able to factorise N by measuring the time, from sending the ciphertext on the network and to receiving the response of the server.

Another way is also possible. Assume that an attacker is able to perform a spy process on the server, operating at a high level. The TCP client performs the ciphertext to be decrypted and sends the message. The spy sniffs TCP socket on the server and measure the time, using `RDMSR` instruction, taken by the server for answering. Once this is completed, the spy process sends the measurement to the client which is responsible for the decision of bit to recover. This scenario should have the same result as `RDMSR` in inter-process.

In the real world scenario the experimental data is limited. However, in PolarSSL's SSL implementation the key does not have timelife, only the SSL's session (lifetime is one day in PolarSSL's example). Our attack in inter-process with a 1024 key size takes about ten minutes. So, the attack presented in our paper seems to be feasible in the real world.

6 Defences

PolarSSL's library is vulnerable during our timing attack. In order to counteract it, we could make constant the time decryption of the RSA-CRT implementation. The attack results show that it is very complicated to obtain an implementation with those characteristics for any key size. In order to protect PolarSSL against this attack, three countermeasures can be developed in this section. The first one is used by OpenSSL and we suggested two others.

6.1 Blinding

This defence makes the time decryption of RSA-CRT independent on the input ciphertext. RSA-CRT blinding is implemented as follows.

Let a is a random value, e the RSA encryption exponent and c the ciphertext. To decrypt c :

- compute : $x = a^e.c \bmod N$,
- decrypt x : $RSA - CRT(x, d, p, q) = m' = a^{ed}c^d \bmod N$,
- compute : $\frac{m'}{a} \bmod N = a^{ed-1}c^d \bmod N = c^d \bmod N = m$.

Since a is random, then x is a random value. Then, the attacker can not choose the ciphertext being input to MM. This approach is preferred by Boneh and Brumley [5].

6.2 Alternatives to Blinding

Our timing attack exploits the behaviour of the MM when an extra bit is carried out, others, when an extra-reduction occurs. Two strategies are possible for cancelling out extra bit : the first one uses particular modulus size whereas the other needs to modify PolarSSL's key generation routine.

In the following, we suppose that attacked library implements a dummy subtraction, such as PolarSSL, used to mask the timing effect of an extra-reduction without extra bit.

Use Particular Modulus Size. Colin D. Walter [9] demonstrated that if we choose $s' > s$ such as $2Q < r^{s'-1}$, $s^{'} \geq s+2$, then MM does not need extra-reduction. For cancelling out extra-reduction, large integer needs to be represented with $s+2$ words which is quite inefficient. We could make our timing attack impracticable with particular modulus size.

Suppose that $|Q| = kw+1$. Then, $s = \lceil \frac{<Q>}{w} \rceil = k+1$ and $R = r^s = r^{k+1} = 2^{kw+w}$. We obtain :

$$2^{kw} < Q < 2^{kw+1} \text{ and } Q < \frac{1}{2^{w-1}} \times R \tag{4}$$

where $w \in \{8, 16, 32, 64\}$. According to lemma 1 extra-bit is cancelling out. Thus, our timing is defeated against's attacked library.

Countermeasure effectiveness is equivalent to blinding with a lower penalty. Penalty is 10% (resp. 6%) between 1026 and 1024 (resp. 2050 and 2048) modulus size .

Suppose that $|Q| = kw-1$, so $s = \lceil \frac{|Q|}{w} \rceil = k$. We obtain :

$$2^{kw-2} < Q < 2^{kw-1} \text{ and } \frac{1}{4} < \frac{Q}{R} < \frac{1}{2} \tag{5}$$

where $w \in \{8, 16, 32, 64\}$. According to lemma 1 extra-bit is cancelling out. Thus, our timing is defeated against attacked library.

Modify PolarSSL's Key Generation Routine. Another way to counteract timing attacks is to generate keys where primes factors are less than $\frac{\sqrt{5}-1}{2} \times R$. Then, according to lemma 1, extra-bit is cancelling out.

7 Conclusion

In this paper, we present a timing attack against PolarSSL - a protected SSL implementation of RSA-CRT. Our attack exploits an unknown arithmetical bias in Montgomery multiplication. In spite of this countermeasure, our experiments show that a timing attack is still possible using in inter-process for different modulus size. We also present a new way for measuring time decryption via performance monitor counters which improves the efficiency.

References

1. Bakker, P.: PolarSSL project. Version 1.1.4 (2012-05-31), http://polarssl.org/download_overview?download=1.1.4
2. Young, E.A., Hudson, T.J.: OpenSSL project. Version 0.9.7, http://openssl.org
3. Schindler, W.: A Timing Attack against RSA with the Chinese Remainder Theorem. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 109–124. Springer, Heidelberg (2000)
4. Aciiçmez, O., Schindler, W., Kooç, K.: Improving Brumley and Boneh timing attack on unprotected SSL implementation. In: Atluri, V., Meadows, C., Juels, A. (eds.) ACM Conference on Computer and Communication Security, pp. 139–146. ACM (2005)
5. Brumley, D., Boneh, D.: Remote timing attacks are practical. In: Proceedings of the 12th USENIX Security Symposium, pp. 1–14 (2003)
6. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computations 44, 519–521 (1995)
7. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. Journal of Cryptology 10, 233–260 (1997)
8. Intel. Intel 64 and IA-32 : Architectures Software Developer's Manual Combined Volumes 3A and 3B, System Programming Guide, Parts 1 and 2
9. Walter, C.D.: Montgomery Exponentiation Needs no Final Subtractions. Electronics Letters 35(21), 1831–1832 (1999)
10. Walter, C.D.: Precise Bounds for Montgomery Modular Multiplication and Some Potentially Insecure RSA Moduli. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 30–39. Springer, Heidelberg (2002)
11. Schindler, W., Walter, C.D.: More Detail for a Combined Timing and Power Attack against Implementations of RSA. In: Paterson, K.G. (ed.) Cryptography and Coding. LNCS, vol. 2898, pp. 245–263. Springer, Heidelberg (2003)

A Proof of Lemma 2

Proof. We assume that $Q > \frac{\sqrt{5}-1}{2} \times R$.
According to assumptions (1) and (3), we write P_{mul} as :

$$\begin{aligned} P_{mul} &= P \text{ (extra bit in } \textsc{montmul}(A, B, Q)) \\ &\simeq P(ABR^{-1} + Y > R) \\ &= \int_{\frac{(R-Q)R}{Q}}^{Q} \int_{\frac{(R-Q)R}{A}}^{Q} \int_{R-\frac{AB}{R}}^{Q} p(A, B, Y)\, \mathrm{dY}\, \mathrm{dB}\, \mathrm{dA} \ , \end{aligned}$$

where Y is unformly distributed on $\mathbb{Z}_Q$ and $p(A, B, Y)$ is the probability density function for $A \times B \times Y$. According to assumptions (1) and (3) A, B and Y are independently distributed mod Q. Thus, $p(A, B, Y) = p(A) \times p(B) \times p(Y)$. As noted above, A, B and Y are uniform on $[0, Q)$.

Thus, $p(A,B,Y) = \frac{1}{Q^3}$. Then

$$\begin{aligned}
P_{mul} &\simeq \frac{1}{Q^3}\int_{\frac{(R-Q)R}{Q}}^{Q}\int_{\frac{(R-Q)R}{A}}^{Q}\int_{R-\frac{AB}{R}}^{Q} \mathrm{dY}\,\mathrm{dB}\,\mathrm{dA} \\
&= \frac{1}{Q^3}\int_{\frac{(R-Q)R}{Q}}^{Q}\int_{\frac{(R-Q)R}{A}}^{Q} Q-R+\frac{AB}{R}\ \mathrm{dB}\,\mathrm{dA} \\
&= \frac{1}{Q^3}\int_{\frac{(R-Q)R}{Q}}^{Q} \frac{AQ^2}{2R}+\frac{1}{A}\times\frac{1}{2}(R-Q)^2R-Q(R-Q)\ \mathrm{dA} \\
&= \frac{Q}{4R}+\frac{(R-Q)^2R}{Q^3}\times\left(\frac{3}{4}+\frac{1}{2}\log\Big(\frac{Q^2}{(R-Q)R}\Big)\right)-\frac{(R-Q)}{R}.
\end{aligned}$$

In the same way, the probability of extra bit in a squaring operation is :

$$\begin{aligned}
P_{square} = P\ (\text{extra bit in } \textsc{montmul}(A,A,Q)) &\simeq P(A^2R^{-1}+Y>R) \\
&= \int_{\sqrt{(R-Q)R}}^{Q}\int_{R-\frac{A^2}{R}}^{Q} p(A,Y)\,\mathrm{dY}\,\mathrm{dA} \\
&= \frac{1}{Q^2}\int_{\sqrt{(R-Q)R}}^{Q} Q-R+\frac{A^2}{R}\,\mathrm{dA} \\
&= \frac{Q}{3R}+\frac{2(R-Q)\sqrt{(R-Q)R}}{3Q^2}-\frac{(R-Q)}{R}.
\end{aligned}$$

For a fixed C with $\frac{(R-Q)R}{Q} < C < Q$, the probability of extra bit is :

$$\begin{aligned}
P_C = P\ (\text{extra bit in } \textsc{montmul}(A,C,Q)) &\simeq P(CAR^{-1}+Y>R) \\
&= \int_{\frac{(R-Q)R}{C}}^{Q}\int_{R-\frac{CA}{R}}^{Q} p(A,Y)\,\mathrm{dY}\,\mathrm{dA} \\
&= \frac{1}{Q^2}\int_{\frac{(R-Q)R}{C}}^{Q} Q-R+\frac{CA}{R}\,\mathrm{dA} \\
&= \frac{C}{2R}+\frac{(R-Q)^2R}{2CQ^2}-\frac{(R-Q)}{R}
\end{aligned}$$

□

Fair Exchange of Short Signatures without Trusted Third Party

Philippe Camacho

Dept. of Computer Science, University of Chile,
Blanco Encalada 2120, 4to piso, Santiago, Chile
pcamacho@dcc.uchile.cl

Abstract. We propose a protocol to exchange Boneh-Boyen short signatures in a fair way and without relying on a trusted third party. Our protocol is quite practical and is the first of the sort to the best of our knowledge. Our construction uses a new non-interactive zero-knowledge (NIZK) argument to prove that a commitment is the encryption of a bit vector. We also design a NIZK argument to prove that a commitment to a bit vector $v = (b_1, b_2, ..., b_\kappa)$ is such that $\sum_{i \in [\kappa]} b_i 2^{i-1} = \theta$ where θ is the discrete logarithm of some public value $D = g^\theta$. These arguments may be of independent interest.

Keywords: Fair exchange, short signatures, gradual release of a secret.

1 Introduction

Nowadays it is more and more common to trade digital goods on the web: E-books, software licenses, avatar-games currencies like *Ultima Online*[1] to cite a few. Whether these goods are exchanged on *E-bay* through *Paypal* or bought directly to their provider *Amazon* or *Microsoft*, the transaction to be secure requires a trusted third party (TTP). Though it works quite well in practice, enabling totally distributed and at the same time secure transaction systems is of clear interest: It would avoid some security issues due to the presence of single points of failure, and also allow smoother electronic commercial transactions that would not rely on some intermediary. A lot of these transactions may be captured by the exchange of digital signatures. Suppose for example you want to buy a software license to some independent developer: Indeed exchanging the software license as well as the money transfer (digital check) can be modeled by signed messages. However we face a non-trivial problem. Given that the transaction is made on-line, a malicious participant may fool his counterpart by not sending his signature or sending some garbage information. A protocol that prevents such a behavior from a corrupted party is called *fair*: This means that at the end of the execution of protocol either both parties obtain the signature they expected or none does.

[1] http://en.wikipedia.org/wiki/Ultima_Online

There are two main approaches to solve this problem. On the one hand, one can assume that both players interact through a TTP. Though this solution does not fit our goal, it is important to note that an important line of research has focused on designing protocols where the TTP is only required when "something goes wrong". These protocols are said to be *optimistically fair*: See [1,25] and [22] for some recent work.

On the other hand, if no TTP exists and we assume that both participants have exactly the same computational resources, then it is impossible *in general* to achieve *complete fairness* [10]. In [2,14] was proposed a way to relax the notion of fairness in order to overcome Cleve's impossibility result. The idea is to assume that both players have roughly the same amount of time, so we can achieve *partial fairness*. Several secure multi-party computations and specific protocols, like [6,11,12,5,16], were built on top of this security notion. The recurrent idea behind these constructions consists in enabling each player to release their secret bit by bit in alternation. Thus, if a player aborts, the other participant will have "only one bit of disadvantage". Formalizing this idea is not an easy task though, in particular because it is hard to reason on the specific amount of time for the players. This issue was noticed in [19] where authors point out that (1) assigning more time to the honest party in order to allow him recover his value is somehow artificial as it does not depend on the participant himself, and (2) implementing such definitions seems to imply the use of strong assumptions related to the *exact* time required to solve some computational problem.

In this work we propose a new security definition that still captures the intuition of partial fairness for the exchange of digital signatures, but without forcing the participants to have access to almost equal computational resources as proposed in [16]. The idea of our definition is to compare the probabilities of computing valid signatures on the agreed messages at the end of the protocol. More precisely, if the adversary aborts the protocol, the honest participant[2] will compute the expected signature by choosing randomly a value from the space of signature candidates, which is defined by the remaining bits to be obtained. The adversary will keep running its own algorithm and also output a signature candidate. We say the protocol is secure if the probabilities that each participant output a valid signature only differ by a polynomial factor. Note that this definition, like previous ones that circumvent Cleve's impossibility result [10], allows the adversary to get some advantage, but it guarantees that this advantage is polynomially bounded. With that definition in hand we can prove the security of our protocol without having to rely on the strong assumptions mentioned above. Our protocol is designed to exchange short signatures [4] without the presence of a TTP. We use bilinear maps as the underlying signature scheme, and also the idea of releasing gradually each bit of some secret θ that will enable to recover the signature. The security of our construction relies on complexity assumptions for bilinear maps, namely the κ-Strong Diffie-Hellman [4], and the κ-Bilinear

[2] Note that we need to consider that at least one participant is honest, as otherwise we cannot really avoid that one of the two adversaries, which are arbitrary polynomial time algorithms, wins.

Diffie-Hellman assumptions [3] and holds in the common reference string model. As we use non-interactive zero-knowledge proofs of knowledge (ZKPoK) in order to make the protocol simpler and more efficient, we require the use of random oracle [15] or some non-black box assumptions [20]. If we like, we can use interactive ZKPoK at a minor expense of round efficiency.

Our Contributions

1. We propose a practical protocol for exchanging short signatures [4] without relying on a TTP. To the best of our knowledge this is the first construction that meets such a goal. The number of rounds of our protocol is $\kappa+1$, where κ is the security parameter. The communication complexity is $16\kappa^2 + 12\kappa$ bits. The protocol requires a linear number of group exponentiations, group multiplications, bilinear map applications, hash computations and also a constant number of group divisions.
2. We introduce a new non-interactive zero-knowledge (NIZK) argument to prove that a commitment is the encryption of a bit vector. This protocol may be of independent interest.
3. We introduce another NIZK argument to prove that a commitment to a bit vector corresponds to the binary decomposition of some value θ which is hidden as the discrete logarithm of some group element. We think this argument may lead to other interesting applications.
4. As stated earlier, we propose a new security definition for partial fairness in the context of the exchange of digital signatures. This definition is simple and avoids the issue of involving the *exact* running time of the participants.

Our Approach. Let $\kappa \in \mathbb{N}$ be the security parameter. Let $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMGen}(1^\kappa)$ be the public parameter where $p = |\mathbb{G}| = |\mathbb{G}_T|$ is prime, $\mathbb{G}, \mathbb{G}_T$ are cyclic groups, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is the bilinear map and g is a random generator. Let s be a random element in $\mathbb{Z}_p$, we consider the following common reference string: $(g, g^s, g^{s^2}, ..., g^{s^\kappa}) = (g_0, g_1, g_2, ..., g_\kappa)$. In practice this common reference string can be computed using generic multi-party computation techniques (see [9] for an efficient implementation) so that the secret s is randomly generated and remains unknown to all the participants. Another alternative is to rely on a TTP that would "securely delete" the secret after the generation of the common reference string. Obviously the intention of this work is to avoid the use of a TTP, but note however that even in this case, the TTP would be required only once.

Our construction can be summarized as follows. The prover chooses a secret $\theta \in \mathbb{Z}_p$, then commits each bit of this secret into a Pedersen [28] commitment, where the bit b_i in position i with randomness $r_i \in \mathbb{Z}_p$ will be committed with respect to the base (g, g_i): That is $\mathsf{Commit}(b_i, r_i, i) = g^{r_i} g_i^{b_i}$. Then we use a NIZK argument [3] to prove this commitment really encrypts a bit. The next step is to publish $D = g^\theta$ and show, using another NIZK argument,

[3] The reader can refer to the full version of this paper [7] for standard definitions related to NIZK protocols.

that θ, the discrete logarithm of D, is “equivalent” to the bit vector committed in $\boldsymbol{C} = (\mathsf{Commit}(b_i, r_i, i))_{i\in[\kappa]}$. More precisely, the argument proves that $\theta = \sum_{i\in[\kappa]} b_i 2^{i-1}$. Now if we consider some signature σ, the prover will blind it using θ to obtain $\tilde{\sigma} = \sigma^{\theta}$. Using bilinear maps it is straightforward to verify that $\tilde{\sigma}$ contains a valid signature σ which is blinded in the exponent by θ, the discrete logarithm of D. The other verifications will consist simply in checking the NIZK arguments. Finally, we need to provide zero-knowledge proofs of knowledge for the representation of each bit commitment in order to be able to simulate the execution of the protocol even if the adversary aborts. By releasing each bit in turn, both players will reconstruct their own blinding factor θ and obtain the signature.

Related Work. Among the abundant literature on the topic of gradual release and fair exchange for digital signatures, [12] is probably the work that is the most similar to ours: It describes a practical fair exchange protocol for digital signatures based on gradual release of a secret. The protocol described in [12] works for Rabin, RSA and El Gamal signatures. The number of rounds of the protocol described in [12] is roughly 2κ for RSA and Rabin signatures and κ for El Gamal signatures.

Due to Cleve’s impossibility result [10], the question of building complete fair protocols with dishonest majority seemed to be closed. However, Gordon et al. showed that non-trivial functions can be computed fairly in the two-party model [18], and left the question of finding a tight characterization of these functions open. In particular it is not known whether functions with a non-polynomial size domain and that return multiple bits as output (like computing a signature) can be computed fairly in Cleve’s setting.

In [19] is proposed a definition for partial fairness that may exhibit some similarities with ours (both definitions involve a $Q(\kappa)$ factor where Q is a polynomial). However our definition and approach differs quite from [19]. First, the setting in [19] is more general than our specific construction to exchange digital signatures. Secondly, in their protocol, the number of rounds is variable and defines the level of fairness, whereas in our construction fairness only depends on the computational power of the participants.

Our NIZK argument to prove that a commitment encrypts a bit vector is inspired by [21,20]. We remark that, though [16] uses the idea of gradual release, the construction proposed is not practical as it requires to code the functionality (signing in our case) as an arithmetic circuit.

Organization of the Paper. In Section 2 we introduce notations and recall some definitions and standard techniques we use in this work. In Section 3 we describe the bit vector commitment scheme. The argument for proving the equivalence between a bit vector commitment $(C_i)_{i\in[\kappa]}$ and the discrete logarithm θ of g^{θ} is introduced in Section 4. The fair exchange protocol is shown in Section 5. We conclude in Section 6.

2 Preliminaries

2.1 Notations

For $m, n \in \mathbb{N}$ with $m < n$, $[m..n]$ means the set of integers $\{m, m+1, ..., n-1, n\}$ and $[n]$ means the set of integers $\{1, ..., n\}$. If $\kappa \in \mathbb{N}$ is the security parameter then 1^κ denotes the unary string with κ ones. We will use p to denote a prime number of κ bits. A function $\nu : \mathbb{N} \rightarrow [0, 1]$ is said to be negligible in κ if for every polynomial $q(\cdot)$ there exists κ_0 such that $\forall \kappa > \kappa_0 : \nu(\kappa) < 1/q(\kappa)$. In the following, neg will denote *some* negligible function in κ. An algorithm is called PPT if it is probabilistic and runs in polynomial time in κ. We write $x \overset{R}{\leftarrow} X$ to denote an element x chosen uniformly at random from a set X. $x \leftarrow v$ means that the variable x is assigned the value v.

A vector of n components and values v_i is denoted $\boldsymbol{v} = (v_i)_{i\in[n]}$. If the vector contains elements of $\mathbb{Z}_p$ we may also write $B[\cdot] = (B[1], B[2], ..., B[n])$. Let $\theta \in \mathbb{Z}_p$, we denote by $\theta[\cdot]$ the binary decomposition (vector) of θ. That is $\theta[\cdot] = (\theta[1], ..., \theta[\kappa])$ and in particular $\theta = \sum_{i\in[\kappa]} \theta[i]2^{i-1}$. $P(\cdot)$ will stand for a formal polynomial with coefficients in $\mathbb{Z}_p$, and $P[\cdot]$ for the vector of its coefficients: Thus if $d = deg(P)$ is the degree of polynomial $P(\cdot)$ then we have: $P(X) = \sum_{i\in[d+1]} P[i]X^{i-1}$.

2.2 Non-interactive Zero-Knowledge Proofs of Knowledge

Our protocol for fair exchange uses zero-knowledge proofs of knowledge relative to bit commitments. In order to simplify the description of the fair exchange protocol we will use non-interactive zero-knowledge proofs of knowledge. We note however that interactive ZKPoK would work as well, though adding 2 rounds to our protocol and loosing possibly security guarantees in case the protocol is run in parallel or involves more than 2 players. The most popular way to implement such protocols is by using the Fiat-Shamir heuristic [15], trading non-interaction for a security proof relying on the random oracle model. We mention that our scheme could also be adapted to fit Groth's short non-interactive argument proof system [20]. In this case the security of non-interactive proofs of knowledge would depend on a non-black box assumption and we would get shorter arguments[4].

Let $\mathbb{G}$ be a cyclic group of prime order p where the discrete logarithm is hard. Let $\mathsf{H} : \mathbb{G} \rightarrow \mathbb{Z}_p$ be a randomly chosen function from a CRHF. Let g, h be two random generators of $\mathbb{G}$ such that the discrete logarithm of h in base g is unknown.

We will need a ZKPoK of the discrete logarithm θ of some public value $D = g^\theta$. Following the notation of [8], we have that $PK\{\theta : g^\theta\} = (c = \mathsf{H}(g^r), z = r - c\theta)$ where $r \overset{R}{\leftarrow} \mathbb{Z}_p$. The verifier checks that $c = \mathsf{H}(D^c g^z)$. We will also use the following ZKPoK that convinces a verifier that the prover knows the representation of a commitment $C = g^\alpha h^\beta$ in base (g, h) where $\alpha, \beta \in \mathbb{Z}_p$.

[4] Note however that the common reference string would need to be of quadratic size in the size of the statements.

$PK\{(\alpha,\beta) : C = g^{\alpha}h^{\beta}\} = (c = \mathsf{H}(g^{r_1}h^{r_2}), z_1 = r_1 - c\alpha, z_2 = r_2 - c\beta)$ where $r_1, r_2 \overset{R}{\leftarrow} \mathbb{Z}_p$. The verifier checks that $c = \mathsf{H}(C^c g^{z_1} h^{z_2})$.

2.3 Bilinear Maps

In this paper we consider bilinear maps which are defined as following:
Let $\mathbb{G}, \mathbb{G}_T$, be cyclic groups of prime order p. We consider a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ which is

- *bilinear*: $\forall a, b \in \mathbb{G}, x, y \in \mathbb{Z}_p : e(a^x, b^y) = e(a,b)^{xy}$.
- *non-degenerate*: let g be a generator of $\mathbb{G}$ then $e(g,g)$ also generates $\mathbb{G}_T$.
- *efficiently computable*: There exists a polynomial time algorithm BMGen with parameter 1^{κ} that outputs $(p, \hat{\mathbb{G}}, \hat{\mathbb{G}_T}, \hat{e}, g)$ where $\hat{\mathbb{G}}, \hat{\mathbb{G}_T}$ is the representation of the corresponding groups of size p (p being a prime number of κ bits), g is a generator of $\mathbb{G}$, and $\hat{e}$ is an efficient algorithm to compute the map. For the sake of simplicity, we will not distinguish between $\mathbb{G}, \mathbb{G}_T, e$, and $\hat{\mathbb{G}}, \hat{\mathbb{G}_T}, \hat{e}$.

2.4 Assumptions

Let $N \in \mathbb{N}$. For the following assumptions, the common public parameter is $\mathsf{PP} =< (p, \mathbb{G}, \mathbb{G}_T, e, g), (g_0, g_1, g_2, \cdots, g_N) >$ where s is chosen randomly in $\mathbb{Z}_p$ and $g_i = g^{s^i}$ for $i \in [0..N]$.

Definition 1. ***N-Diffie-Hellman Inversion (N-DHI) assumption,*** *[26]. The N-Diffie-Hellman Inversion problem consists in computing $g^{\frac{1}{s}}$ given PP. We say the N-DHI assumption holds if for any PPT adversary $\mathcal{A}$ we have*

$$Adv^{N\text{-}DHI}(\mathcal{A}, \kappa, N) = \Pr\left[g^{\frac{1}{s}} \leftarrow \mathcal{A}(1^{\kappa}, \mathsf{PP})\right] = \mathsf{neg}(\kappa)$$

The bilinear variant of the previous assumption was introduced in [3].

Definition 2. ***N-Bilinear Diffie-Hellman Inversion assumption (N-BDHI).*** *The N- Bilinear Diffie-Hellman Inversion problem consists in computing $e(g,g)^{\frac{1}{s}}$ given PP. We say the N-BDHI assumption holds if for any PPT adversary $\mathcal{A}$ we have*

$$Adv^{N\text{-}BDHI}(\mathcal{A}, \kappa, N) = \Pr\left[e(g,g)^{\frac{1}{s}} \leftarrow \mathcal{A}(1^{\kappa}, \mathsf{PP})\right] = \mathsf{neg}(\kappa)$$

Definition 3. ***N-Strong Diffie-Hellman assumption (N-SDH),*** *[4]. The N-Strong Diffie-Hellman (N-SDH) problem consists in computing $(c, g^{\frac{1}{s+c}})$ given PP. We say the N-SDH assumption holds if for any PPT adversary $\mathcal{A}$ we have*

$$Adv^{N\text{-}SDH}(\mathcal{A}, \kappa, N) = \Pr\left[(c, g^{\frac{1}{s+c}}) \leftarrow \mathcal{A}(1^{\kappa}, \mathsf{PP})\right] = \mathsf{neg}(\kappa)$$

As mentioned in [4], the N-SDH assumption is equivalent to the N-DHI assumption when c is fixed. The following assumption can be considered as a particular case of the *poly*-Diffie-Hellman assumption [23], or a generalization of the N+1-Exponent assumption introduced in [30].

Definition 4. ***N+i-Diffie-Hellman Exponent(N+i-DHE) assumption.*** *The N+i-Diffie-Hellman Exponent problem consists in computing $g^{s^{N+i}}$, for $1 \leq i \leq N$ given* PP. *We say the N+i-DHE assumption holds if for any PPT adversary $\mathcal{A}$ we have*

$$Adv^{N+i\text{-}DHE}(\mathcal{A}, \kappa, N) = \Pr\left[g^{s^{N+i}} \leftarrow \mathcal{A}(1^\kappa, \mathtt{PP})\right] = \mathsf{neg}(\kappa)$$

In [30], the N-DHI assumption was shown to be equivalent to the N+1-Exponent assumption (N+1-DHE). We state [5] here the following implication.

Proposition 1. *N-BDHI $\Rightarrow$ N+i-DHE.*

2.5 Digital Signatures

Standard Digital Signatures. We denote by $\mathtt{SSig} = (\mathsf{SKG}, \mathsf{SSig}, \mathsf{SVf})$ a standard signature scheme. A pair of private / public keys (sk, pk) is created by running $\mathsf{SKG}(1^\kappa)$. Given a message $m \in \{0,1\}^*$, a signature on m under pk is $\sigma_m = \mathsf{SSig}(sk, m)$. A signature σ on m is deemed valid if and only if $\mathsf{SVf}(pk, m, \sigma)$ returns valid. Regarding security, we use the standard notion of existential unforgeability under chosen message attack [17].

Boneh and Boyen Signature Scheme [4]. We recall here briefly the short signature scheme [4] introduced by Boneh and Boyen. The setup algorithm $\mathsf{BMGen}(1^\kappa)$ generates the public parameters of the scheme $(p, \mathbb{G}, \mathbb{G}_T, e, g)$[6]. The key generation algorithm $\mathsf{SKG}(1^\kappa)$ selects random integers $x, y \overset{R}{\leftarrow} \mathbb{Z}_p$ and sets $u = g^x$ and $v = g^y$. The secret key is $sk = (g, x, y)$ and the public key is $pk = (g, u, v)$. Given a message m and sk, the signing algorithm $\mathsf{SSig}(sk, m)$ works as follows. It selects $r_\sigma \overset{R}{\leftarrow} \mathbb{Z}_p$ such that $r_\sigma - (x+m)/y \neq 0 \bmod p$ and return the (randomized) signature $\sigma = (g^{\frac{1}{x+m+yr_\sigma}}, r_\sigma) = (\sigma', r_\sigma)$. Finally, in order to verify a signature σ on message m relative to the public key pk, the algorithm $\mathsf{SVf}(pk, m, \sigma)$ consists in checking that $e(\sigma', ug^m v^{r_\sigma}) = e(g, g)$. The scheme is secure in the standard model under the N-SDH assumption.

2.6 Simultaneous Hardness of Bits for Discrete Logarithm

Our construction relies on the idea of releasing gradually the bits of $\theta \in \mathbb{Z}_p$, the discrete logarithm in base g of $D = g^\theta$. A problem that could arise in this situation would be that some θ values are somehow easier to find than others, especially when some of the bits are released. This might help an adversary to retrieve θ much faster (by a factor greater than a polynomial) and thus break the security of our protocol. To overcome this issue we need to introduce the *Simultaneous hardness of bits of the discrete logarithm* assumption which states that

[5] The proof is very similar to the one introduced in [30] and can be found in the full version [7].

[6] We use symmetric bilinear map for the sake of exposition.

a polynomial time adversary cannot distinguish[7] between a random sequence of $l = \kappa - \omega(\log \kappa)$ bits and the first l bits of θ when given $D = g^\theta$.

Definition 5. *(Simultaneous hardness of bits for discrete logarithm) Let $\mathbb{G}$ be a cyclic group of prime order p. We say that the* Simultaneous hardness of bits for discrete logarithm *(SHDL) assumption holds, if for every PPT adversary $\mathcal{A}$ and for any $l = \omega(\log \kappa)$, we have that the following quantity is negligible in κ:*

$$Adv^{SHDL}(\mathcal{A}, \kappa) = |\Pr \left[\begin{array}{c} \theta \overset{R}{\leftarrow} \mathbb{Z}_p : \\ 1 \leftarrow \mathcal{A}(g^\theta, \theta[1..\kappa - l]) \end{array} \right] - \Pr \left[\begin{array}{c} \theta, \alpha \overset{R}{\leftarrow} \mathbb{Z}_p : \\ 1 \leftarrow \mathcal{A}(g^\theta, \alpha[1..\kappa - l]) \end{array} \right] |$$

where the probability is taken over the random choices of $\mathcal{A}$.

Schnorr [29] showed that the *SHDL* holds in the generic group model by computing the following upper bound on the advantage of the adversary:

$$Adv^{SHDL}(\mathcal{A}, \kappa) = O(\kappa(\kappa - l)\sqrt{t}(\frac{2^{\kappa - l}}{2^\kappa})^{1/4})$$

where t is the number of generic group operations of the adversary. Thus, if we set $l = \omega(\log \kappa)$, we obtain that $Adv^{SHDL}(\mathcal{A}, \kappa) = O(\kappa(\kappa - \omega(\log \kappa)) \sqrt{t}(2^{-\omega(\log \kappa)})^{1/4})$, which is negligible.

The recent work [13] by Duc and Jetchev suggests that results applying to groups of integers modulo a safe prime [27,24] can be extended to elliptic curves so to reduce the SHDL assumption to more standard ones.

3 A New Argument to Prove a Commitment Encrypts a Bit

In this section we describe a commitment scheme to encrypt a vector of values in $\mathbb{Z}_p$ and then provide a NIZK proof that each component of this vector is a bit. Our technique borrows from [21] in the sense we use the idea that if the value b encrypted is a bit then $b(b-1)$ must be equal to 0, and also from [20] by implementing a basic form of the restriction argument.

Our commitment scheme requires to generate a common reference string $\mathsf{CRS} = (g, g^s, g^{s^2}, ..., g^{s^N}) = (g_0, g_1, ..., g_N)$ where $s \overset{R}{\leftarrow} \mathbb{Z}_p$ is the trapdoor. To commit a bit b_i in position i using randomness $r_i \in \mathbb{Z}_p$, we compute the following slight variation of the Pedersen commitment $\mathsf{Commit}(b_i, r_i, i) = C_i = g^{r_i} g_i^{b_i}$. The commitment to the vector $\boldsymbol{B} = (b_1, b_2, ..., b_N)$ using the randomness $\boldsymbol{r} = (r_i)_{i \in [N]}$ will simply be the vector formed by the commitments for each bit in position i: $\boldsymbol{C} = (C_i)_{i \in [N]}$. Abusing a bit our notation, we will write $\boldsymbol{C} = \mathsf{Commit}(\boldsymbol{B}, \boldsymbol{r})$.

[7] Note that a PPT adversary can easily distinguish both bit strings if $l = \kappa - O(\log \kappa)$ by performing a brute force attack on the remaining bits as $2^{O(\log \kappa)}$ is a polynomial in κ.

We still need a NIZK that each commitment C_i is the encryption of a bit. The prover proceeds as follows: He computes the "translation" of the commitment by $N - i$ positions to the right, by providing the value $A_i = g_{N-i}^{r_i} g_N^{b_i}$. If we compute $e(A_i, C_i g^{-1})$ and try to express this quantity as $e(B_i, g)$, we realize by simple inspection (see correctness proof of Theorem 1) that a factor $g_{N+i}^{b_i(b_i-1)}$ will appear. Obviously the prover does not know g_{N+i} so in case $b_i \notin \{0, 1\}$ he will not be able to provide the second part of the proof, B_i. If b_i is indeed a bit then the prover will compute the proof $\pi_i = (A_i, B_i)$ in order to convince the verifier that C_i is the encryption of a bit relative to position i. The proofs for the following proposition and theorem can be found in the full version [7].

Proposition 2. *The vector commitment scheme described above is perfectly hiding and computationally binding under the N-BDHI assumption.*

Common reference string: Input $(1^\kappa, N)$

1. $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMGen}(1^\kappa)$
2. $s \overset{R}{\leftarrow} \mathbb{Z}_p$
3. Return $\mathsf{CRS} =< (p, \mathbb{G}, \mathbb{G}_T, g), (g_0, g_1, g_2, ..., g_N) >$ where for all $i \in [0..N] : g_i = g^{s^i}$.

Statement: The statement is formed by a vector of elements of $\mathbb{G}$: $(C_1, C_2, ..., C_N)$. The claim is that for each $i \in [N]$ there exists r_i, b_i such that $C_i = g^{r_i} g_i^{b_i}$ where $b_i \in \{0, 1\}$.

Proof: Input $(\mathsf{CRS}, \boldsymbol{B}, \boldsymbol{r})$

1. Check that $\boldsymbol{B} = (b_1, ..., b_N) \in \{0, 1\}^N$. Return $\perp$ if this is not the case.
2. Check that $\boldsymbol{r} = (r_1, ..., r_N) \in \mathbb{Z}_p^N$. Return $\perp$ if this is not the case.
3. For each $i \in [N]$ compute an argument π_i that C_i is the commitment to a bit in base g_i: $\pi_i = (A_i, B_i)$ where $A_i = C_i^{s^{N-i}}$ and B_i is such that $e(A_i, C_i g_i^{-1}) = e(B_i, g)$.
4. Return $\pi = (\pi_i)_{i \in [N]}$.

Verification: Input $(\mathsf{CRS}, \boldsymbol{C}, \pi)$

1. Parse $\boldsymbol{C}$ as $(C_i)_{i \in [N]}$. Check that $\boldsymbol{C} \in \mathbb{G}^N$.
2. Parse π as $((A_i, B_i))_{i \in [N]}$. Check that $\pi \in (\mathbb{G} \times \mathbb{G})^N$.
3. For each $i \in [N]$ check that:
   (a) $e(C_i, g_{N-i}) = e(A_i, g)$.
   (b) $e(A_i, C_i g_i^{-1}) = e(B_i, g)$.
4. Return valid if and only if all check pass, otherwise return $\perp$.

Fig. 1. NIZK proof of a commitment being the encryption of a binary vector

Theorem 1. *The protocol of Fig. 1 is a NIZK proof that the statement $\boldsymbol{C} = (C_i)_{i \in [N]}$ is such that for every $i \in [N]$ there exists $(r_i, b_i) \in (\mathbb{Z}_p \times \{0, 1\})$ with $C_i = g^{r_i} g_i^{b_i}$. The NIZK proof has perfect completeness, perfect zero-knowledge and computational soundness under the N-BDHI assumption.*

4 Base Equivalence Argument

Let $\theta \overset{R}{\leftarrow} \mathbb{Z}_p$. Consider the commitment to the bit vector $\boldsymbol{C} = (C_i)_{i\in[\kappa]} = (g^{r_i} g_i^{\theta[i]})_{i\in[\kappa]}$ where $r_i \in \mathbb{Z}_p$ for each $i \in [\kappa]$ and also $D = g^\theta$. In this section we introduce a NIZK proof to show that indeed each bit commitment in position i, C_i, encrypts the i^{th} bit of θ, which is hidden as the discrete logarithm of D. This argument will allow us to blind the signature with some random factor θ (in the exponent) and then reveal each bit of this exponent gradually without leaking any additional information. The idea is the following. Given $\theta \in \mathbb{Z}_p$ and $\boldsymbol{C} = (g^{r_i} g_i^{\theta[i]})_{i\in[\kappa]}$, the prover proceeds in two steps. First he computes $D' = \frac{\prod_{i\in[\kappa]} g^{r_i} g_i^{\theta[i]}}{g^r}$ where $r = \sum_{i\in[\kappa]} r_i$. Here the prover computes some compressed representation of the bit vector commitment and removes the randomness. Observe however that as θ is uniformly random, thus so is D'. The prover will need to convince the verifier that r is indeed the accumulated randomness of the bit vector commitment. To do so he computes $U = D'^{\frac{1}{s}} = (\prod_{i\in[\kappa]} g_i^{\theta[i]})^{\frac{1}{s}} = \prod_{i\in[\kappa]} g_{i-1}^{\theta[i]}$ where we recall that $g_0 = g$. Observe that this value can be computed without knowing s. In order to verify this proof, the verifier will check that $e(\frac{\prod_{i\in[\kappa]} C_i}{g^r}, g) = e(U, g_1)$. Intuitively, once the randomness of the bit vector is removed one can move the vector to the left by one position. If r would not be equal to $\sum_{i\in[\kappa]} r_i$, this would not be possible without breaking some assumption. The second step consists in checking that the condensed bit vector commitment $U = \prod_{i\in[\kappa]} g_{i-1}^{\theta[i]}$ is "equivalent" to the simple commitment g^θ. This is done by noting that $U = \prod_{i\in[\kappa]} g_{i-1}^{\theta[i]} = g^{P(s)}$ where $P(\cdot)$ is the polynomial $P(X) = \sum_{i\in[\kappa]} \theta[i] X^{i-1}$. This means in particular that $P(2) = \sum_{i\in[\kappa]} \theta[i] 2^{i-1} = \theta$. Thus, we need to prove that $P(s) - P(2) = P(s) - \theta$ is divisible by $s - 2$. The prover can compute the coefficients of the formal polynomial $W(\cdot)$ such that $P(X) - P(2) = W(X)(X - 2)$, then using the common reference string CRS the prover obtains $V = g^{W(s)}$. Verifying the "base equivalence" statement consists in checking that $e(\frac{U}{D}, g) = e(V, g_1 g^{-2}) = e(V, g^{s-2})$. This means that indeed $\theta = P(2)$ and thus the coefficients of $P(\cdot)$ correspond to the binary decomposition of θ. The full protocol is detailed in Fig. 2 and the proof of the theorem is available in the full version [7].

Theorem 2. *The protocol in Fig. 2 is a NIZK proof that the bits of the discrete logarithm of D correspond to the bit vector committed in $(C_i)_{i\in[\kappa]}$. The NIZK proof has perfect completeness, perfect zero-knowledge and computational soundness under the κ-SDH assumption.*

5 Fair Exchange of Short Signatures without TTP

Our fair exchange protocol for digital signatures works as follows. At the beginning a common reference string CRS is generated. Then each participant runs

Common reference string: Input $(1^\kappa, \kappa)$

1. $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMGen}(1^\kappa)$.
2. $s \overset{R}{\leftarrow} \mathbb{Z}_p$.
3. Return $\mathsf{CRS} =< (p, \mathbb{G}, \mathbb{G}_T, e), (g_0, g_1, g_2, ..., g_\kappa) >$ where for all $i \in [0..\kappa] : g_i = g^{s^i}$.

Statement: The statement is formed by a vector of elements of $\mathbb{G}$: $(D, C_1, C_2, ..., C_\kappa)$ where $(C_i)_{i\in[\kappa]}$ is a commitment to a bit vector as defined in Sect. 3. The claim is that the vector formed by the binary decomposition of the discrete logarithm of D is equal to the bit vector committed in $(C_i)_{i\in[\kappa]}$.

Proof: Input $(\mathsf{CRS}, \theta, r_1, ..., r_\kappa)$

1. Check that $D = g^\theta$. Return $\perp$ if this is not the case.
2. Compute for every $i \in [\kappa]$: $C_i = g^{r_i} g_i^{\theta[i]}$.
3. Compute $r = \sum_{i\in[\kappa]} r_i$.
4. Compute $U = (\frac{\prod_{i\in[\kappa]} C_i}{g^r})^{\frac{1}{s}}$ using the common reference string CRS and the bit vector $\theta[\cdot]$.
5. Compute the formal polynomial $W(\cdot)$ such that $P(X) - P(2) = W(X)(X-2)$ where $P(X) = \sum_{i\in[\kappa]} \theta[i]X^{i-1}$, and $P(2) = \sum_{i\in[\kappa]} \theta[i]2^{i-1} = \theta$. Compute $V = g^{W(s)}$ using the coefficients of the formal polynomial $W(\cdot)$ and the common reference string CRS.
6. Return $\pi = (r, U, V)$.

Verification: Input (CRS, C, π)

1. Parse C as $(D, (C_i)_{i\in[\kappa]})$.
2. Parse π as (r, U, V).
3. Check that $r \in \mathbb{Z}_p$.
4. Check that $(U, V, D, C_1, ..., C_\kappa) \in \mathbb{G}^{\kappa+3}$.
5. Compute $D' = \frac{\prod_{i\in[\kappa]} C_i}{g^r}$.
6. Check that $e(D', g) = e(U, g_1)$.
7. Check that $e(\frac{U}{D}, g) = e(V, g_1 g^{-2})$.
8. Accept if all tests pass in which case return valid otherwise return $\perp$.

Fig. 2. NIZK proof that a basic commitment is equivalent to a bit vector commitment

$\mathsf{FEKeyGen}(1^\kappa)$ to obtain a pair of (public/private) keys (pk, sk) for the signing algorithm. At this point each participant executing $\mathsf{EncSigGen}(\mathsf{CRS}, sk, m)$ will compute an encrypted signature γ for the message m, using the signature σ_m blinded with some factor θ. This value γ will also contain the proofs that relate the signature σ_m with some bit vector commitment to θ.

The rest is straightforward: Each participant sends the encrypted signature. If all the verifications pass, the first participant $\mathcal{P}_A$ will ask to $\mathcal{P}_B$ to open the commitment of the first bit of θ_A. If the opening is successful, $\mathcal{P}_B$ will do the same for its own blinding factor θ_B. The process is repeated for each bit until all the bits of the blinding factors are recovered. Finally, each player can compute the signature by "canceling out" the blinding factor θ. The abstract syntax of the protocol is described in Fig. 3.

We describe now more in detail how the encrypted signature is constructed, which is the core of our construction. The encrypted signature contains:

1. A commitment $\boldsymbol{C}$ to the bit string formed by the bits of θ as described in Section 3.
2. $\tilde{\sigma}$, the signature of the message m blinded by θ.
3. Proofs to guarantee that the bit vector commitment encrypts the binary decomposition of the blinding factor θ.
4. A proof in order to convince the verifier that γ is the encryption of σ_m under some blinding factor θ which is hidden in the basic commitment g^θ.
5. A proof of knowledge of the discrete logarithm of D and a proof of knowledge of the representation of each bit commitment of the vector $\boldsymbol{C}$. These proofs of knowledge will allow us to keep simulating the adversary despite it aborts.

A detailed description of the concrete protocol is given in Fig. 4.

	$\mathcal{P}_A(\mathsf{CRS}, m_A, m_B)$		$\mathcal{P}_B(\mathsf{CRS}, m_A, m_B)$
1	$(sk_A, pk_A) \leftarrow \mathsf{FEKeyGen}(1^\kappa)$		
2	pk_A	$\longrightarrow$	
3			$(sk_B, pk_B) \leftarrow \mathsf{FEKeyGen}(1^\kappa)$
4		$\longleftarrow$	pk_B
5	$(\theta_A, \boldsymbol{r}_A, \gamma_A) \leftarrow \mathsf{EncSigGen}(\mathsf{CRS}, sk_A, m_A)$		
6	γ_A	$\longrightarrow$	
7			$(\theta_B, \boldsymbol{r}_B, \gamma_B) \leftarrow \mathsf{EncSigGen}(\mathsf{CRS}, sk_B, m_B)$
8		$\longleftarrow$	γ_B
10	$v \leftarrow \mathsf{EncSigCheck}(\mathsf{CRS}, pk_B, m_B, \gamma_B)$		
11	**if** $v = 0$ **then ABORT**		
12			$v \leftarrow \mathsf{EncSigCheck}(\mathsf{CRS}, pk_A, m_A, \gamma_A)$
13			**if** $v = 0$ **then ABORT**
for $i = 1$ **to** κ:			
14	$\mathsf{open}_{A,i} \leftarrow \mathsf{KeyBitProofGen}(\mathsf{CRS}, \boldsymbol{r}_A, \theta_A, i)$		
15	$\mathsf{open}_{A,i}$	$\longrightarrow$	
16			$\mathsf{open}_{B,i} \leftarrow \mathsf{KeyBitProofGen}(\mathsf{CRS}, \boldsymbol{r}_B, \theta_B, i)$
17		$\longleftarrow$	$\mathsf{open}_{B,i}$
19	$v_i \leftarrow \mathsf{KeyBitCheck}(\mathsf{CRS}, \mathsf{open}_{B,i}, i)$		
20	**if** $v_i = 0$ **then ABORT**		
21			$v_i \leftarrow \mathsf{KeyBitCheck}(\mathsf{CRS}, \mathsf{open}_{A,i}, i)$
22			**if** $v_i = 0$ **then ABORT**
end for			
23	$\sigma_{m_B} \leftarrow \mathsf{EncSigDecrypt}(\gamma_B, \theta_B)$		
24			$\sigma_{m_A} \leftarrow \mathsf{EncSigDecrypt}(\gamma_A, \theta_A)$

Fig. 3. Abstract fair exchange protocol

We say that the protocol is *perfectly complete*[8] if, and only if, both players $\mathcal{P}_A$ and $\mathcal{P}_B$ that follow the protocol obtain respectively $\sigma_A = \mathtt{SSig}(sk_B, m_B)$, the signature of message m_B and $\sigma_B = \mathtt{SSig}(sk_A, m_A)$, the signature of message m_A, with probability 1.

[8] Here *complete* does not refer to fairness.

We say that the protocol is *(partially) fair* if, at the end of the execution of the protocol (be it normal or anticipated by the abortion of the adversary), the probability of both players to recover their corresponding signature differs at most by a polynomial factor in the security parameter κ. As mentioned in the introduction, the advantage of this approach is that it avoids trying to compare the *exact* running time of the participants and thus allows to capture in a simple, but precise manner, the intuition of *partial fairness.*

Definition 6. *(Partial fairness) We define the* partial fairness *of the protocol through the following experiment: The adversary $\mathcal{A}$ plays the role of the corrupted player say w.l.o.g. $\mathcal{P}_A$. Thus, $\mathcal{P}_B$ is honest and follows the protocol. $\mathcal{O}_{\texttt{SSig}}(\cdot)$ is the signing oracle for the signature scheme* `SSig` *relative to the public key pk_B of $\mathcal{P}_B$.*

1. *$\mathcal{A}$ asks for signature computations for arbitrary messages to $\mathcal{O}_{\mathsf{SSig}}(\cdot)$.*
2. *$\mathcal{A}$ chooses the messages m_A and m_B on which the fair exchange protocol will be run, with the restriction that m_B must not have been requested before to $\mathcal{O}_{\mathsf{SSig}}(\cdot)$.*
   $\mathcal{A}$ computes also its public key pk_A and sends it to $\mathcal{P}_B$.
3. *$\mathcal{A}$ then interacts in arbitrary way with $\mathcal{P}_B$.*
4. *If $\mathcal{A}$ has aborted before ending the protocol, then let $\theta^*_A[1..i]$ $(0 \leq i \leq \kappa)$ be the partial blinding obtained by $\mathcal{P}_B$. At this point we assume that $\mathcal{P}_B$ will try to compute $\mathsf{SSig}(sk_A, m_A)$ by choosing at random some element in the remaining space of size $2^{\kappa-i}$. We call this tentative signature σ_B.*
5. *$\mathcal{A}$ keeps running its own algorithm and finally outputs a tentative signature σ_A on m_B relative to public key pk_B.*

The protocol is said to be partially fair *if and only if there exists some polynomial $Q(\cdot)$ such that*

$$\frac{\Pr\left[\,\mathsf{SVf}(pk_B, m_B, \sigma_A) = \mathsf{valid}\,\right]}{\Pr\left[\,\mathsf{SVf}(pk_A, m_A, \sigma_B) = \mathsf{valid}\,\right]} \leq Q(\kappa)$$

where the probability is taken over the random choices of $\mathcal{A}$ and $\mathcal{P}_B$.

As the signature scheme presented in [4] is secure under the κ-SDH assumption, we have the following result (see the full version [7] for the proof).

Theorem 3. *The protocol described in Fig. 4 is complete. Moreover if the κ-SDH assumption, the κ-BDHI assumption and the SHDL assumption hold, and a securely precomputed common reference string is available, then it is secure in the random oracle model according to definition 6.*

```
FESetup($1^\kappa$)
  1. $(p, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMGen}(1^\kappa)$
  2. $s \overset{R}{\leftarrow} \mathbb{Z}_p$
  3. Return CRS $=< (p, \mathbb{G}, \mathbb{G}_T, e, g), (g_0, g_1, g_2, ..., g_\kappa) >$ where for all $i \in [0..\kappa] : g_i = g^{s^i}$.
FEKeyGen($1^\kappa$)
  1. $(sk, pk) \leftarrow \mathsf{SKG}(1^\kappa)$ where $sk = (g, x, y)$ and $pk = (g, u, v)$ with $u = g^x$ and $v = g^y$,
     like described in section 2.5.
  2. Return $(sk, pk)$.
EncSigGen(CRS, $sk, m$)
  1. Compute $\theta \overset{R}{\leftarrow} \mathbb{Z}_p$.
  2. Compute $D = g^\theta$.
  3. Compute $\boldsymbol{C} = (C_i)_{i \in [\kappa]} = (g^{r_i} g_i^{\theta[i]})_{i \in [\kappa]}$.
  4. Compute $\pi_1$ that shows that $\boldsymbol{C}$ is the encryption of a binary vector as described in
     figure 1.
  5. Compute $\pi_2$ that shows that $\boldsymbol{C}$ is the encryption of the bits of the binary decompo-
     sition of the blinding factor $\theta$ as described in figure 2.
  6. Compute $PK_\theta = PK\{\theta : g^\theta\}$ as described in section 2.2.
  7. Compute $\boldsymbol{PK}$, a vector where each component at position $i$ is ZKPoK for the repre-
     sentation of $C_i$ in base $(g, g_i)$. $\boldsymbol{PK} = (PK\{(r_i, \theta[i]) : g^{r_i} g_i^{\theta[i]}\})_{i \in [\kappa]}$ as described in
     section 2.2.
  8. Parse $sk$ as $(g, x, y)$.
  9. Set $r_\sigma \overset{R}{\leftarrow} \mathbb{Z}_p$.
 10. Compute $\sigma = (\sigma', r_\sigma) \leftarrow \mathsf{SSig}(sk, m)$ where $\sigma' = g^{\frac{1}{x+m+yr_\sigma}}$.
 11. Set $\tilde{\sigma} \leftarrow (\sigma'^\theta = g^{\frac{\theta}{x+m+yr_\sigma}}, r_\sigma) = (\tilde{\sigma}', r_\sigma)$.
 12. Set $\gamma \leftarrow (D, \boldsymbol{C}, \pi_1, \pi_2, PK_\theta, \boldsymbol{PK}, \tilde{\sigma})$.
 13. Return $(\theta, \boldsymbol{r}, \gamma)$, where $\boldsymbol{r} = (r_i)_{i \in [\kappa]}$ is the randomness vector of the commitment $\boldsymbol{C}$.
EncSigCheck(CRS, $pk, m, \gamma$)
  1. Parse $\gamma$ as $\gamma = (D, \boldsymbol{C}, \pi_1, \pi_2, PK_\theta, \boldsymbol{PK}, \tilde{\sigma})$.
  2. Check $\pi_1$ as described in figure 1.
  3. Check $\pi_2$ as described in figure 2.
  4. Check $PK_\theta$ using $D$ and $PK_\theta$ as described in section 2.2.
  5. Check the zero-knowledge proof of knowledge $\boldsymbol{PK}$ using $\boldsymbol{C}$ and $\boldsymbol{PK}$ as described in
     section 2.2.
  6. Parse $pk$ as $pk = (g, u, v)$.
  7. Check that $e(\tilde{\sigma}, ug^m v_\sigma^r) = e(D, g)$.
  8. Return valid if all tests pass, $\perp$ otherwise.
KeyBitProofGen(CRS, $\boldsymbol{r}, \theta, i$)
  1. Opens the $i^{\text{th}}$ commitment of $\boldsymbol{C}$, that is $(\theta[i], r_i)$ such that $C_i = g^{r_i} g_i^{\theta[i]}$.
  2. Return open $\leftarrow (\theta[i], r_i)$.
KeyBitCheck(CRS, open, $i$)
  1. Parse open as open $= (b, r)$
  2. Check that $C_i = g^r g_i^b$ and $b \in \{0, 1\}$.
EncSigDecrypt($\gamma, \theta$)
  1. Parse $\gamma$ as $\gamma = (D, \boldsymbol{C}, \pi, PK_\theta, \boldsymbol{PK}, \tilde{\sigma})$.
  2. Parse $\tilde{\sigma}$ as $\tilde{\sigma} = (\tilde{\sigma}', r_\sigma)$.
  3. Compute $\sigma' = \tilde{\sigma}'^{\frac{1}{\theta}}$.
  4. Return $\sigma = (\sigma', r_\sigma)$.
```

Fig. 4. Implementation of the fair exchange protocol

6 Conclusion and Future Work

In this work we introduced a practical protocol to exchange short signatures [4] fairly without relying on a TTP. It seems our approach can be applicable to other signature schemes or more generally to the exchange of values which are computed from a secret and are publicly verifiable using bilinear maps. Thus, our techniques might be extended in order to obtain a general framework to build practical fair protocols involving bilinear maps.

Acknowledgments. The author is very grateful to Anna Lysyanskaya for pointing out a gap in the proof of Theorem 3.

References

1. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: CCS, pp. 7–17. ACM Press (April 1997)
2. Blum, M.: How to exchange (secret) keys. ACM Transactions on Computer Systems 1(2), 175–193 (1983)
3. 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)
4. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. Journal of Cryptology 21(2), 149–177 (2008)
5. Boneh, D., Naor, M.: Timed Commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000)
6. Brickell, E.F., Chaum, D., Damgård, I.B., van de Graaf, J.: Gradual and Verifiable Release of a Secret. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 156–166. Springer, Heidelberg (1988)
7. Camacho, P.: Fair Exchange of Short Signatures Without Trusted Third Party (2012), http://eprint.iacr.org/2012/288
8. Camenisch, J., Stadler, M.: Proof Systems for General Statements about Discrete Logarithms (1997), ftp://ftp.inf.ethz.ch/pub/crypto/publications/CamSta97b.ps
9. Choi, S.G., Hwang, K.-W., Katz, J., Malkin, T., Rubenstein, D.: Secure Multi-Party Computation of Boolean Circuits with Applications to Privacy in On-Line Marketplaces. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 416–432. Springer, Heidelberg (2012)
10. Cleve, R.: Limits on the security of coin flips when half the processors are faulty. In: STOC, pp. 364–369. ACM Press (November 1986)
11. Cleve, R.: Controlled Gradual Disclosure Schemes for Random Bits and Their Applications. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 573–588. Springer, Heidelberg (1990)
12. Damgård, I.: Practical and Provably Secure Release of a Secret and Exchange of Signatures. Journal of Cryptology 8(4), 201–222 (1995)
13. Duc, A., Jetchev, D.: Hardness of Computing Individual Bits for One-Way Functions on Elliptic Curves. In: Safavi-Naini, R. (ed.) CRYPTO 2012. LNCS, vol. 7417, pp. 832–849. Springer, Heidelberg (2012)
14. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Communications of the ACM 28(6), 637–647 (1985)

15. 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)
16. Garay, J.A., MacKenzie, P.D., Prabhakaran, M., Yang, K.: Resource Fairness and Composability of Cryptographic Protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 404–428. Springer, Heidelberg (2006)
17. Goldwasser, S., Micali, S., Rivest, R.L.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal on Computing 17(2), 281 (1988)
18. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete Fairness in Secure Two-Party Computation. Journal of the ACM 58(6), 1–37 (2011)
19. Gordon, S.D., Katz, J.: Partial Fairness in Secure Two-Party Computation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 157–176. Springer, Heidelberg (2010)
20. Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (2010)
21. Groth, J., Ostrovsky, R., Sahai, A.: Perfect Non-interactive Zero Knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
22. Huang, Q., Wong, D.S., Susilo, W.: The Construction of Ambiguous Optimistic Fair Exchange from Designated Confirmer Signature without Random Oracles. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 120–137. Springer, Heidelberg (2012)
23. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-Size Commitments to Polynomials and Their Applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (2010)
24. MacKenzie, P.D., Patel, S.: Hard Bits of the Discrete Log with Applications to Password Authentication. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 209–226. Springer, Heidelberg (2005)
25. Micali, S.: Simple and fast optimistic protocols for fair electronic exchange. In: PODC, pp. 12–19. ACM Press, New York (2003)
26. Mitsunari, S., Sakai, R., Kasahara, M.: A New Traitor Tracing. In: EICE, vol. E 85-A, pp. 481–484 (2002)
27. Patel, S., Sundaram, G.S.: An Efficient Discrete Log Pseudo Random Generator. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 304–317. Springer, Heidelberg (1998)
28. Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
29. Schnorr, C.P.: Security of Almost ALL Discrete Log Bits. Electronic Colloquium on Computational Complexity (1998)
30. Zhang, F., Safavi-Naini, R., Susilo, W.: An Efficient Signature Scheme from Bilinear Pairings and Its Applications. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 277–290. Springer, Heidelberg (2004)

Fully Secure Attribute-Based Systems with Short Ciphertexts/Signatures and Threshold Access Structures

Cheng Chen[1], Jie Chen[2], Hoon Wei Lim[2], Zhenfeng Zhang[1], Dengguo Feng[1], San Ling[2], and Huaxiong Wang[2]

[1] Institute of Software, Chinese Academy of Sciences, Beijing, China
{chencheng,zfzhang,feng}@is.iscas.ac.cn
[2] Division of Mathematical Sciences,
School of Physical & Mathematical Sciences,
Nanyang Technological University, Singapore
s080001@e.ntu.edu.sg, {hoonwei,lingsan,hxwang}@ntu.edu.sg

Abstract. It has been an appealing but challenging goal in research on *attribute-based encryption* (ABE) and *attribute-based signatures* (ABS) to design a *secure* scheme with *short* ciphertexts and signatures, respectively. While recent results show that some promising progress has been made in this direction, they do not always offer a satisfactory level of security, i.e. achieving *selective* rather than *full* security.

In this paper, we aim to achieve *both* full security and short ciphertexts/signatures for *threshold* access structures in the ABE/ABS setting. Towards achieving this goal, we propose generic property-preserving conversions from inner-product systems to attribute-based systems. We first give concrete constructions of fully secure IPE/IPS with constant-size ciphertexts/signatures in the composite order groups. By making use of our IPE/IPS schemes as building blocks, we then present concrete constructions of fully secure key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE) with constant-size ciphertexts, and a fully secure ABS with constant-size signatures with perfect privacy for threshold access structures. These results give rise to the first constructions satisfying the aforementioned requirements. Our schemes reduce the number of pairing evaluations to a constant, a very attractive property for practical attribute-based systems. Furthermore, we show that our schemes can be extended to support large attribute universes and more expressive access structures.

1 Introduction

Attribute-Based Encryption. The notion of attribute-based encryption (ABE) [14] was initially developed from the fuzzy identity-based encryption (FIBE) primitive [31], which allows some sort of error-tolerance. That is, identities are viewed as sets of attributes, and a user can decrypt if she possesses keys for enough of (but not necessarily all) attributes a ciphertext is encrypted under. At the same time, colluding users cannot combine their keys to decrypt a ciphertext which none of them were able to decrypt independently. Since then, ABE finds many useful applications in cryptographic access control systems, and is further categorized into ciphertext-policy ABE (CP-ABE) and key-policy ABE (KP-ABE). In the former, a secret key is associated with an attribute set; a user can decrypt a ciphertext if and only if the attribute set satisfies the access structure associated with the ciphertext. In the latter, on the contrary, a secret key is associated with an access structure; a user can decrypt a ciphertext associated with an attribute set if and only if the attribute set satisfies the access structure associated with the user's secret key. Recently, the signature analogue of ABE, i.e. attribute-based signatures (ABS), has been introduced [24] (see also [12,23,29]). ABS offers an interesting property in that a signature does not reveal the identity of the signer (hence preserving the privacy of the signer), since it is generated and can be verified based on only the signer's attributes.

While it is desirable that an attribute-based system to be as expressive as possible (in terms of enforcing an access control policy), two major factors to consider when designing an ABE/ABS scheme are *efficiency* and *security*. Majority of existing ABE and ABS schemes have linear-size ciphertexts and signatures, respectively, in the maximal number of attributes. Indeed, recent proposals, such as [1,10,11,15,16], have focused on reducing the sizes of ciphertexts and signatures in the attribute-based setting. Of these, Herranz et al. [16] presented a CP-ABE scheme supporting threshold access policies with constant-size ciphertexts; while Attrapadung and Libert [1] proposed a KP-ABE scheme supporting general access structures with constant-size ciphertexts; and Herranz et al. [15] gave two constructions of ABS with constant-size signatures for threshold predicates. While these works have taken a significant step forward towards improving the efficiency of ABE/ABS, they have so far not achieved a satisfactory level of security. In other words, the aforementioned schemes achieve better efficiency at the expense of weaker security. They are proven to be only *selectively secure*, i.e. an adversary is required to announce the target he intends to attack before seeing the public (system) parameters. The goal of this paper is to offer solutions that achieve both *full security* and *constant-size* ABE ciphertexts or ABS signatures.

Predicate Encryption. Functional encryption (FE) [2,8,22,28] is recently seen as a new vision of public key encryption. In an FE system, a decryption key allows a user to learn a function of the encrypted data. Given a functionality $F(\cdot,\cdot)$, an authority holding a master secret key can generate a decryption key SK_k that is able to compute the function $F(k,x)$ from the encryption of x. The security of the FE system guarantees that one cannot learn anything more about x. ABE and predicate encryption are both example primitives that satisfy the notion of FE.

The concept of predicate encryption (PE) was proposed by Katz, Sahai and Waters [19]. Particularly, they devised a PE scheme for inner products: a ciphertext encrypted for the attribute vector $\boldsymbol{y}$ can only be opened by a key $\boldsymbol{x}$ that gives an inner-product $\boldsymbol{x} \cdot \boldsymbol{y} = 0$. They showed that the inner-product encryption (IPE) suffices to give functional encryption associated with the evaluation of polynomials or formulae in conjunctive/disjunctive normal form (CNF/DNF). Attrapadung and Libert [2] proposed a fully secure IPE scheme with constant-size ciphertexts based on Waters' tag-based IBE scheme [34]; while Okamoto and Takashima [30] also proposed an IPE scheme with analogous properties on dual pairing vector spaces. We note that it seems possible to construct fully secure ABE with constant-size ciphertexts directly from these two IPE schemes. However, the resulting ABE schemes have two notable shortcomings: (i) the ABE schemes are rather complex[1] and it is not always clear how full security can be proven; and (ii) the access structures are restricted to a single AND/OR-gate.

Our Approach. In this paper, we consider how PE can be used to construct fully secure ABE and ABS with constant-size ciphertexts and signatures, respectively. Moreover, we would like our constructions to support *threshold access structures*. (Henceforth, we use a prefix 't' to indicate that an attribute-based system supports threshold access structures, for example tKP-ABE and tCP-ABE.)

Our general idea is to construct attribute-based systems from inner-product systems by extending the technique from [19]: we treat a vector space as an attribute universe, where each coordinate corresponds to an attribute; for an attribute subset S, a coordinate is equal to 1 if its corresponding attribute is an element of S, otherwise, the coordinate equals to 0. If two subsets have t common attributes, the corresponding vectors overlap in exactly t coordinates, and the inner-product of them equals to t. In addition, we require some coordinates to express threshold values and to allow an inner-product between the vector associated with the attribute subset S and the vector associated with an access structure (if S satisfies the access structure).

One major advantage of such a conversion technique is that the resulting attribute-based construction preserves the sizes of ciphertexts/signatures and the security of the corresponding inner-product scheme. This implies that we can obtain fully (or adaptively) secure tABE with constant-size ciphertexts and fully secure tABS (in terms of unforgeability and perfect privacy) with constant-size signatures, so long as the IPE and the signature analogue (IPS) used in the conversion comply to these properties. We also note that there currently seems to be no suitable IPS candidate for our purpose. For the sake of simplicity, we construct IPE and IPS schemes with the required properties in the composite order group setting as an intermediate step towards achieving fully secure and efficient tABE and tABS. Although it is possible to construct the schemes under the prime order groups (as we will discuss in Section 5), our IPE/IPS schemes are more compact in the composite order groups setting since they do not employ additional tags as with the schemes in [2,30].

[1] Current constructions [2,21,28,29,30] under the prime order groups and proven secure using the dual encryption system proof methodology typically have a multitude of parameters and intricate compositions, in comparisons to those under the composite order groups [20,22].

Moreover, since the secret key components (of IPE/IPS) used in our conversion are independent from each other, it becomes more natural to derive the required security proof using the dual system proof technique as compared to those in [15], for example. We can now make an (ABE/ABS) secret key semi-functional by turning the secret key components sequentially in a hybrid security manner.

Our Contributions. We first give appropriate formal definitions and security models for predicate signatures. We then specify three generic property-preserving conversions: (i) IPE to tKP-ABE, (ii) IPE to tCP-ABE, and (iii) IPS to tABS. Further, we give concrete constructions of IPE and IPS in the composite order group setting. Our IPE scheme is fully secure with constant-size ciphertexts and our IPS scheme is fully unforgeable and perfectly private, and has constant-size signatures. They are proven secure under the complexity assumptions used by Lewko and Waters [22].

Using our IPE scheme as a building block, we present concrete constructions of fully secure tKP-ABE and tCP-ABE with constant-size ciphertexts. The ciphertexts of both the tKP-ABE and the tCP-ABE schemes consist of 3 group elements. The security of our tKP-ABE and tCP-ABE inherits the security of the underlying IPE scheme. We also give a fully secure tABS construction that relies on our IPS scheme with constant-size signatures. Our tABS produces signatures that each also consists of 3 group elements. The full unforgeability and prefect privacy properties are preserved from the underlying IPS scheme. To the best of our knowledge, there are no previous schemes that satisfy these properties. In addition, our schemes reduce the number of pairing evaluations to a constant; this appears to be a very attractive property for attribute-based systems. Table 1 shows that in comparisons with previous work, our attribute-based schemes have better efficiency and higher security. Here, PP denotes public parameters, SK denotes secret keys, CT denotes ciphertexts, Sig denotes signatures, all in the attribute-based setting. Pai denotes the number of paring computations required in the scheme.

We remark that all our schemes in Section 4 are for small universes. Thus as a further contribution, we show that the schemes can be extended to support large universes[2] by borrowing the tricks from [31] in the standard model. Moreover, we show that our constructions can deal with more general access structures, as discussed in Section 5.

2 Predicate Encryption and Signatures

We give the definitions and security models for predicate encryption and predicate signature. We also show how these definitions capture the notions of ABE/ABS and IPE/IPS and provide example instantiations of these primitives.

[2] The attribute universe is a set containing all the attributes defined for an attribute-based system. In the small universe case, the size of the attribute universe is defined at system setup. In the large universe case, the number of attributes is unlimited.

Table 1. Comparisons between existing and our ABE/ABS systems

	scheme	security	size of PP	size of SK	size of CT or Sig	expressiveness	Pai
CP-ABE	EM+09 [11]	selective	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	(n,n)-threshold	2
	CZF11 [10]	selective	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	and-gate	2
	HLR10 [16]	selective	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	threshold	3
	GZC12 [18]	selective	$\mathcal{O}(n)$	$\mathcal{O}(n)^2$	$\mathcal{O}(1)$	threshold	3
	OT10 [28]	full	$\mathcal{O}(n)^2$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	general	$\mathcal{O}(n)$
	Our CP-ABE	full	$\mathcal{O}(n)$	$\mathcal{O}(n)^2$	$\mathcal{O}(1)$	threshold	2
KP-ABE	ABP11 [1]	selective	$\mathcal{O}(n)$	$\mathcal{O}(n)^2$	$\mathcal{O}(1)$	general	3
	OT10 [28]	full	$\mathcal{O}(n)^2$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	general	$\mathcal{O}(n)$
	Our KP-ABE	full	$\mathcal{O}(n)$	$\mathcal{O}(n)^2$	$\mathcal{O}(1)$	threshold	2
ABS	HLLR12a [15]	selective	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	threshold	12
	HLLR12b [15]	selective	$\mathcal{O}(n)$	$\mathcal{O}(n)^2$	$\mathcal{O}(1)$	threshold	3
	OT11 [29]	full	$\mathcal{O}(n)^2$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	general	$\mathcal{O}(n)$
	Our ABS	full	$\mathcal{O}(n)$	$\mathcal{O}(n)^2$	$\mathcal{O}(1)$	threshold	3

2.1 Predicate Encryption

Predicate encryption (PE) is a variant of functional encryption, which was formally defined in [8]. We now define the syntax of predicate encryption and its security model. (Our definitions follow the general framework of those given in [2,19].[3])

Let $\mathfrak{R} : \mathcal{K} \times \mathcal{X} \rightarrow \{0,1\}$ be a predicate, where $\mathcal{K}$ and $\mathcal{X}$ denote "role" and "policy" spaces. A predicate encryption scheme $\Pi_{\mathsf{PE}} = (\mathsf{PE.Setup}, \mathsf{PE.KeyGen}, \mathsf{PE.Enc}, \mathsf{PE.Dec})$ for $\mathfrak{R}$ consists of four probabilistic polynomial-time (PPT) algorithms that are described as follows:

- $\mathsf{PE.Setup}(\kappa, des)$: The algorithm takes a security parameter κ and a scheme description des as input. It outputs some public parameters PP and a master secret key MSK.
- $\mathsf{PE.KeyGen}(\mathsf{PP}, \mathsf{MSK}, \boldsymbol{y})$: The algorithm takes as input the public parameters PP, a master key MSK and a role $\boldsymbol{y} \in \mathcal{K}$. It returns a secret key $\mathsf{SK}_{\boldsymbol{y}}$ associated with $\boldsymbol{y}$.
- $\mathsf{PE.Enc}(\mathsf{PP}, \boldsymbol{x}, M)$: The algorithm takes as input a message M, an encrypting policy $\boldsymbol{x} \in \mathcal{X}$ and public parameters PP. It outputs a ciphertext CT.
- $\mathsf{PE.Dec}(\mathsf{PP}, \boldsymbol{x}, \mathsf{SK}_{\boldsymbol{y}}, \mathsf{CT})$: The algorithm takes as input a secret key $\mathsf{SK}_{\boldsymbol{y}}$, a ciphertext CT with a policy $\boldsymbol{x}$ and public parameters PP. It outputs a message M or $\perp$.

For correctness, we require that, for all $\boldsymbol{y} \in \mathcal{K}$ and $\boldsymbol{x} \in \mathcal{X}$, if $\mathfrak{R}(\boldsymbol{x}, \boldsymbol{y}) = 1$, then

$$\mathsf{PE.Dec}(\mathsf{PP}, \boldsymbol{x}, \mathsf{PE.KeyGen}(\mathsf{PP}, \mathsf{MSK}, \boldsymbol{y}), \mathsf{PE.Enc}(\mathsf{PP}, \boldsymbol{x}, M)) = M,$$

[3] Our definition of predicate encryption here and throughout the paper refers to the class of PE with public index (as with [8]), in which the decryption algorithm should input the index component, as well as the bit length, of the plaintext. This type of PE has also been informally referred to as "payload hiding" in the literature [19].

where PP and MSK have been obtained by properly executing the $\mathsf{PE.Setup}$ algorithm.

Security Model. In this paper, we consider only the payload-hiding security, which requires that ciphertexts hide the encrypted messages from an adversary but they do not hide their underlying encrypting policies. Let κ be a security parameter. We describe the security model against chosen plaintext attacks (CPA) for a PE scheme Π_{PE} by considering the following security game between an adversary $\mathcal{A}$ and its challenger.

- **Setup**. The challenger runs the $\mathsf{PE.Setup}(\kappa, des)$ algorithm and gives the public parameters PP to the adversary.
- **Phase 1**. The adversary adaptively submits a role $\boldsymbol{y} \in \mathcal{K}$ and the challenger answers with a secret key $\mathsf{SK}_{\boldsymbol{y}}$ to the adversary.
- **Challenge**. The adversary submits two messages M_0 and M_1 of equal length and a challenge policy $\boldsymbol{x} \in \mathcal{X}$. The challenger chooses $\mu \in \{0, 1\}$ at random and encrypts M_μ under $\boldsymbol{x}$. The resulting ciphertext CT is given to the adversary.
- **Phase 2**. The adversary is allowed to continue to make queries as Phase 1.
- **Guess**. Finally, the adversary outputs a guess μ' of μ. We say that $\mathcal{A}$ is successful if none of the role $\boldsymbol{y}$ in Phases 1 & 2 that satisfies $\mathfrak{R}(\boldsymbol{x}, \boldsymbol{y}) = 1$ has been queried and $\mu' = \mu$. The success probability is defined as $Succ^{CPA}_{\mathcal{A},\Pi_{\mathsf{PE}}}(\kappa)$.

Definition 1. *For a PE scheme Π_{PE}, the advantage of an adversary $\mathcal{A}$ in the game is defined as $Adv^{CPA}_{\mathcal{A},\Pi_{\mathsf{PE}}}(\kappa) = |Succ^{CPA}_{\mathcal{A},\Pi_{\mathsf{PE}}}(\kappa) - \frac{1}{2}|$. A PE scheme Π_{PE} is secure if $Adv^{CPA}_{\mathcal{A},\Pi_{\mathsf{PE}}}(\kappa)$ is negligible with respect to the security parameter κ, for any PPT adversary $\mathcal{A}$.*

Note that a weaker model that considers selective security can be defined as with the above security game with the exception that the adversary $\mathcal{A}$ is allowed to choose the challenge encrypting policy $\boldsymbol{x}$ before the setup phase.

Variants. There exist many public key primitives that can be viewed as special cases of PE, for example, identity-based encryption (IBE) [4,9], hierarchical IBE (HIBE) [13], broadcast encryption [6], ABE [31,14], IPE [2,30], and spatial encryption (SE) [5]. We provide the definitions of ABE and IPE using the syntax of PE in the full version of this paper.

2.2 Predicate Signatures

We now define predicate signatures using the syntax of PE. In predicate signatures, the signing and verification algorithms are parameterized by a role $\boldsymbol{y}$ and a policy predicate $\boldsymbol{x}$, respectively. A predicate signature generated by a signer with role $\boldsymbol{y}$ is said to be correctly verified by the public parameters and a policy predicate $\boldsymbol{x}$ if $\mathfrak{R}(\boldsymbol{x}, \boldsymbol{y}) = 1$ holds. No other information is revealed by the signature. A predicate signature (PS) scheme $\Pi_{\mathsf{PS}} = (\mathsf{PS.Setup}, \mathsf{PS.KeyGen}, \mathsf{PS.Sign}, \mathsf{PS.Verify})$ for $\mathfrak{R}$ then consists of four probabilistic PPT algorithms that are described as follows:

- $\mathsf{PS.Setup}(\kappa, des)$: The algorithm takes a security parameter κ and a scheme description des as input. It outputs some public parameters PP and a master secret key MSK.
- $\mathsf{PS.KeyGen}(\mathsf{PP}, \mathsf{MSK}, \boldsymbol{y})$: The algorithm takes as input the public parameters PP, a master key MSK and a role $\boldsymbol{y} \in \mathcal{K}$. It returns a secret key $\mathsf{SK}_{\boldsymbol{y}}$ associated with $\boldsymbol{y}$.
- $\mathsf{PS.Sign}(\mathsf{PP}, \mathsf{SK}_{\boldsymbol{y}}, \boldsymbol{x}, M)$: The algorithm takes as input a message M, a secret key $\mathsf{SK}_{\boldsymbol{y}}$, a signing policy $\boldsymbol{x} \in \mathcal{X}$ and public parameters PP. It outputs a signature σ.
- $\mathsf{PS.Verify}(\mathsf{PP}, \boldsymbol{x}, \sigma, M)$: The algorithm takes as input a message M, a signature σ with a policy $\boldsymbol{x}$ and public parameters PP. It outputs 1 if the signature is deemed valid and 0 otherwise.

For correctness, for all $\boldsymbol{y} \in \mathcal{K}$ and $\boldsymbol{x} \in \mathcal{X}$, if $\mathfrak{R}(\boldsymbol{x}, \boldsymbol{y}) = 1$, it is required that

$$\mathsf{PS.Verify}(\mathsf{PP}, \boldsymbol{x}, \mathsf{PS.Sign}(\mathsf{PP}, \mathsf{PS.KeyGen}(\mathsf{PP}, \mathsf{MSK}, \boldsymbol{y}), \boldsymbol{x}, M), M) = 1$$

and the values $\mathsf{PP}, \mathsf{MSK}$ have been obtained by properly executing the algorithms $\mathsf{PS.Setup}$.

Security Model. We consider two essential security properties for a PS scheme: unforgeability and signer privacy.

Unforgeability: A PS scheme must provide the typical unforgeability property, even against colluding users. Let κ be a security parameter. We then define unforgeability under chosen message attacks (UF-CMA) for a PS scheme Π_{PS} by considering the following security game between an adversary $\mathcal{A}$ and its challenger:

- **Setup**. The challenger runs $\mathsf{PS.Setup}(\kappa, des)$, and sends the public parameters PP to $\mathcal{A}$.
- **Query**. $\mathcal{A}$ can make secret key and signature queries.
  - **Secret key queries**. $\mathcal{A}$ adaptively chooses a role $\boldsymbol{y} \in \mathcal{K}$ and receives the secret key $\mathsf{SK}_{\boldsymbol{y}} = \mathsf{PS.KeyGen}(\mathsf{PP}, \mathsf{MSK}, \boldsymbol{y})$ from the challenger.
  - **Signature queries**. $\mathcal{A}$ adaptively chooses a pair $(\boldsymbol{x}, M)$ consisting of a policy $\boldsymbol{x}$ and a message M. The challenger chooses a role $\boldsymbol{y}$ that $\mathfrak{R}(\boldsymbol{x}, \boldsymbol{y}) = 1$, runs $\mathsf{SK}_{\boldsymbol{y}} = \mathsf{PS.KeyGen}(\mathsf{PP}, \mathsf{MSK}, \boldsymbol{y})$ and computes a signature $\sigma = \mathsf{PS.Sign}(\mathsf{PP}, \mathsf{SK}_{\boldsymbol{y}}, \boldsymbol{x}, M)$ which is returned to $\mathcal{A}$.
- **Forgery**. At the end of the game, $\mathcal{A}$ outputs a tuple $(\boldsymbol{x}^*, M^*, \sigma^*)$. $\mathcal{A}$ is successful if:
  - $\mathcal{A}$ has not made any signature query for the pair $(\boldsymbol{x}^*, M^*)$;
  - None of the role $\boldsymbol{y}$ in secret key queries phase satisfies $\mathfrak{R}(\boldsymbol{x}^*, \boldsymbol{y}) = 1$;
  - $\mathsf{PS.Verify}(\mathsf{PP}, \boldsymbol{x}^*, \sigma^*, M^*) = 1$.

The advantage of the adversary $\mathcal{A}$ in successfully breaking the UF-CMA security of a PS scheme Π_{PS} is defined as $Succ_{\mathcal{A},\Pi_{\mathsf{PS}}}^{UF\text{-}CMA}(\kappa) = Pr[\mathcal{A}\ wins]$.

Definition 2. *A PS scheme Π_{PS} is UF-CMA if $Succ^{UF\text{-}CMA}_{\mathcal{A},\Pi_{\mathsf{PS}}}(\kappa)$ is negligible with respect to the security parameter κ, for any PPT adversary $\mathcal{A}$.*

Similarly, if the adversary $\mathcal{A}$ is allowed to choose the challenge signing policy $\boldsymbol{x}$ before the setup phase, we then have a weaker model called selective unforgeability.

Perfect Privacy: This property is required to achieve anonymous ABS in the sense that PS signatures reveal no information except that the role information that has been used to generate the signatures. Perfect privacy must hold even against an unbounded adversary which has knowledge of the master secret key.

Definition 3. *A PS scheme Π_{PS} is perfectly private, if for any message M, any two roles $\boldsymbol{y}_1, \boldsymbol{y}_2$, any secret keys $\mathsf{SK}_1 = \mathsf{PS.KeyGen}(\mathsf{PP}, \mathsf{MSK}, \boldsymbol{y}_1), \mathsf{SK}_2 = \mathsf{PS.KeyGen}(\mathsf{PP}, \mathsf{MSK}, \boldsymbol{y}_2)$, and any policy $\boldsymbol{x}$ such that $\mathfrak{R}(\boldsymbol{x}, \boldsymbol{y}_1) = 1$ and $\mathfrak{R}(\boldsymbol{x}, \boldsymbol{y}_2) = 1$, the distribution of $\mathsf{PS.Sign}(\mathsf{PP}, \mathsf{SK}_1, \boldsymbol{x}, M)$ is identical to that of $\mathsf{PS.Sign}(\mathsf{PP}, \mathsf{SK}_2, \boldsymbol{x}, M)$.*

Identity-based signatures (IBS) [17], identity-based ring signatures (IBRS) [37], and ABS [12,15,23,24,29] are example of special cases of PS. Moreover, as we define below, the notion of inner-product signatures (IPS) is also a variant of PS.

Inner-Product Signatures. The notion of inner-product signatures (IPS) can be defined as with PS, except with the following modification:

– The setup algorithm defines a positive integer N and a dimension n;
– The role space $\mathcal{K} := \{\boldsymbol{v} := (v_1, \ldots, v_n) \in \mathbb{Z}_N^n\}$;
– The policy space $\mathcal{X} := \{\boldsymbol{x} := (x_1, \ldots, x_n) \in \mathbb{Z}_N^n\}$;
– The predicate $\mathfrak{R} : \mathcal{K} \times \mathcal{X} \to \{0, 1\}$ is defined as

$$\mathfrak{R}(\boldsymbol{v}, \boldsymbol{x}) := \begin{cases} 1 & \text{if } \langle \boldsymbol{v}, \boldsymbol{x} \rangle = 0 \\ 0 & \text{otherwise.} \end{cases}$$

The detailed description of ABS can refer to [24,29]. In this paper, we are mainly concerned with the notions of ABS and IPS.

3 Generic Constructions

We describe transformation from inner-product systems to attribute-based systems supporting threshold access structures. We first recall the definition of an access structure.

Definition 4. *Let $\mathbb{U} = \{att_1, att_2, \ldots, att_n\}$ be a set of attributes. An access structure is a set collection $\mathbb{A} \subseteq 2^{\{att_1, att_2, \ldots, att_n\}} \backslash \emptyset$. An access structure is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

We use (Ω, t) to denote a threshold access structure Γ in $\mathbb{U}$ if there exist a threshold t and a subset $\Omega \subseteq \mathbb{U}$ such that $S \in \Gamma \Leftrightarrow |S \cap \Omega| \geq t$. When the access structures are restricted to the threshold setting, we call it threshold CP-ABE (tCP-ABE), threshold KP-ABE (tKP-ABE) and threshold ABS (tABS).

3.1 Generic Construction of tKP-ABE from IPE

To construct a tKP-ABE scheme over an attribute universe $\mathbb{U} := \{att_1, \ldots, att_n\}$, we require an $(n+1)$-dimensional IPE scheme. Given an IPE scheme Π_{IPE} with four algorithms: $(\mathsf{IPE.Setup}, \mathsf{IPE.KeyGen}, \mathsf{IPE.Enc}, \mathsf{IPE.Dec})$, we construct a tKP-ABE scheme Π_{tKP} with the corresponding four algorithms: $(\mathsf{tKP.Setup}, \mathsf{tKP.KeyGen}, \mathsf{tKP.Enc}, \mathsf{tKP.Dec})$ as follows:

- $\mathsf{tKP.Setup}(\kappa, \mathbb{U})$: It runs $\mathsf{IPE.Setup}(\kappa, n+1)$ and outputs public parameters PP and a master key MSK.
- $\mathsf{tKP.Enc}(\mathsf{PP}, S, M)$: For a subset $S \subseteq \mathbb{U}$, it first computes a vector $\boldsymbol{x} := (x_1, \ldots, x_{n+1})$ as follows:
$$x_1 := -1, \quad \text{for } 1 \leq i \leq n : x_{i+1} := \begin{cases} 1 & \text{if } att_i \in S \\ 0 & \text{otherwise} \end{cases}$$
Then runs $\mathsf{IPE.Enc}(\mathsf{PP}, \boldsymbol{x}, M)$ and outputs a ciphertext CT.
- $\mathsf{tKP.KeyGen}(\mathsf{PP}, \Gamma := (\Omega, t), \mathsf{MSK})$: For a threshold access structure (Ω, t) (where we denote $m := |\Omega|$), it computes a vector $\boldsymbol{v} := (v_1, \ldots, v_n)$ as follows:
$$\text{for } 1 \leq i \leq n : v_i := \begin{cases} 1 & \text{if } att_i \in \Omega \\ 0 & \text{otherwise} \end{cases}$$
Then for $1 \leq j \leq m - t + 1$ it runs $\mathsf{IPE.SK}_i := \mathsf{IPE.KeyGen}(\mathsf{PP}, \boldsymbol{v}_j, \mathsf{MSK})$, where $\boldsymbol{v}_j := (t + j - 1, v_1, \ldots, v_n)$. Outputs the secret key $\mathsf{KP.SK}_{(\Omega,t)} := \{\mathsf{IPE.SK}_j\}_{1 \leq j \leq m-t+1}$.
- $\mathsf{tKP.Dec}(\mathsf{PP}, \mathsf{CT}, S, \mathsf{KP.SK}_{(\Omega,t)})$: For a ciphertext CT with the subset S and a secret key parsed as $\mathsf{KP.SK}_{(\Omega,t)} := \{\mathsf{IPE.SK}_1, \ldots, \mathsf{IPE.SK}_{m-t+1}\}$, if $k := |S \cap \Omega| \geq t$, it runs $\mathsf{IPE.Dec}(\mathsf{PP}, \boldsymbol{x}, \mathsf{CT}, \mathsf{IPE.SK}_{k-t+1})$ and outputs the message M, where $\boldsymbol{x} := (x_1, \ldots, x_{n+1})$:
$$x_1 := -1, \quad \text{for } 1 \leq i \leq n : x_{i+1} := \begin{cases} 1 & \text{if } att_i \in S \\ 0 & \text{otherwise} \end{cases}$$

Correctness. For the vector $\boldsymbol{x} := (x_1, \ldots, x_{n+1})$ corresponding to the subset S in the ciphertext and the vector $\boldsymbol{v}_{k-t+1} := (v_1, \ldots, v_{n+1})$ corresponding to the secret key component $\mathsf{IPE.SK}_{k-t+1}$ in the tKP-ABE, we have
$$x_1 \cdot v_1 = -k, \quad \sum_{i=2}^{n+1} x_i \cdot v_i = k$$

So we have $\langle \boldsymbol{x}, \boldsymbol{v}_{k-t+1} \rangle = 0$. This implies that the resulting tKP-ABE scheme inherits the decryptability from the underlying IPE scheme, i.e.,

$$\mathsf{IPE.Dec}(\mathsf{PP}, \boldsymbol{x}, \mathsf{IPE.Enc}(\mathsf{PP}, \boldsymbol{x}, M), \mathsf{IPE.KeyGen}(\mathsf{PP}, \boldsymbol{v}, \mathsf{MSK})) = M$$

iff $\langle \boldsymbol{x}, \boldsymbol{v} \rangle = 0$.

Theorem 1. *The resulting tKP-ABE scheme is (selectively) secure if the underlying IPE is (selectively) secure.*

3.2 Generic Construction of tCP-ABE from IPE

To construct a tCP-ABE scheme over an attribute universe $\mathbb{U} := \{att_1, \ldots, att_n\}$, we require an $(n+2)$-dimensional IPE scheme. Given an IPE scheme Π_{IPE} with four algorithms: $(\mathsf{IPE.Setup}, \mathsf{IPE.KeyGen}, \mathsf{IPE.Enc}, \mathsf{IPE.Dec})$, we construct a tCP-ABE scheme Π_{tCP} with the corresponding four algorithms: $(\mathsf{tCP.Setup}, \mathsf{tCP.KeyGen}, \mathsf{tCP.Enc}, \mathsf{tCP.Dec})$ as follows:

- $\mathsf{tCP.Setup}(\kappa, \mathbb{U})$: It runs $\mathsf{IPE.Setup}(\kappa, n+2)$ and outputs public parameters PP and a master key MSK.
- $\mathsf{tCP.Enc}(\mathsf{PP}, \Gamma := (\Omega, t), M)$: For a threshold access structure (Ω, t), it computes a vector $\boldsymbol{x} := (x_1, \ldots, x_{n+2})$ as follows:

$$x_1 := -t, \quad \text{for } 1 \le i \le n : x_{i+1} := \begin{cases} 1 & \text{if } att_i \in \Omega \\ 0 & \text{otherwise} \end{cases}, \quad x_{n+2} := 1$$

  Then runs $\mathsf{IPE.Enc}(\mathsf{PP}, \boldsymbol{x}, M)$ and outputs a ciphertext CT.
- $\mathsf{tCP.KeyGen}(\mathsf{PP}, S, \mathsf{MSK})$: For a subset $S \subseteq \mathbb{U}$, it first computes a vector $\boldsymbol{v} := (v_1, \ldots, v_n)$ as follows:

$$\text{for } 1 \le i \le n : v_i := \begin{cases} 1 & \text{if } att_i \in S \\ 0 & \text{otherwise} \end{cases}$$

  Then for $1 \le i \le |S| - 1$ it runs $\mathsf{IPE.SK}_i := \mathsf{IPE.KeyGen}(\mathsf{PP}, \boldsymbol{v}_i, \mathsf{MSK})$, where $\boldsymbol{v}_i := (1, v_1, \ldots, v_n, 1 - i)$. Outputs the secret key $\mathsf{CP.SK}_S := \{\mathsf{IPE.SK}_i\}_{1 \le i \le |S|-1}$.
- $\mathsf{tCP.Dec}(\mathsf{PP}, \mathsf{CT}, \Gamma := (\Omega, t), \mathsf{CP.SK}_S)$: For a ciphertext CT with the threshold (Ω, t) and a secret key parsed as $\mathsf{KP.SK}_S := \{\mathsf{IPE.SK}_i\}_{1 \le i \le |S|-1}$, if $k := |S \cap \Omega| \ge t$, it runs $\mathsf{IPE.Dec}(\mathsf{PP}, \boldsymbol{x}, \mathsf{CT}, \mathsf{IPE.SK}_{k-t+1})$ and outputs the message M, where $\boldsymbol{x} := (x_1, \ldots, x_{n+2})$:

$$x_1 := -t, \quad \text{for } 1 \le i \le n : x_{i+1} := \begin{cases} 1 & \text{if } att_i \in \Omega \\ 0 & \text{otherwise} \end{cases}, \quad x_{n+2} := 1$$

Correctness. The security reduction can follow closely with that of the generic construction for the above tKP-ABE and we will not discuss any further here.

Theorem 2. *The resulting tCP-ABE scheme is (selectively) secure if the underlying IPE is (selectively) secure.*

3.3 Generic Construction of tABS from IPS

To construct a tABS scheme over an attribute universe $\mathbb{U} := \{att_1, \dots, att_n\}$, we require an $(n+2)$-dimensional IPS scheme. Given an IPS scheme Π_{IPS} with four algorithms: $(\mathsf{IPS.Setup}, \mathsf{IPS.KeyGen}, \mathsf{IPS.Sign}, \mathsf{IPS.Verify})$, we construct a tABS scheme Π_{tABS} with the corresponding four algorithms: $(\mathsf{tABS.Setup}, \mathsf{tABS.KeyGen}, \mathsf{tABS.Sign}, \mathsf{tABS.Verify})$ as follows:

– $\mathsf{tABS.Setup}(\kappa, \mathbb{U})$: It runs $\mathsf{IPS.Setup}(\kappa, n+2)$ and outputs public parameters PP and a master key MSK.

– $\mathsf{tABS.KeyGen}(\mathsf{PP}, S, \mathsf{MSK})$: For a subset $S \subseteq \mathbb{U}$, it first computes a vector $\boldsymbol{v} := (v_1, \dots, v_n)$ as follows:

$$\text{for } 1 \le i \le n : v_i := \begin{cases} 1 & \text{if } att_i \in S \\ 0 & \text{otherwise} \end{cases}$$

Then for $1 \le i \le |S| - 1$ it runs $\mathsf{IPS.SK}_i := \mathsf{IPS.KeyGen}(\mathsf{PP}, \boldsymbol{v}_i, \mathsf{MSK})$, where $\boldsymbol{v}_i := (1, v_1, \dots, v_n, 1 - i)$. Outputs the secret key $\mathsf{ABS.SK}_S := \{\mathsf{IPS.SK}_i\}_{1 \le i \le |S|-1}$.

– $\mathsf{tABS.Sign}(\mathsf{PP}, \mathsf{ABS.SK}_S, \Gamma := (\Omega, t), M)$: If $k := |S \cap \Omega| \ge t$, it runs $\sigma \leftarrow \mathsf{IPS.Sign}(\mathsf{PP}, \boldsymbol{x}, \mathsf{IPS.SK}_{k-t+1}, \Gamma || M)$, where $\boldsymbol{x} := (x_1, \dots, x_{n+2})$ as follows:

$$x_1 := -t, \quad \text{for } 1 \le i \le n : x_{i+1} := \begin{cases} 1 & \text{if } att_i \in \Omega \\ 0 & \text{otherwise} \end{cases}, \quad x_{n+2} := 1$$

And it outputs σ as the signature.

– $\mathsf{tABS.Verify}(\mathsf{PP}, \sigma, \Gamma := (\Omega, t), M)$: It runs $\mathsf{IPS.Verify}(\mathsf{PP}, \boldsymbol{x}, \sigma, \Gamma || M)$ where $\boldsymbol{x} := (x_1, \dots, x_{n+2})$ as follows:

$$x_1 := -t, \quad \text{for } 1 \le i \le n : x_{i+1} := \begin{cases} 1 & \text{if } att_i \in \Omega \\ 0 & \text{otherwise} \end{cases}, \quad x_{n+2} := 1$$

And outputs the result.

Correctness. The deduction can follow closely with that of the generic construction of tKP-ABE above. And we omit it here.

Theorem 3. *The resulting tABS scheme is (selectively) unforgeable and perfectly private if the underlying IPS is (selectively) unforgeable and perfectly private.*

The security proofs of Theorems 1, 2 & 3 can be easily obtained from the definitions of ABE/ABS and IPE/IPS. Due to space constraints, we omit them here.

4 Concrete Constructions of tABE and tABS

We are now ready to describe how to construct a threshold attribute-based system from an inner-product system. For the space considersion, we give concrete constructions of IPE/IPS which are tailored to our needs in the full version of this paper. Making use of our IPE/IPS schemes as building blocks, we propose constructions for fully secure tKP-ABE and tCP-ABE with constant-size ciphertexts, as well as fully secure and perfectly private tABS with constant-size signatures. Due to the space limitation, we only give the instances of tKP-ABE and ABS. The tCP-ABE scheme can be easily obtained from the tKP-ABE scheme and we omit it here. The correctness and security of our schemes follows from the generic conversions. Our schemes are for small universes of attributes $\mathbb{U} := \{att_1, \ldots, att_n\}$ and based on composite order groups. (The definition of composite order bilinear groups can be found in the full version of this paper or [22].)

Composite Order Bilinear Groups. We define composite order bilinear groups as follows: let $\mathcal{G}_c$ be a group generator which outputs $\mathcal{I} := (N = p_1p_2p_3, G, G_T, e)$ where p_1, p_2, p_3 are distinct primes, G and G_T are cyclic groups of order $N = p_1p_2p_3$, and e is a bilinear map, $e : G \times G \to G_T$ such that $e(g, g) \neq 1$ for g and for any $u, v \in \mathbb{Z}_N$, it holds that $e(g^u, g^v) = e(g, g)^{uv}$. We say that G is a bilinear group if the group operation in G and the bilinear map $e : G \times G \to G_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^u, g^v) = e(g, g)^{uv} = e(g^v, g^u)$. We let $G_{p_1}, G_{p_2}, G_{p_3}$ denote the subgroups of order p_1, p_2, p_3 in G, respectively. Furthermore, for $a, b, c \in \{1, p_1, p_2, p_3\}$ we denote by G_{abc} the subgroup of order abc. From the fact that the group is cyclic it is simple to verify that if g and h are group elements of different order (and thus belonging to different subgroups), then $e(g, h) = 1$. This is called the orthogonality property and is a crucial tool in our constructions.

4.1 Fully Secure tKP-ABE with Constant-Size Ciphertexts

– $\mathsf{tKP.Setup}(\kappa, \mathbb{U} := \{att_1, \ldots, att_n\})$: The setup algorithm chooses a random description $\mathcal{I} := (N = p_1p_2p_3, G, G_T, e)$ with $G = G_{p_1} \times G_{p_2} \times G_{p_3}$. It then randomly picks $\alpha, a_0, \ldots, a_{n+1} \in \mathbb{Z}_N$ and $X_3 \in G_{p_3}$. It then sets $\boldsymbol{h} := (h_0, \ldots, h_{n+1}) = (g^{a_0}, g^{a_1}, \ldots, g^{a_{n+1}})$. It outputs the public parameters and master key as $\mathsf{PP} := (\mathcal{I}, g, \boldsymbol{h}, e(g, g)^\alpha)$, $\quad \mathsf{MSK} := (\alpha, X_3)$, respectively.
– $\mathsf{tKP.KeyGen}(\mathsf{PP}, \varGamma := (\varOmega, t), \mathsf{MSK})$: For a threshold access structure $(\varOmega, t)$ (where we let $m := |\varOmega|$), the algorithm computes a vector $\boldsymbol{v} := (v_1, \ldots, v_n)$ as follows:

$$\text{for } 1 \leq i \leq n : v_i := \begin{cases} 1 & \text{if } att_i \in \varOmega \\ 0 & \text{otherwise} \end{cases}.$$

Then for $1 \leq i \leq m - t + 1$, the algorithm randomly picks $r_i \in \mathbb{Z}_N$ and $(R_{0,i}, \ldots, R_{n+1,i}) \in G_{p_3}^{n+2}$, and outputs the secret key element $\mathsf{SK}_i := (K_{0,i}, K_{1,i}, \ldots, K_{n+1,i})$ by setting

$$K_{0,i} := g^{r_i} R_{0,i}, \quad K_{1,i} := g^{\alpha} h_0^{r_i} R_{1,i}, \quad \left\{ K_{j,i} := \left(h_1^{-\frac{v_{j,i}}{v_{1,i}}} h_j \right)^{r_i} R_{j,i} \right\}_{j=2,\ldots,n+1},$$

where $\boldsymbol{v}_i := (v_{1,i}, \ldots, v_{n+1,i}) = (t + i - 1, v_1, \ldots, v_n)$. It also outputs the secret key $\mathsf{KP.SK}_{(\Omega,t)} := \{\mathsf{SK}_i\}_{1 \le i \le m-t+1}$.

– $\mathsf{tKP.Enc}(\mathsf{PP}, S, M)$: For a subset $S \subseteq \mathbb{U}$ and a message $M \in G_T$ to encrypt, the algorithm first computes a vector $\boldsymbol{x} := (x_1, \ldots, x_{n+1})$ as follows:

$$x_1 := -1, \quad \text{for } 1 \le i \le n : x_{i+1} := \begin{cases} 1 & \text{if } att_i \in S \\ 0 & \text{otherwise} \end{cases}.$$

It then randomly picks $s \in \mathbb{Z}_N$ and computes the ciphertext $\mathsf{CT} := (C, C_0, C_1)$ as $C := M \cdot e(g,g)^{\alpha s}, \quad C_0 := g^s, \quad C_1 := \left(h_0 \prod_{j=1}^{n+1} h_j^{x_j} \right)^s$.

– $\mathsf{tKP.Dec}(\mathsf{PP}, \mathsf{CT}, S, \mathsf{KP.SK}_S)$: For a ciphertext CT parsed as (C, C_0, C_1) with the subset S and a secret key $\mathsf{KP.SK}_{(\Omega,t)}$ parsed as $\{\mathsf{SK}_1, \ldots, \mathsf{SK}_{m-t+1}\}$, if $k := |S \cap \Omega| \ge t$, the algorithm first computes $\boldsymbol{x} := (x_1, \ldots, x_{n+1})$:

$$x_1 := -1, \quad \text{for } 1 \le i \le n : x_{i+1} := \begin{cases} 1 & \text{if } att_i \in S \\ 0 & \text{otherwise} \end{cases}.$$

It then uses $\mathsf{SK}_{k-t+1} := (K_0, K_1, \ldots, K_{n+1})$ to decrypt: $e(g,g)^{\alpha s} = e(C_0, K_1 \prod_{j=2}^{n+1} K_j^{x_j}) / e(C_1, K_0)$ and recovers the message as $M := C/e(g,g)^{\alpha s}$.

4.2 Fully Secure tABS with Constant-Size Signatures

– $\mathsf{tABS.Setup}(\kappa, \mathbb{U} := \{att_1, \ldots, att_n\})$: The setup algorithm chooses a random description $\mathcal{I} := (N = p_1 p_2 p_3, G, G_T, e)$ with $G = G_{p_1} \times G_{p_2} \times G_{p_3}$. It then randomly picks $\alpha, a_0, \ldots, a_{n+2}, b_0, b_1, b_2 \in \mathbb{Z}_N$, $X_3 \in G_{p_3}$ and a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_N$. The algorithm sets $\boldsymbol{h} := (h_0, \ldots, h_{n+2}) = (g^{a_0}, g^{a_1}, \ldots, g^{a_{n+2}})$, and outputs the public parameters and master key as $\mathsf{PP} := (\mathcal{I}, g, \boldsymbol{h}, g^{b_0}, g^{b_1}, g^{b_2}, X_3, e(g,g)^{\alpha}), \quad \mathsf{MSK} := (\alpha)$.

– $\mathsf{tABS.KeyGen}(\mathsf{PP}, S, \mathsf{MSK})$: For a subset $S \subseteq \mathbb{U}$, the algorithm first computes a vector $\boldsymbol{v} := (v_1, \ldots, v_n)$ as follows:

$$\text{for } 1 \le i \le n : v_i := \begin{cases} 1 & \text{if } att_i \in S \\ 0 & \text{otherwise} \end{cases}.$$

Then for $1 \le i \le |S| - 1$, the algorithm randomly picks $r_i \in \mathbb{Z}_N$ and $(R_{0,i}, \ldots, R_{n+2,i}) \in G_{p_3}^{n+3}$, and outputs the secret key element $\mathsf{SK}_i := (K_{0,i}, K_{1,i},$
$\ldots, K_{n+2,i})$ by setting

$$K_{0,i} := g^{r_i} R_{0,i}, \quad K_{1,i} := g^{\alpha} h_0^{r_i} R_{1,i}, \quad \left\{ K_{j,i} := \left(h_1^{-\frac{v_{j,i}}{v_{1,i}}} h_j \right)^{r_i} R_{j,i} \right\}_{j=2,\ldots,n+2},$$

where $\boldsymbol{v}_i := (v_{1,i}, \ldots, v_{n+2,i}) = (1, v_1, \ldots, v_n, 1 - i)$. The algorithm also outputs the secret key $\mathsf{ABS.SK}_S := \{\mathsf{SK}_i\}_{1 \le i \le |S|-1}$.

– $\mathsf{tABS.Sign}(\mathsf{PP}, \mathsf{ABS.SK}_S, \Gamma := (\Omega, t), M)$: To sign a message M with a threshold access structure (Ω, t) with a secret key $\mathsf{ABS.SK}_S$ parsed as $\{\mathsf{SK}_i\}_{1 \le i \le |S|-1}$, if $k := |S \cap \Omega| \ge t$, the algorithm uses $\mathsf{SK}_{k-t+1} := (K_0, K_1, \ldots, K_{n+2})$. It first computes $\boldsymbol{v} := (v_1, \ldots, v_{n+2})$ and $\boldsymbol{x} := (x_1, \ldots, x_{n+2})$ as follows:

$$v_1 := 1, \quad \text{for } 1 \le i \le n : v_{i+1} := \begin{cases} 1 & \text{if } att_i \in S \\ 0 & \text{otherwise} \end{cases}, \quad v_{n+2} := k - t;$$

$$x_1 := -t, \quad \text{for } 1 \le i \le n : x_{i+1} := \begin{cases} 1 & \text{if } att_i \in \Omega \\ 0 & \text{otherwise} \end{cases}, \quad x_{n+2} := 1.$$

The algorithm then randomly picks $r_1, r_2, \alpha' \in \mathbb{Z}_N$ and $R_0, \ldots, R_{n+2}, R'_1, R'_2, R'_3 \in G_{p_3}$, and computes

$$K'_0 := K_0 \cdot g^{r_1} R_0, \quad K'_1 := K_1 \cdot g^{\alpha'} h_0^{r_1} R_1, \quad \left\{ K'_i := K_i \cdot (h_1^{-\frac{v_i}{v_1}} h_i)^{r_1} R_i \right\}_{i=2,\ldots,n+2},$$
$$K'_{n+1} := g^{r_2} R'_1, \quad K'_{n+2} := g^{-\alpha'} g^{r_2 b_0} R'_2, \quad K'_{n+3} := ((g^{b_1})^{H(M||\Gamma,\boldsymbol{x})} g^{b_2})^{r_2} R'_3.$$

It outputs the signature $\sigma := (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ by setting

$$\sigma_1 := K'_1 \cdot \prod_{i=2}^{n} (K'_i)^{x_i}, \quad \sigma_2 := K'_0, \quad \sigma_3 := K'_{n+2} \cdot K'_{n+3}, \quad \sigma_4 := K'_{n+1}.$$

– $\mathsf{tABS.Verify}(\mathsf{PP}, \sigma, \Gamma := (\Omega, t), M)$: On input a signature σ parsed as $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ and a threshold (Ω, t), the algorithm computes $\boldsymbol{x} := (x_1, \ldots, x_{n+2})$, where

$$x_1 := -t, \quad \text{for } 1 \le i \le n : x_{i+1} := \begin{cases} 1 & \text{if } att_i \in \Omega \\ 0 & \text{otherwise} \end{cases}, \quad x_{n+2} := 1.$$

The algorithm outputs 1 if

$$e(g, g)^{\alpha} = \frac{e(g, \sigma_1 \cdot \sigma_3)}{e\big(h_0 \prod_{i=1}^{n+2} h_i^{x_i}, \sigma_2\big) \cdot e\big(g^{b_0} (g^{b_1})^{H(M||\Gamma,\boldsymbol{x})} g^{b_2}, \sigma_4\big)}.$$

Otherwise, it outputs 0.

5 Extensions

5.1 Constructions in Prime Order Groups

Using groups of prime order can potentially lead to more efficient systems (via faster group operations) and security under different assumptions. A natural problem is how to construct the prime order group variants of our systems. This depends on the constructions of the underlying IPE/IPS. For IPE, [2,30] gave two fully secure IPE schemes with constant-size ciphertexts in prime order groups. For IPS, we present a fully secure and perfectly private IPS scheme with constant-size signatures in the prime order groups based on the HIPE (Hierarchical Inner-product Encryption) scheme of [30]. The detail of the construction is given in the full version of this paper. With that, we can make use of the above constructions to obtain the desired attribute-based schemes in the prime order groups.

5.2 Large Universe Constructions

Our constructions in Section 4 are limited to the small-universe case where the set of attributes $\mathbb{U}$ is defined at system setup and the size of the public parameters grows with $|\mathbb{U}|$. We now show how to extend them to the large universe setting where the number of attributes is unlimited and the public parameter size is constant. In the random oracle model, it is easy to overcome the dimension-limitation and achieve a "large-dimension" in the inner-product systems.

We now turn to realizing a large universe constructions in the standard model. From the concrete constructions presented in Section 4, we can apply the tricks used for the large universe constructions in [31,35]. As in the random oracle model, we "program" each coordinate parameter element by using a hash function that has enough degrees of randomness to plug in the same information. The tradeoff is that we need to define the maximum number of attributes max that any one key may have in the setup phase. Moreover, the public parameters grow linearly with max. We stress that this does not limit the number of attributes that may be used in the system. We realize construction in the standard model by adapting the construction of tKP-ABE in the full version of this paper. We remark that similar techniques can be used to realize large universe variant of our ABE/ABS constructions based on composite groups and ABE constructions based on the prime order groups converted from [2] in the standard model, although we do not provide the details in this paper.

5.3 More General Access Structures

Our generic conversions can be extended to admit weighted threshold access structures which are more general than threshold. We use $\varGamma := (\varOmega, \omega, t)$ to denote a weighted threshold access structure [3] over $\mathbb{U}$ if there exist a threshold t and an assignment of weights $\omega : \mathbb{U} \to \mathbb{Z}_N$ such that $S \in \varGamma \Leftrightarrow \varSigma_{att \in S} \omega(att) \geq t$. We can make our generic conversion support weighted threshold access structure by a slight modification. For a weighted threshold access structure $\varGamma := (\varOmega, \omega, t)$, we set the vector $\boldsymbol{x} := (x_1, \ldots, x_n)$ as

$$\text{for } 1 \leq i \leq n : x_i := \begin{cases} \omega(att_i) & \text{if } att_i \in \varOmega \\ 0 & \text{otherwise} \end{cases}$$

and the vector $\boldsymbol{v} := (v_1, \ldots, v_n)$ expressing subset $S \subseteq \mathbb{U}$ is unchange.

We can compute the sum of weights of the attributes in S by computing the inner-product of the two vectors. This way, we can realize the weighted threshold access structures.

Acknowledgment. The work is supported by the National Basic Research Program of China (No. 2013CB338003), and the National Natural Science Foundation of China (No.61170278, 91118006).

References

1. Attrapadung, N., Libert, B., de Panafieu, E.: Expressive Key-Policy Attribute-Based Encryption with Constant-Size Ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 90–108. Springer, Heidelberg (2011)
2. Attrapadung, N., Libert, B.: Functional Encryption for Inner Product: Achieving Constant-Size Ciphertexts with Adaptive Security or Support for Negation. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 384–402. Springer, Heidelberg (2010)
3. Beimel, A., Tassa, T., Weinreb, E.: Characterizing Ideal Weighted Threshold Secret Sharing. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 600–619. Springer, Heidelberg (2005)
4. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
5. Boneh, D., Hamburg, M.: Generalized Identity Based and Broadcast Encryption Schemes. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 455–470. Springer, Heidelberg (2008)
6. 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)
7. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
8. Boneh, D., Sahai, A., Waters, B.: Functional Encryption: Definitions and Challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
9. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003, vol. 2656, pp. 254–271. Springer, Heidelberg (2003)
10. Chen, C., Zhang, Z., Feng, D.: Efficient Ciphertext Policy Attribute-Based Encryption with Constant-Size Ciphertext and Constant Computation-Cost. In: Boyen, X., Chen, X. (eds.) ProvSec 2011. LNCS, vol. 6980, pp. 84–101. Springer, Heidelberg (2011)
11. Emura, K., Miyaji, A., Nomura, A., Omote, K., Soshi, M.: A Ciphertext-Policy Attribute-Based Encryption Scheme with Constant Ciphertext Length. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 13–23. Springer, Heidelberg (2009)

12. Escala, A., Herranz, J., Morillo, P.: Revocable Attribute-Based Signatures with Adaptive Security in the Standard Model. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 224–241. Springer, Heidelberg (2011)
13. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
14. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS 2006, pp. 89–98. ACM Press (2006)
15. Herranz, J., Laguillaumie, F., Libert, B., Ràfols, C.: Short Attribute-Based Signatures for Threshold Predicates. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 51–67. Springer, Heidelberg (2012)
16. Herranz, J., Laguillaumie, F., Ràfols, C.: Constant Size Ciphertexts in Threshold Attribute-Based Encryption. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 19–34. Springer, Heidelberg (2010)
17. Hess, F.: Efficient Identity Based Signature Schemes Based on Pairings. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 310–324. Springer, Heidelberg (2003)
18. Ge, A., Zhang, R., Chen, C., Ma, C., Zhang, Z.: Threshold Ciphertext Policy Attribute-Based Encryption with Constant Size Ciphertexts. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 336–349. Springer, Heidelberg (2012)
19. Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
20. Lewko, A., Waters, B.: New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
21. Lewko, A.: Tools for Simulating Features of Composite Order Bilinear Groups in the Prime Order Setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012)
22. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
23. Li, J., Au, M.H., Susilo, W., Xie, D., Ren, K.: Attribute-based signature and its applications. In: Asiaccs 2010, pp. 60–69. ACM Press, New York (2010)
24. 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)
25. Mordini, E., Massari, S.: Body, biometrics and identity. Bioethics 22(9), 488–498 (2008)
26. Nagar, A., Rane, S., Vetro, A.: Alignment and bit extraction for secure fingerprint biometrics. In: SPIE Conference on Electronic Imaging (2010)
27. Nandakumar, K., Jain, A.: Multibiometric Template Security Using Fuzzy Vault. In: International Conference on Biometrics: Theory, Applications and Systems, pp. 1–6 (2008)
28. Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)

29. Okamoto, T., Takashima, K.: Efficient Attribute-Based Signatures for Non-monotone Predicates in the Standard Model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 35–52. Springer, Heidelberg (2011)
30. Okamoto, T., Takashima, K.: Achieving Short Ciphertexts or Short Secret-Keys for Adaptively Secure General Inner-Product Encryption. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 138–159. Springer, Heidelberg (2011)
31. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
32. Shamir, A.: How to share a secret. Communications. ACM 22(11), 612–613 (1979)
33. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
34. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
35. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)
36. Yang, P., Cao, Z., Dong, X.: Fuzzy Identity Based Signature. Cryptology ePrint Archive, Report 2008/002 (2008), http://eprint.iacr.org/
37. Zhang, F., Kim, K.: ID-Based Blind Signature and Ring Signature from Pairings. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 533–547. Springer, Heidelberg (2002)

A Robust and Plaintext-Aware Variant of Signed ElGamal Encryption

Yannick Seurin and Joana Treger

ANSSI, Paris, France
yannick.seurin@m4x.org, joana.marim@ssi.gouv.fr

Abstract. Adding a Schnorr signature to ElGamal encryption is a popular proposal aiming at thwarting chosen-ciphertext attacks by rendering the scheme *plaintext-aware*. However, there is no known security proof for the resulting scheme, at least not in a weaker model than the one obtained by combining the Random Oracle Model (ROM) and the Generic Group Model (Schnorr and Jakobsson, ASIACRYPT 2000). In this paper, we propose a very simple modification to Schnorr-Signed ElGamal encryption that leaves keys and ciphertexts size unchanged, for which the resulting scheme is semantically secure under adaptive chosen-ciphertext attacks (IND-CCA2-secure) in the ROM under the Decisional Diffie-Hellman assumption. In fact, we even prove that our new scheme is plaintext-aware in the ROM as defined by Bellare *et al.* (CRYPTO '98). Interestingly, we also observe that Schnorr-Signed ElGamal is *not* plaintext-aware (again, for the definition of Bellare *et al.*) under the Computational Diffie-Hellman assumption. We show that our new scheme additionally achieves anonymity as well as robustness, a notion formalized by Abdalla *et al.* (TCC 2010) which captures the fact that it is hard to create a ciphertext that is valid under two different public keys. Finally, we study the hybrid variant of our new proposal, and show that it is IND-CCA2-secure in the ROM under the Computational Diffie-Hellman assumption when used with a symmetric encryption scheme satisfying the weakest security notion, namely ciphertext indistinguishability under one-time attacks (IND-OT-security).

1 Introduction

ElGamal and Variants. The ElGamal encryption scheme [18] is one of the oldest discrete-log based public key encryption scheme. It works as follows. Given a cyclic group $\mathbb{G}$ of prime order p and a generator G, a secret/public key pair is a pair $(x, X = G^x)$, where x is randomly drawn in $\mathbb{Z}_p^*$. To encrypt a message $M \in \mathbb{G}$, one draws a random integer $r \in \mathbb{Z}_p^*$, and computes $R = G^r$ and $Y = MX^r$. The ciphertext is (Y, R), and can be decrypted by computing Y/R^x.

ElGamal encryption is one-way under the Computational Diffie-Hellman (CDH) assumption, and its semantic security against chosen-plaintext attacks (IND-CPA-security) is equivalent to the Decisional Diffie-Hellman (DDH) assumption [42]. However, it is not secure against adaptive[1] chosen-ciphertext attacks since it is malleable: given a ciphertext (Y, R) corresponding to plaintext M, one can easily generate a ciphertext corresponding to MG^z as (YG^z, R). This implies that the scheme cannot be IND-CCA2-secure.

There have been mainly two approaches aiming at enhancing the security of ElGamal encryption to resist adaptive chosen-ciphertext attacks in the Random Oracle Model (ROM). The first one is to use the so-called *hybrid* (or *hashed*) variant of the scheme, where X^r is hashed through a random oracle $\boldsymbol{H_K}$ to give a secret key K subsequently used in a symmetric encryption scheme. The resulting scheme is often referred to as DHIES [4,15,28]. To establish the IND-CCA2-security of hybrid ElGamal encryption, one has to rely on the IND-CCA2-security of the symmetric encryption scheme[2] and on the non-standard Strong Diffie-Hellman (SDH) assumption, which states that the CDH problem remains hard even given access to an oracle solving the DDH problem.[3] Recently, Cash *et al.* [12] removed the need to rely on the SDH assumption by proposing the Twin ElGamal encryption scheme, a variant of hybrid ElGamal which is IND-CCA2-secure under the CDH assumption.

The second approach is to add a non-interactive proof of knowledge of the random integer r used to encrypt the plaintext, with the hope to make the scheme *plaintext-aware*. The notion of plaintext awareness was introduced in [9] and captures the intuitive idea that it should be impossible for an attacker to create a valid ciphertext without knowing the corresponding plaintext, thereby rendering the decryption oracle available to an IND-CCA2 adversary useless. The most natural idea is to add a Schnorr signature [37] to the ciphertext, as was proposed in [24,42]: to encrypt a message $M \in \mathbb{G}$, in addition to the usual ElGamal ciphertext $R = G^r$, $Y = MX^r$, a Schnorr signature (with secret key r) is computed using a random oracle $\boldsymbol{H_c}$ by drawing a random integer $a \in \mathbb{Z}_p^*$, and setting $A = G^a$, $c = \boldsymbol{H_c}(Y, R, A)$, and $s = a + cr \mod p$. The ciphertext is then (Y, R, s, c). To decrypt with the secret key x, first check whether $c = \boldsymbol{H_c}(Y, R, G^s R^{-c})$, then return $Y/R^x = M$ (or $\perp$ if the check failed). We call this scheme Schnorr-Signed ElGamal (SS-EG for short) encryption.

Intuitively, since the Schnorr proof of knowledge is extractable in the ROM [33,34], the security reduction should be able to extract the integer r used to form a ciphertext (Y, R, s, c) and answer decryption queries issued by an IND-CCA2 adversary without knowing the secret key x as Y/X^r. However, no one knows how to turn this intuition into a formal security proof. As clearly explained by Shoup and Gennaro [41, Sec. 7.2], the problem is that Schnorr signatures are not known to be *online* (*a.k.a. straight-line*) extractable [32,20]: the extractor

[1] Regarding non-adaptive chosen-ciphertext attacks, *i.e.* IND-CCA1 security of ElGamal encryption, see [29].

[2] In fact, the weaker notion of indistinguishability under one-time chosen-ciphertext attacks (IND-OTCCA) is sufficient [15,22].

[3] The SDH assumption is slightly weaker than the Gap Diffie-Hellman (GDH) assumption [31]: the first element of the triplets submitted to the DDH oracle is fixed for the SDH problem, whereas it can be freely chosen for the GDH problem.

needs to rewind the prover (here, the adversary) in order to extract r. This causes problems if the adversary orders its random oracle and decryption queries in a certain way (*e.g.*, if it asks n random oracle queries $c_i = \boldsymbol{H_c}(Y_i, R_i, A_i)$ corresponding to the Schnorr signature for the encryption of n messages M_i, and then asks the decryption queries for the messages in reverse order, that is for $M_n, \ldots, M_1$). Tsiounis and Yung [42] gave a proof that SS-EG encryption is IND-CCA2-secure using a strong assumption on Schnorr signatures (which basically amounts to saying that they are online extractable). Schnorr and Jakobsson [38] showed that SS-EG encryption is IND-CCA2-secure by combining the ROM and the Generic Group Model (GGM). However, as noted by Fischlin [19], combining these two idealized models may be even more problematic than considering each one independently.

Shoup and Gennaro [41] proposed two variants of SS-EG encryption named TDH1 and TDH2, whose hybrid variants are IND-CCA2-secure in the ROM under respectively the CDH and the DDH assumption (regarding non-hybrid variants, see the full version of this paper [39]). They were primarily concerned with the context of threshold encryption, where it is necessary that decryption servers be able to publicly verify (*i.e.* without knowing the secret key) the validity of a ciphertext before starting the decryption process. This constraint leads to schemes which, though efficient, are more complex than SS-EG. It is therefore a natural question whether removing the necessity to be able to publicly verify the validity of a ciphertext could yield more efficient provably IND-CCA2-secure variants of SS-EG encryption.

Robustness. Robustness for an encryption scheme informally means that it is hard to create a ciphertext that decrypts to a valid plaintext under two different secret keys.[4] This notion was only recently formalized by Abdalla *et al.* [3], and further studied by Mohassel [30]. Robustness can always be achieved by appending the public key of the intended receiver to the ciphertext (and checking it on decryption). However, the notion becomes really interesting when coupled with anonymity (*a.k.a.* key privacy) [6], which requires that a ciphertext does not reveal the public key under which it was created—and then the previous solution for robustness becomes inadequate. Anonymity and robustness not only help make encryption more resistant to misuse, but also are inherently important properties for some applications of public key encryption such as encryption with keyword search [1] or auction protocols [36].

Basic ElGamal encryption can be shown anonymous under CPA-attacks assuming the DDH problem is hard [6], however it is clearly not robust since any ciphertext is valid. This is also true for Schnorr-Signed ElGamal[5] since the validity check does not depend on the secret nor the public key (anonymity and robustness

[4] This is strong robustness. Weak robustness means that it is impossible to produce a plaintext that, once encrypted under some public key, decrypts to some valid plaintext under the secret key corresponding to a second, different public key.

[5] Schnorr-Signed ElGamal can easily be seen to achieve anonymity under CPA-attacks. We expect however that proving anonymity under CCA2-attacks will run into the same problems as a proof of IND-CCA2-security.

for TDH1/2 are more subtle, but neither of them achieves both anonymity and robustness, see the full version of the paper [39]). It is claimed in [3] that a very minimal modification to DHIES [4] (which is anonymous under CCA2-attacks) renders the scheme robust. Though no formal proof is available, such a proof seems to require non-standard assumptions on the MAC scheme used.[6] We also note that all variants of ElGamal encryption with compact ciphertexts where the decryption algorithm never rejects [28,11,5] cannot be robust (independently of whether they are anonymous or not). Hence, it seems to be an open problem to propose a simple variant of ElGamal encryption achieving both anonymity under CCA2-attacks and robustness without additional non-standard assumption.

Contributions of This Work. We propose a very simple variant of Schnorr-Signed ElGamal encryption, and prove that it is IND-CCA2-secure in the ROM under the DDH assumption by showing that it is plaintext-aware. Additionally, we show that the resulting encryption scheme is not only anonymous under CCA2 attacks assuming the DDH problem is hard, but also strongly robust (assuming the additional check at decryption that the randomness used to encrypt was not 0). The proof of robustness only requires that the hash function used to instantiate the random oracle be collision-resistant.

The modification only affects the way the random challenge c for the signature is generated; in particular the key size and the ciphertexts size remain unchanged compared with Schnorr-Signed ElGamal. We name our new scheme Chaum-Pedersen-Signed ElGamal (CPS-EG for short) encryption since it uses a proof of *equality* of discrete logarithms (introduced by Chaum and Pedersen [13]) rather than a proof of knowledge of discrete logarithm. This idea already proved fruitful for signature schemes [25,21]. In fact, our scheme can be seen as an optimized variant of TDH2 [41]. A comparison of CPS-EG with existing related schemes can be found in Table 1.

We prove our scheme to be in a sense *minimal* relatively to plaintext awareness: removing any input to the random oracle when computing the challenge for the signature makes the scheme provably (under some computational assumption) *not* plaintext-aware.[7] In particular, we show that Schnorr-Signed ElGamal encryption is *not* plaintext-aware as defined in [7] under the CDH assumption. We think that this observation might help explain why progress has remained elusive regarding IND-CCA2-security of this scheme.

We also analyze the hybrid version of our scheme, named HCPS-EG, and show that it is IND-CCA2-secure in the ROM under the CDH assumption. As the non-hybrid variant, the scheme is anonymous under CCA2-attacks (assuming the CDH problem is hard) and strongly robust (assuming a collision-resistant hash function). An interesting feature of this scheme is that it achieves IND-CCA2-security using the weakest form of security for the data encapsulation

[6] Personal communication with the authors of [3].

[7] We stress that we refer here to the strong notion of plaintext awareness in the ROM as defined in [7]. Our results seem unlikely to be extensible to weaker notions such as the PA2-RO notion suggested in [8].

mechanism (DEM), namely ciphertext indistinguishability under one-time attacks (IND-OT-security) [15,22]. In more concrete terms, the HCPS-EG scheme can be safely used with AES in counter mode. This is a property shared with other ElGamal variants, notably [11,5]. However for these two schemes, the decryption algorithm never rejects (they aim at compact ciphertexts), and hence they cannot be robust. A summary of results regarding HCPS-EG as well as a comparison with related schemes can be found in Table 1. Fully detailed analysis is deferred to the full version of the paper [39].

Related Work. Abe [2] studied generic ways to combine a hybrid encryption scheme and a signature scheme (which in particular applies to Hashed ElGamal and Schnorr signatures) to obtain an IND-CCA2 encryption scheme in the ROM where validity of a ciphertext can be publicly checked. Besides, a lot of efforts were devoted to variants of ElGamal encryption provably IND-CCA2-secure in the standard model. This started with the "double-base" variant by Damgård [16], which is IND-CCA1-secure under the non-standard Diffie Hellman Knowledge (DHK) assumption [8]. Later, Cramer and Shoup [14] proposed their famous cryptosystem provably IND-CCA2-secure under the DDH assumption, and formally introduced the notion of hybrid encryption [15]. Notable subsequent work includes [27,26].

Organization. In Section 2, we recall the necessary background on ElGamal encryption and plaintext awareness. We describe our new scheme and prove that it is plaintext-aware, hence IND-CCA2-secure in Section 3. Then, we show in Section 4 that our scheme is in a sense minimal if one wants to obtain plaintext awareness. In Section 5, we study anonymity and robustness of our new scheme. Some proofs are omitted for reasons of space and can be found (except for Theorem 2, a known result) in the full version of this paper [39].

2 Preliminaries

2.1 Basic Definitions

The security parameter will be denoted k. When S is a non-empty finite set, we write $s \leftarrow_{\$} S$ to mean that a value is sampled uniformly at random from S and assigned to s. By $z \leftarrow \mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\cdots}(x, y, \ldots)$ we denote the operation of running the (possibly probabilistic) algorithm $\mathcal{A}$ on inputs $x, y, \ldots$ with access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$ (possibly none), and letting z be the output. PPT will stand for probabilistic polynomial-time.

A (prime-order) group generator `GpGen` is a PPT algorithm that takes a security parameter 1^k and outputs a triplet $(\mathbb{G}, p, G)$ where $\mathbb{G}$ is a group[8] of prime order $p \in [2^{k-1}, 2^k[$ and G is a generator of $\mathbb{G}$. In all the following, we will assume that all algorithms are given $(\mathbb{G}, p, G)$ (*e.g.*, as part of a public key) and will not denote it explicitly. We denote $1_{\mathbb{G}}$ the identity element of $\mathbb{G}$.

[8] More precisely it outputs the description of the group, but we confound it here for simplicity.

Table 1. Comparison of variants of basic (non-hybrid) ElGamal encryption. The secret, resp. public key size is in number of mod p integers, resp. group elements. The next two columns give the number of exponentiations per encryption and decryption. For encryption, we separate off-line/online exponentiations. We count $G^s R^{-c}$ as one single exponentiation even though it is slightly more expensive. The ciphertext size is given in number of group elements plus mod p integers. The IND column gives the assumption needed to prove IND-CCA2-security (all schemes except ElGamal use the ROM). The ANON+SROB column indicates whether the scheme achieves ANON-CCA2-security and strong robustness, as well as the assumptions needed. DDH stands for Decisional Diffie-Hellman, GGM for Generic Group Model, and CRHF for Collision-Resistant Hash Function. Regarding the security properties of TDH1/2, see the full version of the paper [39].

scheme	`sk`/`pk` size	exp./enc.	exp./dec.	cip. size	IND	ANON +SROB	Refs
ElGamal	1	2/0	1	$2G$	DDH (CPA only)	No	[18,42]
SS-EG	1	3/0	2	$2G+2p$	GGM	No	[38]
TDH1	1	3/2	3	$3G+2p$	DDH	No	[41]
TDH2	1/2	5/0	3	$3G+2p$	DDH	No	[41]
CPS-EG	1	4/0	3	$2G+2p$	DDH	Yes (DDH+ CRHF)	Sec. 3 & 5

Table 2. Comparison of variants of hashed ElGamal encryption. The first three columns are as for Table 1. The ciphertext expansion is given in number of group elements, plus the size $|\tau|$ of a MAC in case an authenticated DEM is needed. The DEM column indicates with which type of Data Encapsulation Mechanism the scheme is assumed to be used (see the full version [39] for definitions of AE-OT and IND-OT). SPRP means that the DEM must be a (variable-input length) strong pseudorandom permutation. The IND and ANON+SROB columns are as for Table 1. For (Twin-)DHIES, the star indicates that a non-standard assumption is needed for the MAC to achieve strong robustness. CDH stands for Computational Diffie-Hellman, SDH for Strong DH, and GDH for Gap DH.

scheme	`sk`/`pk` size	exp./enc.	exp./dec.	cip. exp.	DEM	IND	ANON +SROB	Refs
DHIES	1	2	1	$G+\|\tau\|$	AE-OT	SDH	Yes*	[4]
Twin-DHIES	2	3	2	$G+\|\tau\|$	AE-OT	CDH	Yes*	[12]
KM	1	2	1	G	SPRP	SDH	No	[28]
Twin-KM	2	3	2	G	SPRP	CDH	No	[12]
AKO	1	2	1	G	IND-OT	SDH	No	[5]
Twin AKO	2	3	2	G	IND-OT	CDH	No	[5]
Boyen	1	3	2	G	IND-OT	GDH	No	[11]
HCPS-EG	1	4	3	$G+2p$	IND-OT	CDH	Yes (CDH+ CRHF)	full version [39]

We say that the Computational Diffie-Hellman (CDH) problem is hard relatively to GpGen if the following advantage:

$$\mathbf{Adv}^{\mathrm{cdh}}_{\mathtt{GpGen},\mathcal{A}}(k) = \Pr\left[(\mathbb{G}, p, G) \leftarrow \mathtt{GpGen}(1^k), x, y \leftarrow_\$ \mathbb{Z}_p, Z \leftarrow \mathcal{A}(G^x, G^y) : Z = G^{xy}\right]$$

is negligible for any PPT adversary $\mathcal{A}$.

We say that the Decisional Diffie-Hellman (DDH) problem is hard relatively to GpGen if the following advantage:

$$\mathbf{Adv}^{\mathrm{ddh}}_{\mathtt{GpGen},\mathcal{A}}(k) = \Big| \Pr\left[(\mathbb{G}, p, G) \leftarrow \mathtt{GpGen}(1^k), x, y \leftarrow_\$ \mathbb{Z}_p : 1 \leftarrow \mathcal{A}(G^x, G^y, G^{xy})\right] - \Pr\left[(\mathbb{G}, p, G) \leftarrow \mathtt{GpGen}(1^k), x, y, z \leftarrow_\$ \mathbb{Z}_p : 1 \leftarrow \mathcal{A}(G^x, G^y, G^z)\right] \Big|$$

is negligible for any PPT adversary $\mathcal{A}$.

Random oracles are used for distinct goals in the various schemes considered in this paper, and will be denoted as follows: $\boldsymbol{H_c}$ will denote the random oracle used to generate the challenge for the signature, $\boldsymbol{H_K}$ will denote the random oracle used to derive the key for the DEM in hybrid schemes, and $\boldsymbol{H_G}$ will denote a random oracle mapping to $\mathbb{G}$ (only used in TDH1). We will use $\boldsymbol{H}$ as a shortcut for the set of random oracles accessed by a scheme.

Definition 1 (Encryption scheme). *A public key encryption scheme* PKE *is a triplet of polynomial-time algorithms* (PKE.Kg, PKE.Enc, PKE.Dec) *where:*

- PKE.Kg*, the (probabilistic)* key generation algorithm*, takes a security parameter* 1^k *and returns a secret/public key pair* (sk, pk)*.*
- PKE.Enc*, the (probabilistic)* encryption algorithm*, takes a public key* pk *and a message* M *and returns a ciphertext* ψ*.*
- PKE.Dec*, the (deterministic)* decryption algorithm*, takes a secret key* sk *and a ciphertext* ψ *and returns either a message* M *or the special symbol* $\perp$ *that indicates that the ciphertext is invalid.*

We assume that a public key pk defines a message space MsgSp(pk) and for consistency we impose that for any k:

$$\Pr\left[(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathtt{PKE.Kg}(1^k), M \leftarrow_\$ \mathtt{MsgSp}(\mathtt{pk}), \psi \leftarrow \mathtt{PKE.Enc}(\mathtt{pk}, M) : \mathtt{PKE.Dec}(\mathtt{sk}, \psi) = M\right] = 1 \ .$$

We recall the usual security definitions for PKE schemes in the full version of the paper [39].

The ElGamal PKE encryption scheme is formally defined in Figure 1 (recall that the group parameters $(\mathbb{G}, p, G)$ are implicitly included in pk, and here $\mathtt{MsgSp}(\mathtt{pk}) = \mathbb{G}$).

The following classical security result regarding ElGamal encryption is due to Tsiounis and Yung.

Theorem 1 ([42]). *The ElGamal encryption scheme is IND-CPA-secure if and only if the DDH problem is hard relatively to* GpGen.

ElGamal PKE scheme

$\texttt{PKE.Kg}(1^k)$	$\texttt{PKE.Enc}(\texttt{pk} = X, M)$	$\texttt{PKE.Dec}(\texttt{sk} = x, \psi)$
$(\mathbb{G}, p, G) \leftarrow \texttt{GpGen}(1^k)$ $x \leftarrow_\$ \mathbb{Z}_p^*$; $X \leftarrow G^x$ $\texttt{sk} \leftarrow x$; $\texttt{pk} \leftarrow X$ Return $(\texttt{sk}, \texttt{pk})$	$r \leftarrow_\$ \mathbb{Z}_p^*$ $R \leftarrow G^r$; $R' \leftarrow X^r$ $Y \leftarrow MR'$ Return $\psi \leftarrow (Y, R)$	Parse ψ as (Y, R) $R' \leftarrow R^x$ Return $M \leftarrow Y/R'$

Fig. 1. The ElGamal encryption scheme

2.2 Plaintext Awareness in the ROM

The notion of plaintext awareness was first suggested in [9] to capture the idea that the only way that an adversary can produce a valid ciphertext is to apply the encryption algorithm to the public key and a message. The motivation was that IND-CPA-security coupled with plaintext awareness should yield IND-CCA2-security, since this property would make the decryption oracle available to an IND-CCA2 adversary useless. The original definition in [9], which was formalized in the ROM, was found too weak to imply IND-CCA2-security, and was later adequately refined in [7]. Providing a satisfactory definition for the standard model turned out to be more subtle and was achieved in [8]. Though it was initially thought as a simple tool geared towards proofs of IND-CCA2-security, intrinsic motivations for studying plaintext awareness were later proposed, in particular in order to securely instantiate the ideal encryption functions of the Dolev-Yao model [17,23], or to provide deniability to key exchange protocols [35].

In this work, we use the definition of plaintext awareness in the ROM introduced in [7] and we refer to this definition as ROM-PA-security. This definition involves two types of algorithms: a ciphertext creator $\mathcal{C}$ and a plaintext extractor $\mathcal{P}$. The ciphertext creator is given a public key pk and has access to the random oracle $\boldsymbol{H}$ and to the encryption algorithm $\texttt{PKE.Enc}_{\text{pk}}^{\boldsymbol{H}}$. All queries of $\mathcal{C}$ to the random oracle and corresponding answers are recorded in a list $L_{\boldsymbol{H}}$. All answers (ciphertexts) received from the encryption oracle are recorded in a list L_ψ (the corresponding plaintexts are *not* recorded). $\mathcal{C}$ outputs a ciphertext $\psi \notin L_\psi$. We write $(L_{\boldsymbol{H}}, L_\psi, \psi) \leftarrow \texttt{run}\ \mathcal{C}^{\boldsymbol{H}, \texttt{PKE.Enc}_{\text{pk}}^{\boldsymbol{H}}}(\texttt{pk})$. The plaintext extractor $\mathcal{P}$ takes as input $(L_{\boldsymbol{H}}, L_\psi, \psi, \texttt{pk})$ and aims at returning the plaintext corresponding to ψ.

Definition 2 (ROM-PA). *Let* $\texttt{PKE} = (\texttt{PKE.Kg}, \texttt{PKE.Enc}, \texttt{PKE.Dec})$ *be an encryption scheme.* PKE *is said to be* secure in the sense of plaintext awareness in the ROM *(ROM-PA-secure) if there is a PPT algorithm* $\mathcal{P}$ *(the* plaintext extractor*) such that for any PPT ciphertext creator* $\mathcal{C}$*, the failure probability of* $\mathcal{P}$ *relatively to* $\mathcal{C}$*, defined as:*

$$\begin{aligned}&\texttt{Fail}^{\text{pa}}_{\texttt{PKE},\mathcal{P},\mathcal{C}}(k) = \Pr\big[(\texttt{pk}, \texttt{sk}) \leftarrow \texttt{PKE.Kg}(1^k);\\ &\quad (L_{\boldsymbol{H}}, L_\psi, \psi) \leftarrow \texttt{run}\ \mathcal{C}^{\boldsymbol{H}, \texttt{PKE.Enc}_{\text{pk}}^{\boldsymbol{H}}}(\texttt{pk})\ : \mathcal{P}(L_{\boldsymbol{H}}, L_\psi, \psi, \texttt{pk}) \neq \texttt{PKE.Dec}_{\texttt{sk}}^{\boldsymbol{H}}(\psi)\big]\ ,\end{aligned}$$

is a negligible function (the probability is taken over the random tape of all algorithms and the answers of the random oracle).

One may wonder why the ciphertext creator is given access to an encryption oracle since it can encrypt plaintexts by itself using the public key. However, this reflects the fact that an adversary may obtain ciphertexts it has not encrypted by itself (in particular, this is the case of the challenge ciphertext in an IND-CCA2 security experiment), in which case it may not necessarily know the corresponding random oracle queries (which are consequently not listed in $L_{\boldsymbol{H}}$). See [40,7] for a detailed discussion. Bellare *et al.* [7] showed the following theorem.

Theorem 2 ([7]). *If a PKE scheme is both IND-CPA-secure and ROM-PA-secure, then it is IND-CCA2-secure.*

They also showed that ROM-PA-security is strictly stronger than IND-CCA2-security: there exist IND-CCA2-secure PKE schemes that are not ROM-PA-secure (provided IND-CCA2-secure PKE schemes exist at all).

Note that in the above definition of ROM-PA-security, the plaintext extractor is not given access to the random oracle: it must work with the list of random oracle queries of the ciphertext creator. If one allows the plaintext creator to freely access the random oracle, one loses the intuitively appealing constraint for the plaintext extractor to work given only the view of the ciphertext creator. However, for the hybrid version of the scheme (see the full version [39]), we will need this relaxation of the definition, and will call ROM-PA'-security the resulting property. Fortunately, it can be checked that the proof of Theorem 2 can be straightforwardly transposed to ROM-PA'-security, namely IND-CPA-security plus ROM-PA'-security implies IND-CCA2-security.

Though the ElGamal encryption scheme does not use the ROM, it is clear that it cannot satisfy any notion of plaintext awareness since an adversary can simply output a random pair $(Y, R) \leftarrow_{\$} \mathbb{G}^2$, which implies that a plaintext extractor should be able to break the one-wayness of the scheme, which holds under the CDH assumption. An attempt to make ElGamal encryption plaintext-aware is to add a Schnorr signature, resulting in what we call the Schnorr-Signed ElGamal (SS-EG) encryption scheme [42,24]. Let $\boldsymbol{H_c} : \mathbb{G}^3 \to \mathbb{Z}_p$ be a random oracle. It is defined in Figure 2.

Clearly, this scheme inherits IND-CPA-security in the ROM under the DDH assumption from basic ElGamal encryption (the reduction only needs to simulate the signature when preparing the challenge ciphertext by programming the random oracle adequately).[9] However, as discussed in the introduction, there is no known proof that this scheme is IND-CCA2-secure in a weaker model than the combination of the ROM and the GGM. In Section 4, we prove that this scheme is in fact provably *not* ROM-PA-secure under the CDH assumption.

[9] Note that in the standard model, one can easily come with instantiations of $\boldsymbol{H_c}$ that ruin even IND-CPA-security of the scheme.

SS-EG PKE scheme

```
$\mathtt{PKE.Kg}(1^k)$
$(\mathbb{G}, p, G) \leftarrow \mathtt{GpGen}(1^k)$
$x \leftarrow_\$ \mathbb{Z}_p^*$; $X \leftarrow G^x$
$\mathtt{sk} \leftarrow x$; $\mathtt{pk} \leftarrow X$
Return $(\mathtt{sk}, \mathtt{pk})$
```

```
$\mathtt{PKE.Enc}(\mathtt{pk} = X, M)$
$r, a \leftarrow_\$ \mathbb{Z}_p^*$
$R \leftarrow G^r$; $R' \leftarrow X^r$
$Y \leftarrow MR'$
$A \leftarrow G^a$
$c \leftarrow \boldsymbol{H_c}(Y, R, A)$
$s = a + cr \mod p$
Return $\psi \leftarrow (Y, R, s, c)$
```

```
$\mathtt{PKE.Dec}(\mathtt{sk} = x, \psi)$
Parse $\psi$ as $(Y, R, s, c)$
$R' \leftarrow R^x$
$A \leftarrow G^s R^{-c}$
if $\boldsymbol{H_c}(Y, R, A) \neq c$
    Return $\perp$
Return $M \leftarrow Y/R'$
```

Fig. 2. The SS-EG encryption scheme

3 Chaum-Pedersen-Signed ElGamal Encryption

In this section, we describe our modification of the SS-EG encryption scheme and analyze its security. We name the new scheme Chaum-Pedersen-Signed ElGamal (CPS-EG for short) encryption. The change is quite small: we simply add two elements, $R' = R^x$ and $A' = A^x$, in the call to the random oracle $\boldsymbol{H_c}$ when computing the challenge c for the signature. This corresponds to moving from a proof of knowledge of the discrete logarithm $r = \mathtt{DLog}_G(R)$ to a Chaum-Pedersen [13] proof of *equality* of discrete logarithms $\mathtt{DLog}_G(R) = \mathtt{DLog}_X(R')$. The scheme uses a random oracle $\boldsymbol{H_c} : \mathbb{G}^5 \rightarrow \mathbb{Z}_p$. It is defined in Figure 3.

CPS-EG PKE scheme

```
$\mathtt{PKE.Kg}(1^k)$
$(\mathbb{G}, p, G) \leftarrow \mathtt{GpGen}(1^k)$
$x \leftarrow_\$ \mathbb{Z}_p^*$; $X \leftarrow G^x$
$\mathtt{sk} \leftarrow x$; $\mathtt{pk} \leftarrow X$
Return $(\mathtt{sk}, \mathtt{pk})$
```

```
$\mathtt{PKE.Enc}(\mathtt{pk} = X, M)$
$r, a \leftarrow_\$ \mathbb{Z}_p^*$
$R \leftarrow G^r$; $R' \leftarrow X^r$
$Y \leftarrow MR'$
$A \leftarrow G^a$; $A' \leftarrow X^a$
$c \leftarrow \boldsymbol{H_c}(Y, R, R', A, A')$
$s = a + cr \mod p$
Return $\psi \leftarrow (Y, R, s, c)$
```

```
$\mathtt{PKE.Dec}(\mathtt{sk} = x, \psi)$
Parse $\psi$ as $(Y, R, s, c)$
$R' \leftarrow R^x$
$A \leftarrow G^s R^{-c}$; $A' \leftarrow A^x$
if $\boldsymbol{H_c}(Y, R, R', A, A')$
                          $\neq c$
    Return $\perp$
Return $M \leftarrow Y/R'$
```

Fig. 3. The CPS-EG encryption scheme

Note that the correctness of a ciphertext cannot be checked publicly (*i.e.* without knowledge of the secret key x). As for the Schnorr-Signed variant, it is straightforward to prove that the scheme remains IND-CPA-secure under the DDH assumption for $\mathtt{GpGen}$.

Theorem 3. *Assume that the DDH problem is hard for* $\mathtt{GpGen}$. *Then the CPS-EG encryption scheme is IND-CPA-secure.*

The intuition regarding why the CPS-EG scheme is ROM-PA-secure is simple: since R' is now included in the random oracle queries, the plaintext extractor will directly be able to decrypt Y once it has located the corresponding query. Namely, upon reception of a ciphertext (Y, R, s, c), it can compute $A = G^s R^{-c}$, and look throughout the list of random oracle queries to find the (unique with high probability) query (Y, R, R', A, A') such that the answer was c. There is however a small caveat: the plaintext extractor must check that this query was correctly formed, *i.e.* $R' = R^x$ and $A' = A^x$. This is ensured by the following lemma (which is the core of the proof of soundness of the Chaum-Pedersen protocol [13], and also the base of the "twinning" technique of Cash *et al.* [12]).

Lemma 1. *Fix $X = G^x$, $x \in \mathbb{Z}_p^*$. Let $R = G^r, R', A = G^a, A' \in \mathbb{G}$ be four group elements such that $r, a \neq 0 \mod p$, and $R' \neq R^x$ or $A' \neq A^x$. Then there is at most one integer $c \in \mathbb{Z}_p$ such that there exists $s \in \mathbb{Z}_p$ satisfying both $G^s = AR^c$ and $X^s = A'R'^c$.*

Proof. Denote $R' = R^y$ and $A' = A^z$, and assume that there exists (s_1, c_1) and (s_2, c_2) with $c_1 \neq c_2$ satisfying $G^{s_i} = AR^{c_i}$ and $X^{s_i} = A'R'^{c_i}$ for $i = 1, 2$. Then $G^{xs_i} = A^x R^{xc_i} = A'R'^{c_i} = A^z R^{yc_i}$, which implies $a(x - z) = rc_i(y - x) \mod p$. Hence if $y \neq x$, then $c_1 = c_2 = a(x - z)/r(y - x) \mod p$, contradicting the assumption that $c_1 \neq c_2$. If $y = x$ then $z = x$ (since $a \neq 0 \mod p$) which contradicts the assumption that $R' \neq R^x$ or $A' \neq A^x$. □

We now give the formal proof that the scheme is ROM-PA-secure.

Theorem 4. *The CPS-EG encryption scheme is ROM-PA-secure. Hence, it is IND-CCA2-secure in the ROM under the assumption that DDH is hard for* `GpGen`.

Proof. Consider a ciphertext creator $\mathcal{C}$ making at most q_h random oracle queries and q_e queries to the encryption oracle (these queries are irrelevant to the analysis). Let $(L_{\boldsymbol{H}}, L_\psi, \psi)$ be the output of a run of $\mathcal{C}$ on public key $\mathtt{pk} = X = G^x$, where $\psi = (Y, R, s, c)$. The plaintext extractor $\mathcal{P}$ proceeds as follows: it computes $A = G^s R^{-c}$ and looks throughout $L_{\boldsymbol{H}}$ for all queries of the form $(Y, R, *, A, *)$, where $*$ denotes any group element, such that the answer was c. If there is no such query, it returns $\perp$ (meaning that the ciphertext is invalid). If there is more than one such query, the plaintext extractor aborts. Otherwise, denote (Y, R, R', A, A') the unique query such that the answer was c. $\mathcal{P}$ checks whether $X^s = A'R'^c$. If this does not hold, it returns $\perp$ (meaning that the ciphertext is invalid). Otherwise it returns $M = Y/R'$ as the plaintext.

We now analyze the failure probability of the plaintext extractor. First, note that since $\psi \notin L_\psi$, the query (Y, R, R^x, A, A^x) cannot have been issued to the random oracle during any of the q_e calls of the ciphertext creator to $\mathtt{PKE.Enc}_{\mathtt{pk}}^{\boldsymbol{H}}$. Hence, if this query is not in $L_{\boldsymbol{H}}$, then the view of $\mathcal{C}$ and $\mathcal{P}$ is independent of the value of $\boldsymbol{H_c}$ at this point. We consider four distinct cases.

1. If there is no query of the form $(Y, R, *, A, *)$ such that the answer was c, then the ciphertext is valid with probability at most $1/p$. Hence, by returning $\perp$, the plaintext extractor errs with probability at most $1/p$.

2. The probability that there are two queries to the random oracle such that the answers were the same is upper bounded by $q_h^2/2p$; this is also an upper bound on the probability that the plaintext extractor aborts.
3. If there is a unique query (Y, R, R', A, A') such that the answer was c, but $A' \neq X^s R'^{-c}$, then either $R' \neq R^x$ or $A' \neq A^x$. Consequently, the ciphertext is valid with probability at most $1/p$ and again, by returning $\perp$, the plaintext extractor errs with probability at most $1/p$.
4. If there is a unique query (Y, R, R', A, A') such that the answer was c, and $X^s = A'R'^c$, then two situations can occur. If $R' = R^x$ and $A' = A^x$, then it is clear that the plaintext extractor returns the plaintext that would be returned by $\texttt{PKE.Dec}_{\texttt{sk}}^{\boldsymbol{H}}$. Else, $R' \neq R^x$ or $A' \neq A^x$, and according to Lemma 1, there is at most one integer c_0 such that there exists s satisfying both $G^s = AR^{c_0}$ and $X^s = A'R'^{c_0}$, and for each query the random oracle returned $c = c_0$ with probability at most $1/p$. Hence the plaintext extractor is fooled into returning an incorrect plaintext with probability at most q_h/p.

Collecting all cases shows that the failure probability of the plaintext extractor is at most $2q_h^2/p$, which is negligible in the security parameter. It follows from Theorems 2 and 3 that the scheme is IND-CCA2-secure in the ROM under the assumption that DDH is hard for `GpGen`. □

We note that CPS-EG seems to be the simplest IND-CCA2-secure scheme in the ROM that retains some kind of homomorphic property. Namely, given two ciphertexts (Y_1, R_1, s_1, c_1) and (Y_2, R_2, s_2, c_2) corresponding respectively to plaintexts M_1 and M_2, one can compute the first two elements of the ciphertext for $M_1 M_2$ as for basic ElGamal encryption. It is however impossible to "sign" the new ciphertext without knowledge of the random values used to encrypt M_1 and M_2. This property appears to be useful in e-voting applications [43,10].

4 Minimality of CPS-EG Regarding Plaintext Awareness

In this section, we consider whether the CPS-EG encryption scheme remains ROM-PA-secure when some inputs are removed from the call to the random oracle $\boldsymbol{H_c}$. Surprisingly, we are able to show that under an adequate assumption, the resulting scheme *cannot* be ROM-PA-secure. Namely:

– If we remove R' and A', we exactly recover the SS-EG scheme, and we can show that the scheme is not ROM-PA-secure under the CDH assumption.
– If we remove one single element among R, R', A, or A', then the scheme is not ROM-PA-secure under the DDH assumption (removing Y trivially makes the scheme malleable).

This is captured by the following two theorems.

Theorem 5. *Assume that the CDH problem is hard for* `GpGen`*. Then the SS-EG encryption scheme is not ROM-PA-secure.*

Proof. Assume for contradiction that SS-EG encryption is ROM-PA-secure, and let $\mathcal{P}$ be a plaintext extractor. We build a reduction $\mathcal{R}$ that solves the CDH problem. Let $(\mathbb{G}, p, G) \leftarrow \texttt{GpGen}(1^k)$ and $(X = G^x, R = G^r)$ denote the input to $\mathcal{R}$. $\mathcal{R}$ simulates the run of a ciphertext creator as follows. It draws two integers $s, c \leftarrow_\$ \mathbb{Z}_p$, a random group element $Y \leftarrow_\$ \mathbb{G}$, computes $A = G^s R^{-c}$, and programs the random oracle with $\boldsymbol{H_c}(Y, R, A) = c$. Then it runs the plaintext extractor with input $L_{\boldsymbol{H_c}} = ((Y, R, A), c)$, $L_\psi = \emptyset$, $\psi = (Y, R, s, c)$, and $\texttt{pk} = X$. It is clear that the ciphertext is valid, and that the simulation of the random oracle is perfect. Hence, with overwhelming probability, $\mathcal{P}$ returns the plaintext M corresponding to ψ, from which $\mathcal{R}$ can compute $G^{rx} = Y/M$. $\square$

Since ROM-PA-security is strictly stronger than IND-CCA2-security, the result above does not imply that SS-EG is not IND-CCA2-secure in the ROM. Would SS-EG be proved IND-CCA2-secure this would yield a natural separation between this notion and ROM-PA-security (the separation provided in [7] used a rather contrived counter-example, but without any computational assumption). Also, there are many possible ways to weaken the definition of plaintext awareness in the ROM. The theorem above seems to crucially rely on the impossibility for the plaintext extractor to rewind the ciphertext creator or to program the random oracle: it must work online, and can only observe the ciphertext creator queries. In particular, [8] proposed weaker notions of PA in the ROM where the plaintext extractor is not black-box (it can depend on the code of the ciphertext creator), and is given the random coins of the ciphertext creator, which enables to rewind it. Exploring whether SS-EG may fulfill such weaker definitions is an interesting open question.

Regarding the minimality of CPS-EG, we have the following result.

Theorem 6. *Assume that the DDH problem is hard for* `GpGen`. *Then the CPS-EG encryption scheme does not remain ROM-PA-secure when any of the four elements R, R', A or A' is removed from the inputs to $\boldsymbol{H_c}$ when generating c. When Y is removed the scheme becomes (unconditionally) malleable and hence is not IND-CCA2-secure nor ROM-PA-secure.*

5 Anonymity and Robustness of CPS-EG

The formal definition of anonymity for a PKE scheme is recalled in the full version [39]. Informally, this requires that an adversary cannot distinguish under which public key an (adversarially chosen) message was encrypted. We start with showing that the CPS-EG scheme provides anonymity under CCA2 attacks.

Theorem 7. *Assume that the DDH problem is hard for* `GpGen`. *Then the CPS-EG encryption scheme is ANON-CCA2-secure.*

The definition of strong robustness is recalled in the full version [39]. We will now see that the CPS-EG scheme achieves strong robustness, assuming the following very simple tweak to the scheme (in fact, the same was proposed for Cramer-Shoup encryption in [3]). We define the CPS-EG* scheme exactly as the CPS-EG scheme, with the additional check on decryption that $R \neq 1_{\mathbb{G}}$ (if the check fails, then the decryption algorithm returns $\perp$).

Theorem 8. *Assume $\boldsymbol{H_c}$ is instantiated with a collision-resistant hash function family. Then the CPS-EG* encryption scheme is SROB-CCA-secure.*

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. Journal of Cryptology 21(3), 350–391 (2008)
2. Abe, M.: Securing Encryption + Proof of Knowledge in the Random Oracle Model. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 277–289. Springer, Heidelberg (2002)
3. Abdalla, M., Bellare, M., Neven, G.: Robust Encryption. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 480–497. Springer, Heidelberg (2010)
4. Abdalla, M., Bellare, M., Rogaway, P.: The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)
5. Abe, M., Kiltz, E., Okamoto, T.: Compact CCA-Secure Encryption for Messages of Arbitrary Length. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 377–392. Springer, Heidelberg (2009)
6. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-Privacy in Public-Key Encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001)
7. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among Notions of Security for Public-Key Encryption Schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998)
8. Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption Without Random Oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)
9. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
10. Bernhard, D., Cortier, V., Pereira, O., Smyth, B., Warinschi, B.: Adapting Helios for Provable Ballot Privacy. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 335–354. Springer, Heidelberg (2011)
11. Boyen, X.: Miniature CCA2 PK Encryption: Tight Security Without Redundancy. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 485–501. Springer, Heidelberg (2007)
12. Cash, D.M., Kiltz, E., Shoup, V.: The Twin Diffie-Hellman Problem and Applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008)
13. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
14. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)

15. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM Journal on Computing 33(1), 167–226 (2003)
16. Damgård, I.B.: Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)
17. Dolev, D., Yao, A.C.-C.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–207 (1983)
18. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
19. Fischlin, M.: A Note on Security Proofs in the Generic Model. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 458–469. Springer, Heidelberg (2000)
20. Fischlin, M.: Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005)
21. Goh, E.-J., Jarecki, S., Katz, J., Wang, N.: Efficient Signature Schemes with Tight Reductions to the Diffie-Hellman Problems. Journal of Cryptology 20(4), 493–514 (2007)
22. Herranz, J., Hofheinz, D., Kiltz, E.: Some (in)sufficient conditions for secure hybrid encryption. Inf. Comput. 208(11), 1243–1257 (2010)
23. Herzog, J.C., Liskov, M., Micali, S.: Plaintext Awareness via Key Registration. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 548–564. Springer, Heidelberg (2003)
24. Jakobsson, M.: A Practical Mix. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 448–461. Springer, Heidelberg (1998)
25. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) ACM Conference on Computer and Communications Security, pp. 155–164. ACM (2003)
26. Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: A New Randomness Extraction Paradigm for Hybrid Encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 590–609. Springer, Heidelberg (2009)
27. Kurosawa, K., Desmedt, Y.G.: A New Paradigm of Hybrid Encryption Scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
28. Kurosawa, K., Matsuo, T.: How to Remove MAC from DHIES. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 236–247. Springer, Heidelberg (2004)
29. Lipmaa, H.: On the CCA1-Security of Elgamal and Damgård's Elgamal. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt 2010. LNCS, vol. 6584, pp. 18–35. Springer, Heidelberg (2011)
30. Mohassel, P.: A Closer Look at Anonymity and Robustness in Encryption Schemes. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 501–518. Springer, Heidelberg (2010)
31. Okamoto, T., Pointcheval, D.: The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In: Kim, K.-C. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
32. Pass, R.: On Deniability in the Common Reference String and Random Oracle Model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003)

33. Pointcheval, D., Stern, J.: Security Proofs for Signature Schemes. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
34. Pointcheval, D., Stern, J.: Security Arguments for Digital Signatures and Blind Signatures. Journal of Cryptology 13(3), 361–396 (2000)
35. Raimondo, M.D., Gennaro, R., Krawczyk, H.: Deniable authentication and key exchange. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security - CCS 2006, pp. 400–409. ACM (2006)
36. Sako, K.: An Auction Protocol Which Hides Bids of Losers. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 422–432. Springer, Heidelberg (2000)
37. Schnorr, C.-P.: Efficient Signature Generation by Smart Cards. Journal of Cryptology 4(3), 161–174 (1991)
38. Schnorr, C.-P., Jakobsson, M.: Security of Signed ElGamal Encryption. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 73–89. Springer, Heidelberg (2000)
39. Y. Seurin and J. Treger. A Robust and Plaintext-Aware Variant of Signed ElGamal Encryption. Full version of this paper. Available from the authors or from, `http://eprint.iacr.org`.
40. Shoup, V.: OAEP Reconsidered. Journal of Cryptology 15(4), 223–249 (2002)
41. Shoup, V., Gennaro, R.: Securing Threshold Cryptosystems against Chosen Ciphertext Attack. Journal of Cryptology 15(2), 75–96 (2002)
42. Tsiounis, Y., Yung, M.: On the Security of ElGamal Based Encryption. In: Imai, H., Zheng, Y. (eds.) PKC 1998. LNCS, vol. 1431, pp. 117–134. Springer, Heidelberg (1998)
43. Wikström, D.: Simplified Submission of Inputs to Protocols. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 293–308. Springer, Heidelberg (2008)

Efficient Public Key Cryptosystem Resilient to Key Leakage Chosen Ciphertext Attacks

Shengli Liu[1,4,⋆], Jian Weng[2,5,⋆⋆], and Yunlei Zhao[3,4,⋆⋆⋆]

[1] Dept. of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
slliu@sjtu.edu.cn

[2] Department of Computer Science, Emergency Technology Research Center
of Risk Evaluation and Prewarning on Public Network Security, Jinan University
cryptjweng@gmail.com

[3] Software School, Fudan University, Shanghai 201203, China
ylzhao@fudan.edu.cn

[4] State Key Laboratory of Information Security, Institute of
Information Engineering, Chinese Academy of Sciences, Beijing 100093

[5] Shanghai Key Laboratory of Scalable Computing and Systems, Shanghai

Abstract. Leakage-resilient public key encryption (PKE) schemes are designed to resist "memory attacks", i.e., the adversary recovers the cryptographic key in the memory adaptively, but subject to constraint that the total amount of leaked information about the key is bounded by some parameter λ. Among all the IND-CCA2 leakage-resilient PKE proposals, the leakage-resilient version of the Cramer-Shoup cryptosystem (CS-PKE), referred to as the KL-CS-PKE scheme proposed by Naor and Segev in Crypto09, is the most practical one. But, the key leakage parameter λ and plaintext length m of KL-CS-PKE are subject to $\lambda + m \le \log q - \omega(\log \kappa)$, where κ is security parameter and q is the prime order of the group on which the scheme is based. Such a dependence between λ and m is undesirable. For example, when λ (resp., m) approaches to $\log q$, m (resp., λ) approaches to 0.

In this paper, we designed a new variant of CS-PKE that is resilient to key leakage chosen ciphertext attacks. Our proposal is $\lambda \le \log q - \omega(\log \kappa)$ leakage-resilient, and the leakage parameter λ is independent of the plaintext space that has the constant size q (exactly the same as that in CS-PKE). The performance of our proposal is almost as efficient as the original CS-PKE. As far as we know, this is the first leakage-resilient CS-type cryptosystem whose plaintext length is independent of the key leakage parameter, and is also the most efficient IND-CCA2 PKE scheme resilient to up to $\log q - \omega(\log \kappa)$ leakage.

[⋆] Funded by NSFC (No. 61170229, 61133014), Innovation Project (No.12ZZ021) of Shanghai Municipal Education Commission, and Specialized Research Fund (No.20110073110016) for the Doctoral Program of Higher Education.

[⋆⋆] Funded by NSFC No. 61272413, and the Opening Project of Shanghai Key Laboratory of Integrate Administration Technologies for Information Security under Grant No. AGK201100.

[⋆⋆⋆] Contact author, funded by NSFC (No. 61070248, 61272012), and Innovation Project (No.12ZZ013) of Shanghai Municipal Education Commission.

1 Introduction

IND-CCA2 security is well-accepted as the standard security notion in public key encryption. However, the security model does not consider many realistic attacks, like side-channel attacks[2,3,16,17]. These attacks are very powerful since they exploit information leakage of physical implementations. Lots of works were devoted to the formalization of side-channel attacks. The framework proposed by Micali and Reyzin [18] assumed that only computation leaks information, and constructions of cryptographic primitives resilient to computational information leakage have been presented in [22,21,13,12,18]. In 2008, Halderman et.al. [15] observed that "cold boot attacks" make possible for the adversary to recover some bits of the cryptographic key in the memory, with momentary physical access to the DRAM. In 2008, Akavia, Goldwasser and Vaikuntanathan[1] recognized these attacks as a new class of side-channel attacks, namely "memory attacks". They also gave a new framework of modeling "key leakage" in a "memory attack". In such an attack, the adversary is able to choose arbitrary functions of the cryptographic key adaptively, and obtains the outputs of these functions. The only constraint is that the total amount of leaked information about the key is bounded by some parameter λ. Following this general framework, lots of key-leakage resilient public key encryption (KL-PKE) schemes were proposed in the past a few years.

1.1 Related Work

In 2005, Akavia et.al.[1] showed that the lattice-based PKE scheme proposed by Regev [23] is resilient to any leakage of $L/\text{polylog}(L)$ bits, with L the bit length of the secret key. In 2009, Naor and Segev [20] distinguished between chosen-plaintext key leakage attacks and chosen-ciphertext key leakage attacks. They suggested that any PKE based on *hash proof systems* [6] can be made secure against chosen-plaintext key leakage attacks (IND-KL-CPA) with help of randomness extractors. In the extended framework of chosen-ciphertext key leakage attacks (IND-KL-CCA2), Naor and Segev proved that Naor-Yung 's "double encryption" paradigm works as well. The requirement is that the two PKE components are IND-KL-CPA secure, and non-interactive zero-knowledge (NIZK) defines a language that the two PKEs encrypt the same plaintext. Naor-Yung 's paradigm achieves key leakage λ up to $L(1 - o(1))$.

The lack of efficient implementations of NIZK makes Naor-Yung 's paradigm less practical. In 2010, Dodis et.al. [10] exploited more efficient ways to achieve IND-KL-CCA2 security. They proposed a new concept of *true-simulation extractable* NIZK arguments, and gave a construction of IND-KL-CCA2-secure PKE from an IND-KL-CPA secure one together with a strong *f-true-simulation extractable* NIZK argument. The key leakage λ is also up to $L(1 - o(1))$.

Among all the proposals to achieve IND-KL-CCA2 security, the variant of Cramer-Shoup cryptosystem presented in [20] is the most practical one. For presentation simplicity, we will call the original IND-CCA2 secure Cramer-Shoup cryptosystem "CS-PKE" and the IND-CCA2 leakage-resilient Cramer-Shoup scheme proposed in [20] "KL-CS-PKE". The KL-CS-PKE follows the original

CS-PKE and uses extractors to deal with key leakage. Let $\mathbb{G}$ be a group of prime order q, the CS-PKE scheme is roughly recalled as follows. The secret key is $(x_1, x_2, y_1, y_2, z_1, z_2)$, all uniformly at random from $\mathbb{Z}_q^*$, and the corresponding public key is $c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^{z_1} g_2^{z_2}$. The encryption (decryption) algorithm creates three ephemeral keys $(c^r, d^r, h^r) = (u_1^{x_1} u_2^{x_2}, u_1^{y_1} u_2^{y_2}, u_1^{z_1} u_2^{z_2})$ with r taken uniformly at random from $\mathbb{Z}_q$. Two of them, $(c^r, d^r) = (u_1^{x_1} u_2^{x_2}, u_1^{y_1} u_2^{y_2})$ are used to ensure (check) the consistence of the ciphertext. The last one $h^r = u_1^{z_1} u_2^{z_2}$ is used to mask (recover) the plaintext.

Encryption	Decryption
$u_1 = g_1^r, u_2 = g_2^r, r \in \mathbb{Z}_q^*, s \in \{0,1\}^t$;	$\alpha = T(u_1, u_2, e, s)$;
$e = M \oplus \mathsf{Ext}(h^r, s)$;	If $v \neq u_1^{x_1 + y_1 \alpha} u_2^{x_2 + y_2 \alpha}$, output $\perp$;
$\alpha = T(u_1, u_2, e, s)$; $\quad v = c^r d^{r\alpha}$;	otherwise $M = e \oplus \mathsf{Ext}(u_1^{z_1} u_2^{z_2}, s)$;
Output (u_1, u_2, s, e, v).	Output M.

Fig. 1. The KL-CS-PKE Scheme

The key leakage of CS-PKE brings two effects: (1) The information leakage related to (x_1, x_2, y_1, y_2) will deteriorate the probability of rejecting inconsistent ciphertexts, hence making the security reduction looser than that of CS-PKE. The information leakage related to (z_1, z_2) makes it unsuitable to mask the plaintext directly. To solve these problems, Naor and Segev employed an extractor Ext to distill a random shorter key from the ephemeral key $u_1^{z_1} u_2^{z_2}$ that is in turn used to mask the plaintext. The bit length of plaintext is then determined by the length of the distilled random key. As for the proof of the IND-KL-CCA2 security, the key observation (with some careful calculation) is that the average min-entropy of (z_1, z_2) conditioned on all the information leaked is upper bounded by $\log q - \lambda$ bits. According to randomness extraction theory, on a security parameter κ, the distilled randomness, hence the plaintext, both are upper bounded by $\log q - \lambda - \omega(\log \kappa)$ bits, which is much shorter than that in the CS-PKE. The KL-CS-PKE scheme proposed by Naor and Segev [20] is recalled in Figure 1, where the public and secret keys and system parameters remain the same as in CS-PKE, and T is a target collision resistant hash function.

1.2 Our Contributions

Based on the original Cramer-Shoup cryptosystem (CS-PKE) [7] and the KL-CS-PKE variant [20], we propose a new leakage-resilient CCA2-secure PKEs. However, we essentially follow a new line.

Like CS-PKE and KL-CS-PKE, the secret key includes $(x_1, x_2), (y_1, y_2)$, (z_1, z_2) and the public key is given by $c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^{z_1} g_2^{z_2}$. The new idea is that all the three parts of secret key, namely (x_1, x_2), (y_1, y_2) and

(z_1, z_2), are involved both in the ciphertext consistence check and the random distillation to mask plaintexts. A key observation is: the three parts of secret key altogether imply larger average min-entropy, even conditioned on all the leaked information bounded by $\lambda \leq \log q - \omega(\log \kappa)$. Larger min-entropy implies more randomness can be distilled. On the other hand, we use a special universal hash function (i.e., $H_s(a, b) = a \cdot b^s$, for $a, b \in \mathbb{G}$ and $s \in \mathbb{Z}_q^*$) as an extractor, where $a = (cd)^r$ and $b = h^r$ for $r \in \mathbb{Z}_q^*$ with our proposal, which allows plaintext space to be $\mathbb{G}$, and makes the security proof neat and tighter. The actual design of our proposal was also carefully guided by the underlying analysis, particularly for ensuring non-zero matrix determinant. Our proposal is briefly described in Figure 2, where the framed texts mark the differences between KL-CS-PKE and our proposal.

Our proposal is almost efficient as the CS-PKE, and we show it is $\lambda \leq \log q - \omega(\log \kappa)$ leakage resilient. In addition, it has the following two advantages:

– The plaintext space is the group $\mathbb{G}$, enjoying a constant size q. It is independent of the leakage parameter λ.
– The security reduction is tighter than that of KL-CS-PKE [20].

Encryption	Decryption
$u_1 = g_1^r, u_2 = g_2^r, \quad r, s \in \mathbb{Z}_q^*$;	$\alpha = T(u_1, u_2, e, s)$;
$\boxed{e = M \cdot (cd)^r \cdot h^{rs};}$	If $\boxed{v \neq u_1^{x_1+y_1\alpha+z_1} u_2^{x_2+y_2\alpha+z_2},}$ output $\perp$;
$\alpha = T(u_1, u_2, e, s)$;	otherwise
$\boxed{v = (c \cdot h)^r \cdot d^{r\alpha};}$	$\boxed{M = e \cdot u_1^{-(x_1+y_1+z_1 s)} u_2^{-(x_2+y_2+z_2 s)};}$
Output (u_1, u_2, s, e, v)	Output M.

Fig. 2. Our proposal

2 Preliminaries

2.1 Notations and Assumptions

Let $\mathcal{H}$ denote a finite set, $|\mathcal{H}|$ denote the cardinality of the set $\mathcal{H}$, and $h \leftarrow \mathcal{H}$ denote sampling uniformly at random from the set $\mathcal{H}$. If $\mathcal{H}$ is a probability distribution, then $h \leftarrow \mathcal{H}$ denotes sampling h according to the distribution. If $A(\cdot)$ is an algorithm, then $a \leftarrow A(\cdot)$ denotes running the algorithm and obtaining a as an output, which is distributed according to the internal randomness of $A(\cdot)$. A function $f(\lambda)$ is *negligible* if for every $c > 0$ there exists an λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$. Let PPT denote probabilistic polynomial-time.

Definition 1. *Let $\mathcal{H}$ be a set of hash functions, mapping $\mathcal{X}$ to $\mathcal{Y}$. Let $k \leftarrow$ **HGen**(1^κ) denote the index generation algorithm, which on input the security parameter κ outputs an index k that in turn determines a hash function $H_k \in$*

$\mathcal{H}$. $\mathcal{H}$ is target collision-resistant if any PPT adversary $\mathcal{A}$ has negligible (in κ) advantage $\boldsymbol{Adv}^{TCR}_{\mathcal{H},\mathcal{A}}(\kappa)$, which is defined as

$$\Pr\left[H_k(x) = H_k(x'), x \neq x' \mid k \leftarrow \boldsymbol{HGen}(1^\kappa); x \leftarrow \mathcal{X}; x' \leftarrow \mathcal{A}(H_k, x)\right].$$

Definition 2. *Let $\mathbb{G}$ be a group of prime order q, which is determined by some security parameter κ (normally, κ is just set to be the bit length of q, i.e, $\log q$)). Define two distributions $\mathcal{D} = \{(g, g^x, g^y, g^{xy})\}$ and $\mathcal{R} = \{(g, g^x, g^y, g^z)\}$, where $x, y, z \leftarrow \mathbb{Z}_q$. The DDH problem is to distinguish the two distributions. A PPT distinguisher $\mathcal{A}$'s advantage is defined as*

$$\boldsymbol{Adv}^{DDH}_{\mathbb{G},\mathcal{A}}(\kappa) = \left|\Pr\left[\mathcal{A}((g, X, Y, Z) \leftarrow \mathcal{D}) = 1\right] \right| - \Pr\left[\mathcal{A}((g, X, Y, Z) \leftarrow \mathcal{R}) = 1\right] |.$$

The DDH assumption means the advantage of any PPT $\mathcal{A}$ is negligible (in κ).

Lemma 1. *[24]* **(Difference Lemma)** *Let A, B, F be events defined over some probability distribution, and $A \wedge \neg F = B \wedge \neg F$. Then $|\Pr[A] - \Pr[B]| \leq \Pr[F]$.*

2.2 Statistical Distance, Min-Entropy, and Leftover Hash Lemma

Definition 3. *[5,11] Let $X \in \mathcal{X}, Y \in \mathcal{Y}$ be two random variables. The min-entropy of X is defined as $\boldsymbol{H}_\infty(X) := -\log \max_{x\in\mathcal{X}} \Pr[X = x]$. The average min-entropy of X given Y is defined as the logarithm of the average guessing probability of X given Y, i.e., $\tilde{\boldsymbol{H}}_\infty(X|Y) := -\log\left(\mathbb{E}_{y\leftarrow\mathcal{Y}}[2^{-\boldsymbol{H}_\infty(X|Y=y)}].\right)$ The statistic distance of two distributions of variable X and Y, both of which take values from $\mathcal{X}$, is defined as $SD(X, Y) = \frac{1}{2}\sum_{a\in\mathcal{X}} |\Pr[X = a] - \Pr[Y = a]|$.*

Lemma 2. *[11] Let Y, Z be random variables and Y takes r possible values. Then $\tilde{\boldsymbol{H}}_\infty(X|(Y, Z)) \geq \tilde{\boldsymbol{H}}_\infty(X|Z) - r$.*

Definition 4. *[11] A function $\mathsf{Ext} : \mathcal{X} \times \{0, 1\}^t \to \mathcal{Y}$ is an average-case (l, δ)-strong extractor if for random variables X, Z such that $X \in \mathcal{X}$ and $\tilde{\boldsymbol{H}}_\infty(X|Z) \geq l$, the formula $SD((\mathsf{Ext}(X, R), R, Z), (U_\mathcal{Y}, R, Z)) \leq \delta$ holds, where R is uniformly chosen from $\{0, 1\}^t$, and $U_\mathcal{Y}$ denotes a uniform distribution over $\mathcal{Y}$.*

Definition 5. *[4,26]* **(Universal hash)** *A family of functions $\{H_k : \mathcal{X} \to \mathcal{Y}\}_{k\in\mathcal{K}}$ is universal if $Pr_{k\leftarrow\mathcal{K}}[H_k(x_1) = H_k(x_2)] \leq \frac{1}{|\mathcal{Y}|}$ for all distinct $x_1, x_2 \in \mathcal{X}$.*

Example 1. [25] The family of functions $\{H_{k_1,k_2,\cdots,k_l} : \mathbb{Z}_q^{l+1} \to \mathbb{Z}_q, k_i \in \mathbb{Z}_q, i = 1, 2, \cdots, l\}$ is universal, where $H_{k_1,k_2,\cdots,k_l}(x_0, x_1, \cdots, x_l) = x_0 + k_1x_1 + \cdots + k_lx_l$. All operations are in the prime field $\mathbb{F}_q$.

The fact that a multiplicative group $\mathbb{G}$ of prime order q is isomorphic to $(\mathbb{Z}_q, +)$ gives us another family of universal hash functions.

Example 2. Let $\mathbb{G}$ be a multiplicative group of prime order q, and $g \in \mathbb{G}$, $g \neq 1$. The family of functions $\{H_{k_1,k_2,\cdots,k_l} : \mathbb{G}^{l+1} \to \mathbb{G}\}_{k_i\in\mathbb{Z}_q, i=1,2,\cdots,l}$ is universal, where $H_{k_1,k_2,\cdots,k_l}(g_0, g_1, \cdots, g_l) = g_0 \cdot g_1^{k_1} \cdot \ldots \cdot g_l^{k_l}$.

Lemma 3. *[11] **(Leftover Hash Lemma and Generalized version)** Let $\{H_k : \mathcal{X} \to \mathcal{Y}\}_{k \in \mathcal{K}}$ be a family of universal hash functions. Let $U_\mathcal{Y}$ denote a uniform distribution over $\mathcal{Y}$. For any random variables $X \leftarrow \mathcal{X}$, $Z \leftarrow \mathcal{Z}$, and $K \leftarrow \mathcal{K}$, $SD((H_k(X), K), (U_\mathcal{Y}, K)) \leq \frac{1}{2}\sqrt{P_c(X)|\mathcal{Y}|} \leq \frac{1}{2}\sqrt{2^{-\boldsymbol{H}_\infty(X)}|\mathcal{Y}|}$, and $SD((H_k(X), K, Z), (U_\mathcal{Y}, K, Z)) \leq \frac{1}{2}\sqrt{2^{-\tilde{\boldsymbol{H}}_\infty(X|Z)}|\mathcal{Y}|}$.*

The leftover hash lemma shows that a family of universal hash functions $\{H_k : \mathcal{X} \to \mathcal{Y}\}_{k \in \mathcal{K}}$ is able to play the role of an average-case (l, δ)-Extractor $\mathsf{Ext} : \mathcal{X} \times \mathcal{K} \to \mathcal{Y}$, with $\log |\mathcal{Y}| \leq l - 2\log(1/\delta) + 2$.

2.3 Key Leakage Chosen-Ciphertext Security (KL-CCA2)

A public key encryption scheme (relative to a plaintext space $\mathcal{M}$) consists of three PPT algorithms, namely (KeyGen, Enc, Dec). The key generation algorithm KeyGen takes as input a security parameter κ, output a public/privat key pair (pk, sk). The encryption algorithm Enc takes as input a public key pk and a plaintext $m \in \mathcal{M}$ and outputs a ciphertext c. The decryption algorithm Dec takes as input a private key sk and a ciphertext c, and outputs a plaintext m or the special symbol $\perp$ meaning that the ciphertext is inconsistent.

It requires that decryption "undoes" encryption for all $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$ and for $m \in \mathcal{M}$.

```
$\mathbf{Expt}^{\mathbf{IND\text{-}KL\text{-}CCA2}}_{\mathbf{PKE},\mathcal{A}}(b)$
1. $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$
2. $(M_0, M_1, state) \leftarrow \mathcal{A}_1^{\mathbf{LK}(sk), \mathbf{Dec}(sk,\cdot)}(pk)$ with $|M_0| = |M_1|$
3. $C^* \leftarrow \mathbf{Enc}_{pk}(M_b)$
4. $b' \leftarrow \mathcal{A}_2^{\mathbf{Dec}_{\neq C^*}(sk,\cdot)}(C^*, state)$
5. Output $b'$
```

The indistinguishability-based definition of chosen-ciphertext security in the key leakage setting (IND-KL-CCA2) is defined by the experiment with adversary $\mathcal{A}$, i.e., $\mathbf{Expt}^{\mathbf{IND\text{-}KL\text{-}CCA2}}_{\mathbf{PKE},\mathcal{A}}(b)$, where $\mathbf{Dec}(sk, \cdot)$ is a decryption oracle and $\mathbf{KL}(sk)$ is a key leakage oracle. The adversary can adaptively query $\mathbf{KL}(sk)$ with any function $f_i, i \geq 1$, and then gets back $f_i(sk)$. Let $\mathbf{Dec}_{\neq C^*}(sk, \cdot)$ denote a decryption oracle who decrypts any ciphertext other than C^*.

The advantage of the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the above experiment is defined as $\mathsf{Adv}^{\mathbf{IND\text{-}KL\text{-}CCA2}}_{\mathsf{PKE},\mathcal{A}}(\kappa) := \left|\Pr[b = b'] - \frac{1}{2}\right|$.

Definition 6. (IND-KL-CCA2 Security). *A public key encryption scheme PKE=(KeyGen, Enc, Dec) is (m, λ, ϵ)-secure against a-posteriori key-leakage chosen-ciphertext attacks, if for sufficiently large κ it holds:*

– *the plaintext space is of size* 2^m,
– *the sum of output lengths of all the queried leakage functions is bounded by* λ,
– *any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *has* $\mathsf{Adv}^{\textbf{\textit{IND-KL-CCA2}}}_{\mathsf{PKE},\mathcal{A}}(\kappa) \leq \epsilon$.

2.4 Key-Leakage Resilient Cramer-Shoup PKE (KL-CS-PKE)

Now we review the key-leakage resilient Cramer-Shoup PKE (KL-CS-PKE) scheme proposed in [20].

Key Generation: On input the security parameter κ, the key generation algorithm generates a group $\mathbb{G}$ of prime order q with two generators g_1, g_2. Choose an average-case $(\log q - \lambda, \delta)$-Exactor $\mathsf{Ext} : \mathbb{G} \times \{0,1\}^t \to \{0,1\}^m$. Choose $x_1, x_2, y_1, y_2, z_1, z_2 \leftarrow \mathbb{Z}_q^*$. Compute $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$ and $h = g_1^{z_1} g_2^{z_2}$. Let T be a Target Collision Resistant (TCR) hash function, randomly chosen from a TCR family $\mathcal{T}$. Output the public key and the secret key as $PK = \{\mathbb{G}, q, g_1, g_2, c, d, h, T, \mathsf{Ext}\}$, $\quad SK = \{PK, x_1, x_2, y_1, y_2, z_1, z_2\}$.

Encryption: The encryption algorithm encrypts an m-bit plaintext M with the public key PK to obtain the corresponding ciphertext $C = (u_1, u_2, e, s, v)$ as follows. Choose random elements $r \leftarrow \mathbb{Z}_q^*$ and $s \leftarrow \{0,1\}^t$. Compute

$$u_1 = g_1^r, u_2 = g_2^r, e = M \oplus \mathsf{Ext}(h^r, s), \alpha = T(u_1, u_2, s, e), v = c^r d^{r \cdot \alpha}.$$

Decryption: The decryption algorithm decrypts a ciphertext $C = (u_1, u_2, e, s, v)$ with the secret key $\{x_1, x_2, y_1, y_2, z_1, z_2\}$ to obtain the corresponding plaintext M as follows. The consistency of the ciphertext is checked by $v = u_1^{x_1 + y_1 \alpha} u_2^{x_2 + y_2 \alpha}$, where $\alpha = T(u_1, u_2, e, s)$. If the ciphertext is not consistent, output "$\perp$", otherwise compute $M = e \oplus \mathsf{Ext}(u_1^{z_1} u_2^{z_2}, s)$.

Choose a proper universal hash function as $(\log q - \lambda, \delta)$ extractor, then $\delta = \frac{2^{(\lambda+m)/2-1}}{\sqrt{q}}$. We have the following theorem for the KL-CS-PKE scheme.

Theorem 1. *[20] Let q be the prime order of the group $\mathbb{G}$ on which the KL-CS-PKE scheme in [20] is based, and $Q(\kappa)$ be the number of decryption queries. Then the KL-CS-PKE is (m, λ, ϵ)-IND-KL-CCA2 secure, where $m \leq \log q - \lambda - \omega(\log \kappa)$, and*

$$\epsilon \leq \mathbf{Adv}^{DDH}_{\mathbb{G},\mathcal{A}'}(\kappa) + \mathbf{Adv}^{TCR}_{\mathcal{H},\mathcal{A}''}(\kappa) + \frac{2^\lambda Q(\kappa)}{q - Q(\kappa)} + \frac{2^{(\lambda+m)/2-1}}{\sqrt{q}},$$

where $\mathcal{A}'$ and $\mathcal{A}''$ are PPT algorithms, derived from the assumed algorithm $\mathcal{A}$ against the IND-KL-CCA2 security of KL-CS-PKE, for breaking the DDH and TCR assumptions respectively.

The KL-CS-PKE scheme proposed in [20] follows the line of the original Cramer-Shoup cryptosystem. The secret key is divided into three parts: (x_1, x_2), (y_1, y_2) and (z_1, z_2). The first two parts (x_1, x_2), (y_1, y_2) are used to check the consistence of the ciphertext, and the last part (z_1, z_2) is used to generate an ephemeral key $g_1^{rz_1} g_2^{rz_2}$ from which a shorter random string is distilled by an extractor to mask the plaintext. Compared to the original IND-CCA2 secure CS-PKE, the λ-bit key leakage has two effects. It may occur in (x_1, x_2) and (y_1, y_2), which deteriorates the probability that an invalid ciphertext, where $u_1 = g_1^{r_1}$ and $u_2 = g_2^{r_2}$ for $r_1 \neq r_2$, is accepted as a consistent one (i.e., passing the consistence check in decryption). The probability is $1/q$ in the original CS-PKE, but increases to $2^\lambda/q$ in KL-CS-PKE. The λ-bit key leakage may occur in (z_1, z_2), which shrinks the length of the output of the extractor (consequently the length of the plaintext) to $m \leq \log q - \lambda - \omega(\log \kappa)$.

3 New Variant of Cramer-Shoup Cryptosystem with IND-KL-CCA2 Security

As contrast to the well-separated functionality of (x_1, x_2, y_1, y_2) and (z_1, z_2), we propose to employ all the three parts of secret key, namely (x_1, x_2), (y_1, y_2) and (z_1, z_2), both in the ciphertext consistence check and the random distillation. On the one hand, there are more uncertainties (measured by average min-entropy) about $(x_1, x_2, y_1, y_2, z_1, z_2)$ to an adversary, and this makes possible for an extractor to distill more randomness to mask plaintexts. On the other hand, the random distillation is implemented by a special universal hash function $H_s(a, b) = a \cdot b^s$ as extractor, where $s \leftarrow \mathbb{Z}_q^*$ and $a = (cd)^r$ and $b = h^r$ with our proposal, which is defined in Example 2 with $l = 1$. This allows plaintext space to be $\mathbb{G}$, and makes the security analysis neat and tighter. The actual design of our proposal was also carefully guided by the underlying analysis, particularly for ensuring non-zero matrix determinant.

The new variant of Cramer-Shoup scheme consists of three PPT algorithms, PKE=(KeyGen, Enc, Dec), as shown below in details.

Key Generation $(PK, SK) \leftarrow \textsf{KeyGen}(1^\kappa)$**:** On input the security parameter 1^κ, the key generation algorithm generates a group $\mathbb{G}$ of prime order q with two generators g_1, g_2. Choose $x_1, x_2, y_1, y_2, z_1, z_2 \in \mathbb{Z}_q^*$. Compute $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$ and $h = g_1^{z_1} g_2^{z_2}$. Let T be a Target Collision Resistant (TCR) hash function, randomly chosen from a TCR family $\mathcal{T}$. Output the public key $PK = \{\mathbb{G}, q, g_1, g_2, c, d, h, T\}$ and the secret key $SK = \{PK, x_1, x_2, y_1, y_2, z_1, z_2\}$.

Encryption $C \leftarrow \textsf{Enc}(PK, M)$**:** The encryption algorithm encrypts a plaintext $M \in \mathbb{G}$ with the public key PK to obtain the corresponding ciphertext $C = (u_1, u_2, e, s, v)$ as follows. Choose random elements $r, s \leftarrow \mathbb{Z}_q^*$. Compute
- $u_1 = g_1^r$, $u_2 = g_2^r$, $e = M \cdot ((c \cdot d)^r \cdot h^{r \cdot s})$,
- $\alpha = T(u_1, u_2, s, e)$, $v = (c \cdot h)^r \cdot d^{r \cdot \alpha}$.

Decryption $M \leftarrow \textsf{Dec}(SK, C)$**:** The decryption algorithm decrypts a ciphertext $C = (u_1, u_2, e, s, v)$ with the secret key $\{x_1, x_2, y_1, y_2, z_1, z_2\}$ to obtain the corresponding plaintext M as follows.

– The consistency of the ciphertext is checked by

$$v = u_1^{x_1+y_1\cdot\alpha+z_1} \cdot u_2^{x_2+y_2\cdot\alpha+z_2}, \tag{1}$$

where $\alpha = T(u_1, u_2, e, s)$. If Eq.(1) does not hold, output $\perp$, otherwise go to the next step.

– Compute $M = \frac{e}{(u_1^{x_1+y_1+z_1 s} \cdot u_2^{x_2+y_2+z_2 s})}$.

The correctness of the scheme comes from the facts that $(c\cdot h)^r \cdot d^{r\cdot\alpha} = u_1^{x_1+y_1\alpha+z_1} \cdot u_2^{x_2+y_2\alpha+z_2}$ and $c^r \cdot d^r \cdot h^{sr} = u_1^{x_1+y_1+z_1 s} \cdot u_2^{x_2+y_2+z_2 s}$.

The plaintext space of our proposal is $\mathbb{G}$ instead of the bit strings. To remove this restriction, $\mathbb{G}$ can be chosen as the subgroup of quadratic residues of $\mathbb{F}_p$, where $p = 2q + 1$, and q is a Sophie German prime, as suggested in [7].

Theorem 2. *The above scheme is* $(\log q, \lambda, \epsilon)$*-IND-KL-CCA2 secure public key encryption scheme. Here* q *is the prime order of the group* $\mathbb{G}$ *that* PKE *is based on,* $\lambda \leq \log q - \omega(\log \kappa)$ *(more precisely,* $\lambda \leq \log q - 2\log\frac{1}{\delta} + 2$ *where* $\delta = \frac{2^{\lambda/2-1}}{\sqrt{q}}$*)) and*

$$\epsilon \leq \mathbf{Adv}_{\mathbb{G},\mathcal{A}'}^{DDH}(\kappa) + \mathbf{Adv}_{T,\mathcal{A}''}^{TCR}(\kappa) + \frac{2^\lambda Q(\kappa)}{q - Q(\kappa)} + \frac{2^{\lambda/2-1}}{\sqrt{q}},$$

where $Q(\kappa)$ *is the number of decryption queries, and* $\mathcal{A}'$ *and* $\mathcal{A}''$ *have the same meanings as in Theorem 1.*

Before going into the formal proof, we briefly give a high-level description of the following gamed-based security proof.

Let $C^* = (u_1^*, u_2^*, e^*, s^*, v^*) = \mathsf{Enc}(PK, M_b)$ be the challenge ciphertext in the IND-KL-CCA2 game. The aim is to prove that any PPT adversary cannot determine whether M_b is M_0 or M_1 except with probability 1/2 biased with a negligible probability. A ciphertext $C = (u_1, u_2, e, s, v)$ is called a valid ciphertext if (g_1, g_2, u_1, u_2) is a DDH tuple, otherwise, it is called an invalid one. A ciphertext $C = (u_1, u_2, e, s, v)$ is called consistent if it can pass the consistence check with Eq.(1) in the decryption algorithm. First of all, the challenge ciphertext C^* is changed into an invalid but consistent ciphertext. This can be easily done with secret key $(x_1, x_2, y_1, y_2, z_1, z_2)$. The adversary is infeasible to detect such a change in C^* due to the DDH assumption.

As to an adversary $\mathcal{A}$, the information about the secret key is learned with the public key elements (c, d, h), the challenge ciphertext $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$ and the λ-bit leakage. In C^*, the secret key functions in v^* are for consistence check, and in e^* for randomness extraction to mask M_b. Note that $\mathcal{A}$ learns no more information about secret-key (other than what can be publicly derived) by submitting valid ciphertext ciphertexts to the decryption oracle. Next step is to prove that the probability that an invalid ciphertext C $(\neq C^*)$ is accepted as a consistent one, given (c, d, h, C^*) and λ-bit leakage, is negligible. This is guaranteed with the TCR property of T and the average min-entropy of secret key.

The last step is to prove that e^* leaks no information about M_b at all. Note that the value $e^*/M_b = H_{s^*}\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2}\right)$ is just the output of the extractor $H_{s^*}(\cdot,\cdot)$. The remaining thing is to prove that, given (c, d, h, v^*) and the λ-bit leakage, the average min-entropy of the two variables $\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2}\right)$ is at least $2\log q - \lambda$. Then, by the generalized leftover hash lemma (Lemma 3), we have that $m = \log q \leq 2\log q - \lambda - 2\log\frac{1}{\delta} + 2$ and thus $\lambda \leq \log q - 2\log\frac{1}{\delta} + 2$, and $H_{s^*}(\cdot,\cdot)$ is a $(2\log q - \lambda, \delta)$-extractor with $\delta = \frac{2^{\lambda/2-1}}{\sqrt{q}}$. This finally establishes that e^*/M_b is δ-close to uniform distribution, and e^* hides M_b statistically except with negligible probability δ.

Proof. We proceed with a series of games played between a simulator $\mathcal{D}$ and an adversary $\mathcal{A}$, and show that Game i and Game $i+1$ are indistinguishable except with negligible probability, $i = 0, 1, 2, 3, 4, 5$. We define S_i as the event that the adversary $\mathcal{A}$ outputs a correct guess of b.

Game 0: This is the original game. The simulator $\mathcal{D}$ generates the public key PK and secret keys SK with KeyGen, where $PK = \{\mathbb{G}, q, g_1, g_2, c, d, h\}$ and $SK = \{PK, x_1, x_2, y_1, y_2, z_1, z_2\}$. The simulator sends PK to the adversary. For each decryption query C or key leakage function query f_i made by $\mathcal{A}$, the simulator returns $M \leftarrow \mathsf{Dec}(sk, C)$ or $f_i(SK)$ using the secret key SK. Then the adversary $\mathcal{A}$ submits two plaintext M_0, M_1 of equal length to $\mathcal{D}$. The simulator computes the corresponding challenge ciphertext $C^* = \mathsf{Enc}(PK, M_b)$ with $b \leftarrow \{0,1\}$. The adversary $\mathcal{A}$ outputs b', which is the guess of b. Let S_0 be the event that $b' = b$ in Game 0.

Game 1: It is the same as Game 0 except for the generation of the challenge ciphertext C^*. In this game, the simulator $\mathcal{D}$ generates $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$ with its secret key SK as follows.

Choose random elements $r^*, s^* \leftarrow \mathbb{Z}_q^*$. Compute

- $u_1^* = g_1^{r^*}$,
- $u_2^* = g_2^{r^*}$,
- $e^* = M_b \cdot (u_1^*)^{x_1+y_1+z_1 s} \cdot (u_2^*)^{x_2+y_2+z_2 s}$,
- $\alpha^* = T(u_1^*, u_2^*, e^*, s^*)$,
- $v^* = (u_1^*)^{x_1+y_1\alpha^*+z_1}(u_2^*)^{x_2+y_2\alpha^*+z_2}$.

This change is only conceptual, hence $\Pr[S_1] = \Pr[S_0]$.

Game 2: It is the same as Game 1 except for the generation of the challenge ciphertext $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$, where $u_1^* = g_1^{r_1^*}, u_2^* = g_2^{r_2^*}$, with r_1^*, r_2^* chosen uniformly at random from $\mathbb{Z}_q^*$.

Just as done in the analysis of the CS-PKE scheme (say, the original Cramer-Shoup scheme) [7], any difference between Game 1 and Game 2 can be used to build a PPT algorithm $\mathcal{A}'$ to distinguish a DDH tuple from a random tuple. Hence

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathbf{Adv}^{DDH}_{\mathbb{G},\mathcal{A}''}(\lambda).$$

Game 3: It is the same as Game 2 except that the simulator $\mathcal{D}$ applies a special rejection rule. Let F denote the event that there exists a ciphertext $C = (u_1, u_2, e, s, v)$ among the decryption queries such that $(u_1, u_2, e, s) \neq$

(u_1^*, u_2^*, e^*, s^*) but $T(u_1, u_2, e, s) = T(u_1^*, u_2^*, e^*, s^*)$, which means a hash collision occurs. The simulator $\mathcal{D}$ rejects the corresponding queried ciphertext C when F occurs. Then $\Pr[S_2 \wedge \neg F] = \Pr[S_3 \wedge \neg F]$. According to the Difference Lemma of [7] (recalled in Definition 1) and the analysis of CS-PKE [7] and KL-CS-PKE [20], we can construct another PPT algorithm $\mathcal{A}''$, such that

$$|\Pr[S_2] - \Pr[S_3]| \leq \Pr[F] \leq \mathbf{Adv}^{\mathbf{TCR}}_{\mathcal{H},\mathcal{A}''}(\lambda).$$

Game 4: It is the same as Game 3 except that the simulator $\mathcal{D}$ applies a special rejection rule. If $\mathcal{A}$ asks for decryption of an invalid ciphertext $C = (u_1, u_2, e, s, v)$, i.e., (g_1, g_2, u_1, u_2) is not a DDH tuple, the simulator $\mathcal{D}$ rejects with $\perp$ and the game aborts. Let F' be the event that $\mathcal{D}$ outputs $\perp$ for this reason, then $\Pr[S_4|\neg F'] = \Pr[S_3|\neg F']$. According to the Difference Lemma,

$$|\Pr[S_4] - \Pr[S_3]| \leq \Pr[F'].$$

Now we analyze the probability that the event F' occurs.
Before submitting the *first* invalid ciphertext, the adversary's view consists of the public key $(q, g_1, g_2, c, d, h, T)$, the chosen plaintexts (M_0, M_1) and the challenge ciphertext $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$, λ-bit leakage about secret key elements $(x_1, x_2, y_1, y_2, z_1, z_2)$, and the valid ciphertext queries to the decryption oracle and the answers from it (recall that we are considering the view prior to submitting the first invalid ciphertext by the adversary). Let $\beta = \log_{g_1} g_2$. Firstly, note that by submitting valid ciphertexts to the decryption oracle the adversary does not learn any more information on $(x_1, x_2, y_1, y_2, z_1, z_2)$. In fact, by submitting a valid ciphertext the adversary only learns linear combinations of the constraints $\log_{g_1} c = x_1 + \beta x_2$, $\log_{g_1} d = y_1 + \beta y_2$ and $\log_{g_1} h = z_1 + \beta z_2$, which are already known from the public-keys. Also note that $q, g_1, g_2, T, u_1^*, u_2^*, s^*, M_{1-b}$ all are independent of the secret key. Then, what can be learnt about secret key $(x_1, x_2, y_1, y_2, z_1, z_2)$, from c, d, h, v^*, e^*, M_b and λ-bit leakage, can be formulated with the following equations.

$$\log_{g_1} c = x_1 + \beta x_2 \tag{2}$$

$$\log_{g_1} d = y_1 + \beta y_2 \tag{3}$$

$$\log_{g_1} h = z_1 + \beta z_2 \tag{4}$$

$$\log_{g_1} v^* = r_1^* x_1 + r_2^* \beta x_2 + \alpha^* r_1^* y_1 + \alpha^* r_2^* \beta y_2 + r_1^* z_1 + r_2^* \beta z_2 \tag{5}$$

$$\log_{g_1} e^*/M_b = r_1^* x_1 + r_2^* \beta x_2 + r_1^* y_1 + r_2^* \beta y_2 + s^* r_1^* z_1 + s^* r_2^* \beta z_2; \tag{6}$$

$$\lambda\text{-bit leakage of } (x_1, x_2, y_1, y_2, z_1, z_2). \tag{7}$$

Since the secret key elements $(x_1, x_2, y_1, y_2, z_1, z_2)$ are uniformly chosen from $\mathbb{Z}_q^6$, we have

$$\begin{aligned}
&\tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, C^*, M_b, \lambda\text{-leakage}) \\
&= \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, u_1^*, u_2^*, s^*, v^*, e^*/M_b, \lambda\text{-leakage}) \\
&= \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, v^*, e^*/M_b, \lambda\text{-leakage}) \qquad (8)\\
&\geq \log q - \lambda. \qquad (9)
\end{aligned}$$

Eq. (8) follows from the fact that u_1^*, u_2^*, s^* are independently chosen, and (9) according to Lemma 2.

Let $C = (u_1, u_2, e, s, v)$ be the first invalid ciphertext submitted by $\mathcal{A}$. Let $r_1 = \log_{g_1} u_1$ and $r_2 = \log_{g_1} u_2$, then $r_1 \neq r_2$.

1. If $C = C^*$, the simulator $\mathcal{D}$ has already rejected in Game 0.
2. If $C \neq C^*$ but $\alpha = \alpha^*$, the simulator $\mathcal{D}$ has already rejected in Game 3.
3. If $C \neq C^*$ but $\alpha \neq \alpha^*$, and the simulator $\mathcal{D}$ accepts the invalid ciphertext C, then the following linear equations hold.

$$\begin{aligned}
\log_{g_1} c &= x_1 + \beta x_2; \\
\log_{g_1} d &= y_1 + \beta y_2; \\
\log_{g_1} h &= z_1 + \beta z_2; \\
\log_{g_1} v^* &= r_1^* x_1 + r_2^* \beta x_2 + \alpha^* r_1^* y_1 + \alpha^* r_2^* \beta y_2 + r_1^* z_1 + r_2^* \beta z_2; \\
\log_{g_1} e^*/M_b &= r_1^* x_1 + r_2^* \beta x_2 + r_1^* y_1 + r_2^* \beta y_2 + s^* r_1^* z_1 + s^* r_2^* \beta z_2; \\
\log_{g_1} v &= r_1 x_1 + r_2 \beta x_2 + \alpha r_1 y_1 + \alpha r_2 \beta y_2 + r_1 z_1 + r_2 \beta z_2.
\end{aligned} \qquad (10)$$

Equation (10) is reformed to be

$$\begin{pmatrix} 1 & \beta & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \beta & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \beta \\ r_1^* & r_2^*\beta & \alpha^* r_1^* & \alpha^* r_2^* \beta & r_1^* & r_2^* \beta \\ r_1^* & r_2^*\beta & r_1^* & r_2^*\beta & s^* r_1^* & s^* r_2^* \beta \\ r_1 & r_2 \beta & \alpha r_1 & \alpha r_2 \beta & r_1 & r_2 \beta \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \log_{g_1} c \\ \log_{g_1} d \\ \log_{g_1} h \\ \log_{g_1} v^* \\ \log_{g_1} e^*/M_b \\ \log_{g_1} v \end{pmatrix}. \qquad (11)$$

Let $\mathbf{A}$ be the matrix of 6 by 6 in Eq. (11), we have

$$det(\mathbf{A}) = \beta^3 (r_1^* - r_2^*)^2 (r_1 - r_2)(\alpha^* - \alpha)(s^* - 1).$$

Then $det(\mathbf{A}) \neq 0$ except with a negligible probability $1/q$, due to the fact that $r_1^* \neq r_2^*, r_1 \neq r_2, \alpha \neq \alpha^*$ and the random choice of s^*. Different values of v give different solutions for $(x_1, x_2, y_1, y_2, z_1, z_2)$. According to Eq. (9), the adversary guesses $(x_1, x_2, y_1, y_2, z_1, z_2)$ correctly with probability at most $2^\lambda/q$. Therefore, the first invalid ciphertext C is accepted by $\mathcal{D}$ with probability at most $2^\lambda/q$. Similarly, the i-th invalid ciphertext is accepted by $\mathcal{D}$ with probability at most $2^\lambda/(q - i + 1) \leq 2^\lambda/(q - Q(\kappa))$, where $Q(\kappa)$

is the total number of decryption queries. By the union bound, we have $\Pr[F'] \le \frac{2^\lambda Q(\kappa)}{q-Q(\kappa)}$ and

$$|\Pr[S_4] - \Pr[S_3]| \le \Pr[F'] \le \frac{2^\lambda Q(\kappa)}{q-Q(\kappa)}.$$

The number of queries $Q(\kappa)$ is a polynomial in κ, hence $\Pr[F']$ is negligible.

Game 5: It is the same as Game 4 except for the generation of the challenge ciphertext $C^* = (u_1^*, u_2^*, e^*, s^*, v^*)$. The only change is that e^* is replaced with an element $\hat{e}$ chosen uniformly at random from G. Since all the invalid ciphertexts submitted by $\mathcal{A}$ are rejected by the decryption oracle, decryption oracle cannot help $\mathcal{A}$ gain more information about the secret key. The only information related to secret key known by $\mathcal{A}$ is still characterized by the public key elements (c, d, h), the λ-bit leakage, and (v^*, e^*) in C^*.
Next, we will show that e^*/M_b is in fact the output a $(2\log q - \lambda, \delta)$ extractor with $(u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}$ and $(u_1^*)^{z_1}(u_2^*)^{z_2}$ as input. Given the information c, d, h, v^* and the λ-bit leakage, we determine the average min-entropy $\tilde{\mathrm{H}}_\infty\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2} \mid c, d, h, v^*, \lambda\text{-leakage}\right)$.
Let us check the following equations in $x_1, x_2, y_1, y_2, z_1, z_2$.

$$\begin{aligned}
\log_{g_1} c &= x_1 + \beta x_2;\\
\log_{g_1} d &= y_1 + \beta y_2;\\
\log_{g_1} h &= z_1 + \beta z_2;\\
\log_{g_1} v^* &= r_1^* x_1 + r_2^*\beta x_2 + \alpha^* r_1^* y_1 + \alpha^* r_2^* \beta y_2 + r_1^* z_1 + r_2^* \beta z_2;\\
\log_{g_1}\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}\right) &= r_1^* x_1 + r_2^*\beta x_2 + r_1^* y_1 + r_2^*\beta y_2;\\
\log_{g_1}\left((u_1^*)^{z_1}(u_2^*)^{z_2}\right) &= r_1^* z_1 + r_2^*\beta z_2.
\end{aligned} \tag{12}$$

Equivalently,

$$\begin{pmatrix}
1 & \beta & 0 & 0 & 0 & 0\\
0 & 0 & 1 & \beta & 0 & 0\\
0 & 0 & 0 & 0 & 1 & \beta\\
r_1^* & r_2^*\beta & \alpha^* r_1^* & \alpha^* r_2^*\beta & r_1^* & r_2^*\beta\\
r_1^* & r_2^*\beta & r_1^* & r_2^*\beta & 0 & 0\\
0 & 0 & 0 & 0 & r_1^* & r_2^*\beta
\end{pmatrix} \cdot \begin{pmatrix} x_1\\ x_2\\ y_1\\ y_2\\ z_1\\ z_2 \end{pmatrix} = \begin{pmatrix} \log_{g_1} c\\ \log_{g_1} d\\ \log_{g_1} h\\ \log_{g_1} v^*\\ \log_{g_1}\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}\right)\\ \log_{g_1}(u_1^*)^{z_1}(u_2^*)^{z_2} \end{pmatrix}. \tag{13}$$

The determinant of the above matrix is $-\beta^3(\alpha^*-1)(r_1^*-r_2^*)^3 \neq 0$. Given c, d, h, v^*, each pair $\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2}\right)$ determines a unique tuple $(x_1, x_2, y_1, y_2, z_1, z_2)$.
Hence,

$$\tilde{\mathrm{H}}_\infty\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2} | c, d, h, v^*, \lambda\text{-leakage}\right) \tag{14}$$
$$= \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, v^*, \lambda\text{-leakage}) \tag{15}$$
$$\ge \tilde{\mathrm{H}}_\infty((x_1, x_2, y_1, y_2, z_1, z_2)|c, d, h, v^*) - \lambda \ge 2\log q - \lambda, \tag{16}$$

where (15) is from the observation that, *conditioned on* (c, d, h, v^*), the function $(x_1, x_2, y_1, y_2, z_1, z_2) \to ((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2})$ is injective, and applying an injective function to a function preserves its min-entropy, (16) is derived by applying Lemma 2 twice.
Applying the universal hash function $H_{s^*} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ (defined in Example 2 with $l = 1$ and $H_{s^*}(a, b) = a \cdot b^{s^*}$ where $a = (u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}$ and $b = (u_1^*)^{z_1}(u_2^*)^{z_2}$) as a $(2\log q - \lambda, \delta)$ extractor to the two variables $(u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2}$, we have

$$\begin{aligned} e^*/M_b &= H_{s^*}\left((u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}, (u_1^*)^{z_1}(u_2^*)^{z_2}\right) \\ &= (u_1^*)^{x_1+y_1}(u_2^*)^{x_2+y_2}\left((u_1^*)^{z_1}(u_2^*)^{z_2}\right)^{s^*} = (u_1^*)^{x_1+y_1+z_1 s^*}(u_2^*)^{x_2+y_2+z_2 s^*}. \end{aligned}$$

According to the generalized leftover lemma (Lemma 3), the statistical distance between e^* and $\hat{e}$ is given by $SD(e^*, \hat{e}) \leq \frac{1}{2}\sqrt{q \cdot \frac{2^\lambda}{q^2}} = \frac{2^{\lambda/2-1}}{\sqrt{q}}$. Hence $|\Pr[S_5] - \Pr[S_4]| \leq \delta = \frac{2^{\lambda/2-1}}{\sqrt{q}}$. Now that $\hat{e}$ is uniformly distributed, we have $\Pr[S_5] = 1/2$. □

Remark: By putting all the secret-key elements into ciphertext consistency checking and random distillation, and by employing the specific universal hash based extractor, we can have that the average entropy of secret-key given the view of the adversary is at least $2\log q - \lambda$. According to the generalized leftover lemma, and recall that the range of the specific universal hash based extractor is just $\mathbb{G}$ (i.e., the plaintext length $m = \log q$), we have that $m \leq 2\log q - \lambda - 2\log\frac{1}{\delta} + 2$ and thus $\lambda \leq \log q - 2\log\frac{1}{\delta} + 2$. In comparison, the average entropy of secret-key for the KL-CS-PKE scheme proposed by Naor and Segev [20], given the adversary's view, is only at least $\log q - \lambda$. Then, by the $(\log q - \lambda, \delta)$-extractor, it requires that $m \leq \log q - \lambda - 2\log\frac{1}{\delta} + 2$ (equivalently, $m + \lambda \leq \log q - 2\log\frac{1}{\delta} + 2$).

4 Performance Analysis

Now we compare our proposal with the original Cramer-Shoup cryptosystem (CS-PKE) in [7] and the KL-CS-PKE scheme proposed by Naor and Segev in [20] in the following tables.

Let $\epsilon_1 = \mathbf{Adv}_{\mathbb{G},\mathcal{A}'}^{DDH}(\kappa), \epsilon_2 = \mathbf{Adv}_{\mathcal{H},\mathcal{A}''}^{\mathbf{TCR}}(\kappa)$, and $\mathcal{M}$ denote the plaintext space. Let λ be the amount of leakage bits, and $Q(\kappa)$ be the number of decryption queries. Table 1 shows that our proposal gives a larger plaintext space and the security reduction is tighter than KL-CS-PKE in [20]. The plaintext space of our proposal is $\mathbb{G}$, which is independent of the amount of leakage λ. This is exactly like that in the original CS-PKE scheme [7]. In contrast, the KL-CS-PKE scheme in [20] requires that the length of the plaintext is limited by $\log q - \lambda - \omega(\log \kappa)$. Also, on the same leakage parameter λ, the IND-KL-CCA2 security reduction of our proposal is also tighter than that of KL-CS-PKE.

Let "1 Ex" denote a modular exponentiation, which evaluates g^x in group $\mathbb{G}$, and "1 SE" denote a simultaneous modular exponentiation of $g_1^x g_2^y$, which

Table 1. Parameters of CS-PKE, KL-CS-PKE and our proposal

Scheme	$\lvert\mathcal{M}\rvert$	leakage	$\mathsf{Adv}^{\mathbf{IND\text{-}KL\text{-}CCA2}}_{\mathsf{PKE},\mathcal{A}}(1^\kappa)$
CS-PKE [7]	q	—	$\epsilon_1+\epsilon_2+\frac{Q(\kappa)}{q-Q(\kappa)}$
KL-CS-PKE [20]	2^m $(2^m<q/2^\lambda)$	λ $(\lambda\le\log q-m-\omega(\log\kappa))$	$\epsilon_1+\epsilon_2+\frac{2^\lambda Q(\kappa)}{q-Q(\kappa)}+\frac{2^{(\lambda+m)/2-1}}{\sqrt{q}}$ $(m+\lambda\le\log q-\omega(\log\kappa))$
Our proposal	q	λ $(\lambda\le\log q-\omega(\log\kappa))$	$\epsilon_1+\epsilon_2+\frac{2^\lambda Q(\kappa)}{q-Q(\kappa)}+\frac{2^{\lambda/2-1}}{\sqrt{q}}$ $(\lambda\le\log q-\omega(\log\kappa))$

Table 2. Efficiency and ciphertext sizes of CS-PKE, KL-CS-PKE and our proposal

Scheme	KeyGen	Enc	Dec	Ciphertext Size
CS-PKE [7]	3 SE	3 Ex +1 SE	2 SE	$4\mathbb{G}$
KL-CS-PKE [20]	3 SE	3 Ex + 1 SE+ 1 Ext	2 SE	$4\mathbb{G}+t$-bit
Our proposal	3 SE	2 Ex + 2 SE	2 SE	$4\mathbb{G}+\log q$-bit

amounts for about 1.27 exponentiations [19,14,9]. Let "1 Ext" denote an evaluation of the extractor $\mathsf{Ext}:\mathbb{G}\times\{0,1\}^t\to\{0,1\}^m$. Let "1$\mathbb{G}$" denote an element from group $\mathbb{G}$, and "m-bit" denote an m-bit string. In order to achieve the parameter relations among plaintext space, allowed leakage and IND-KL-CCA2 security reduction as specified in Table 1, the needed computational efficiency and ciphertext sizes, among CS-PKE [7], KL-CS-PKE [20] and our proposal, are compared in Table 2. It is shown that our proposal is almost as efficient as CS-PKE. For KL-CS-PKE and our proposal, if both schemes work to encrypt a $\log q$-bit plaintext, KL-CS-PKE can be slightly more efficient than ours. However, in this case, KL-CS-PKE does not allow key leakage any more, as λ approaching to 0 in this case. On the other hand, assuming key leakage approaching to $\log q-\omega(\log\kappa)$, our proposal is significantly more efficient than KL-CS-PKE, as in this case the plaintext size of KL-CS-PKE approaches to 0.

5 Conclusion

As a response to Naor and Segev's calling for further refinement of key leakage resilient variant of Cramer-Shoup Cryptosystem in order to get rid of the dependency between plaintext length m and leakage parameter λ, in this paper we designed a new variant of Cramer-Shoup cryptosystem. With some careful observations and a calculation guided design, our proposal follows a new line: (1) the whole secret key is involved in both ciphertext consistence checking and randomness distillation, and (2) a special universal hashing based extractor is employed (alternatively, randomness extractor is only implicitly used with our proposal). Our scheme is IND-KL-CCA2 secure with a tighter reduction than that of KL-CS-PKE, $\lambda = \log q - \omega(\log \kappa)$ leakage resilient, and the plaintext space is the whole group that the scheme is based on and is independent of the leakage parameter. The performance of our proposal is comparable to the original Cramer-Shoup cryptosystem. As far as we know, this is the first leakage-resilient CS-type cryptosystem whose plaintext length is independent of the key leakage parameter, and is also the most efficient IND-CCA2 PKE scheme resilient to up to $\log q - \omega(\log \kappa)$ leakage. In some sense, our result also further demonstrates the elegance of the original Cramer-Shoup cryptosystem [7] (as well as its key leakage resilient variant by Naor and Segev [20]). As a new IND-KL-CCA2 variant of Cramer-Shoup with independent plaintext length and leakage parameter, our proposal may also be of independent value.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous Hardcore Bits and Cryptography against Memory Attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
2. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
3. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the Importance of Checking Cryptographic Protocols for Faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
4. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. Journal of Computer and System Sciences 18, 143–154 (1979)
5. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. SIAM Journal on Computing 17(2), 230–261 (1988)
6. Cramer, R., Shoup, V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
7. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)
8. Damgård, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-key System. In: Kim, K.-C. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001) (Full version with additional co-author J. B. Nielsen)
9. Dimitrov, V.S., Jullien, G.A., Miller, W.C.: Complexity and Fast Algorithms for Multiexponentiations. IEEE Transactions on Computers 49(2), 141–147 (2000)
10. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient Public-Key Cryptography in the Presence of Key Leakage. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 613–631. Springer, Heidelberg (2010)
11. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM Journal on Computing 38(1), 97–139 (2008)

12. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 293–302 (2008)
13. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-Time Programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
14. Gordon, D.M.: A Survey of Fast Exponentiation Methods. Journal of Algorithms 27(1), 129–146 (1998)
15. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: Proceedings of the 17th USENIX Security Symposium, pp. 45–60 (2008)
16. 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)
17. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
18. Micali, S., Reyzin, L.: Physically observable cryptography. In: Proceedings of the 1st Theory of Cryptography Conference, pp. 278–296 (2004)
19. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography, pp. 617–619. CRC Press (1995)
20. Naor, M., Segev, G.: Public-Key Cryptosystems Resilient to Key Leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
21. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
22. Petit, C., Standaert, F.-X., Pereira, O., Malkin, T., Yung, M.: A block cipher based pseudo random number generator secure against side-channel key recovery. In: Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS), pp. 56–65 (2008)
23. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 84–93 (2005)
24. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. In: IACR Cryptology ePrint Archive, p. 332 (2004)
25. Shoup, V.: A Computational Introduction to Number Theory and Algebra. Cambridge University Press (2005)
26. Wegman, M.N., Carter, J.L.: New hash functions and their use in authentication and set equality. Journal of Computer and System Sciences 22, 265–279 (1981)

Simple, Efficient and Strongly KI-Secure Hierarchical Key Assignment Schemes

Eduarda S.V. Freire[⋆], Kenneth G. Paterson[⋆⋆], and Bertram Poettering[⋆⋆]

Information Security Group,
Royal Holloway, University of London, U.K.

Abstract Hierarchical Key Assignment Schemes can be used to enforce access control policies by cryptographic means. In this paper, we present a new, enhanced security model for such schemes. We also give simple, efficient, and strongly-secure constructions for Hierarchical Key Assignment Schemes for arbitrary hierarchies using pseudorandom functions and forward-secure pseudorandom generators. We compare instantiations of our constructions with state-of-the-art Hierarchical Key Assignment Schemes, demonstrating that our new schemes possess an attractive trade-off between storage requirements and efficiency of key derivation.

1 Introduction

Access control: There are numerous examples where it is desirable to provide differentiated access to data according to an *access control policy.* As an illustration, consider a hospital where doctors are assigned access permission to a set of files containing some personal information in a patient's medical record, depending on their seniority, while nurses, being at a lower level in the hierarchy, have more restricted access to that information. As another example, consider a building management scenario where sensors are installed to capture temperature, humidity, light, motion, sound, or other data. These data have different levels of sensitivity, and access to information of different types might be restricted to different personnel, depending on their roles in the organization. Normal employees would only be able, for example, to access temperature, humidity and light of the floor where they work, while managers of that floor would be able to have access to information related to presence in rooms on that floor, like motion and sound data. The manager of the building would, however, have access to all information for the different floors of the building. As a third example, broadcasters wish to control access to broadcast services in such a way that only paying customers can access the programmes included in the package to which they have subscribed, and nothing else. Other application domains include management of databases containing sensitive information, military and government communication, and protection of industrial secrets. Indeed the field of access control is a healthy sub-discipline of Information Security in its own right.

[⋆] This author supported by CAPES Foundation/Brazil on grant 0560/09-0 and Royal Holloway, University of London.

[⋆⋆] This author supported by EPSRC Leadership Fellowship EP/H005455/1.

Cryptographic enforcement: The use of cryptographic techniques to enforce access control policies for hierarchical structures was first proposed in 1983, by Akl and Taylor [1], who put forward the concept of a *(hierarchical) key assignment scheme (KAS).* Such a scheme is a method to assign some private information and encryption keys to each class in a hierarchy in such a way that the private information assigned to a class, along with some public information, can be used to derive symmetric encryption keys assigned to all classes lower down in the hierarchy. Formally, the hierarchy is modelled as a *partially ordered set (poset)*, each data item is labelled by a class u in the hierarchy, and is encrypted using the encryption key k_u corresponding to that class. Now a user, given access to the private information S_u, can derive the relevant encryption key k_v for any descendant class v, and hence gain access to the data of class v. Since the original paper by Akl and Taylor, a large number of different schemes have been proposed, offering different trade-offs in terms of the amount of public and private storage required and the complexity of key derivation – see for example [2–17]. Many additional issues are addressed in these works: time-dependent constraints, dynamic addition and removal of classes, and revocation, for example. A recent survey of this area by Crampton *et al.* [18] provides a detailed classification and analyses of many of the schemes proposed in the last decades.

Many of the early schemes lacked any formal security analysis, but this shortcoming has been gradually addressed beginning with the work of Atallah *et al.* [8], who proposed two different security notions: security against key recovery attacks (KR-security) and security with respect to key indistinguishability (KI-security). Informally, KR-security captures the notion that an adversary should not be able to compute a key to which it should not have access; whereas in the notion of KI-security, the adversary should not even be able to distinguish between the real key and a random string of the same length. The stronger KI-security notion is important in enabling secure composability for hierarchical key assignment schemes, that is, in achieving the property that any secure key assignment scheme can be safely used alongside any suitably secure encryption scheme.

Our contributions: We first argue that the KI-security notion introduced in [8] needs to be strengthened in order to capture the widest possible range of realistic attacks. In particular, the current model does not allow an adversary to gain access to encryption keys k_v for classes above the target class u, even though these encryption keys might leak through usage and their compromise need not directly lead to a compromise of the private information S_u or encryption key k_u for the target class. We then define a model that provides this additional compromise capability to the adversary, and show that our new model is strictly stronger than the existing KI-security notion. Section 2 contains the details.

We next propose two very simple and efficient hierarchical key assignment schemes for arbitrary posets, and prove them to be secure in the sense of our strengthened security notion. Both of our schemes exploit the chain partition idea recently introduced by Crampton *et al.* [15]. This gives a method of constructing a KAS for an arbitrary access structure (modelled as a poset), represented by a directed acyclic graph $P = (V, E)$, from a KAS for a simple chain (i.e. a KAS for a totally ordered set) by partitioning the poset into chains and building the keys for the more complex scheme for P in a particular way from the keys of the simpler chain KAS. This approach has the nice property that the amount of private storage needed per class is bounded by the *width* of the poset P. This approach was proposed without any formal security analysis in [15], and analysed in some specific cases in [16]. We provide in Section 3 a generic security analysis of this approach, showing that the security of the resulting scheme for $P = (V, E)$ in our strengthened model is equivalent to the security (also in our strengthened model) for the chain scheme. It is worth noting that this construction can support different levels of security or efficiency of key derivation for different subgroups in a hierarchy, by using different schemes in each chain.

This construction enables us to focus on constructing efficient KAS for chain posets in our strengthened model. Our first construction in Section 4 is based only on pseudorandom functions (PRFs), which can be efficiently implemented using, for example, HMAC [19] built using only a cryptographic hash function, or by using an iterated version of the simple and efficient factoring-based construction proposed by Naor and Reingold [20] (wherein the evaluation of the function at a given point is comparable in cost to two modular exponentiations). When implemented in the latter way, our scheme enjoys provable security based on one of the most established concrete intractability assumption used in cryptography, namely the factoring assumption.

Our second construction, in Section 5, is based on any forward-secure pseudorandom generator (FS-PRG). This construction is a generalization and a strengthening of the construction for chains given in [16], which implicitly makes use of the known forward-security of the BBS PRG [21] in order to achieve KI-security. Note that the BBS generator was not originally presented as a stateful generator and its forward-security property was first used in the Blum-Goldwasser cryptosystem [22] and later by Bellare and Yee [23]. An FS-PRG can be obtained cheaply and generically from any PRG using the constructions of Bellare and Yee [23]; moreover a PRG can be easily obtained from a PRF. Thus our second scheme can be instantiated in a variety of ways.

In the full version of this paper [24], we provide a detailed comparison of instantiations of our new constructions with a variety of proven-secure KAS from the literature.

2 Hierarchical Key Assignment Schemes

2.1 Basic Definitions

A partially ordered set (poset) is a pair $(V, \leq)$ where V is a finite set of pairwise disjoint classes, called *security classes*, and '$\leq$' is a partial order on V, i.e. is a reflexive, antisymmetric, and transitive binary relation. A security class can represent a person, a department, or a user group in an organisation. Relation

$\leq$ is defined in accordance with authority for each class in V: for any two classes $u, v \in V$ we write $v \leq u$ or $u \geq v$ to indicate that users in class u can access the data of users in class v. We say that u *covers* v, denoted $v \lessdot u$ or $u \gtrdot v$, if $v < u$ and there does not exist $c \in V$ such that $v < c < u$. $(V, \leq)$ is a *totally ordered set* (or *chain*) if for all $u, v \in V$, either $v < u$ or $u > v$ or $u = v$. We say that $A \subseteq V$ is an *antichain* in V if for all $u, v \in A, u \neq v$, we have $v \nleq u$ and $v \ngeq u$. Any poset $(V, \leq)$ can be represented by a specific directed acyclic graph $G = (V, E)$, called *access graph*, where the vertices coincide with the security classes and there is an edge from class u to class v if and only if $u > v$. A *partition* of set V is a collection of sets $\{V_1, \ldots, V_s\}$ such that (i) $V_i \subseteq V \, \forall i$, (ii) $V_1 \cup \ldots \cup V_s = V$, and (iii) $i \neq j \Rightarrow V_i \cap V_j = \emptyset$.

The problem that we address consists of assigning keys (e.g., to be used in a symmetric encryption scheme) to each class in a poset in such a way that it should be possible to efficiently derive the keys for any descendant class in the poset. The cryptographic primitive that solves this challenge is called a hierarchical key assignment scheme [1], and is defined as follows.

Definition 1 (Key Assignment Scheme). *Let Γ denote a set of access graphs, i.e. of graphs that correspond to posets. A* hierarchical key assignment scheme (KAS) *for Γ is a pair of algorithms* (`Gen`, `Derive`) *satisfying the following conditions:*

1. `Gen`$(1^\rho, G)$ *is a probabilistic polynomial-time algorithm that takes as input a security parameter 1^ρ and a graph $G = (V, E) \in \Gamma$ and outputs (a) for all classes $u \in V$: private information S_u and key $k_u \in \{0,1\}^{p(\rho)}$, for a fixed polynomial p; (b) public information pub. We denote by (S, k, pub) the output of* `Gen`$(1^\rho, G)$*, where $S = (S_u)_{u \in V}$ and $k = (k_u)_{u \in V}$ are the vectors of private information and keys, respectively.*
2. `Derive`(G, u, v, S_u, pub) *is a deterministic polynomial-time algorithm that takes as input a graph G, classes $u, v \in V$ such that $v \leq u$, private information S_u, and public information pub, and outputs a key $k \in \{0,1\}^{p(\rho)}$ assigned to class v.*

Correctness requires that for all $\rho \in \mathbb{N}$, all $G \in \Gamma$, all (S, k, pub) output by `Gen`$(1^\rho, G)$, and all $u, v \in V, v \leq u$: `Derive`$(G, u, v, S_u, pub) = k_v$.

2.2 Security of Key Assignment Schemes

Various informal security models for key assignment schemes have been developed and proposed in the past. Formal security modelling began with [8]. However, as we will argue, all these models – both formal and informal – are inadequate for practical application in the most challenging of security environments. In the following, we first describe our new, strengthened models, and discuss the differences to the established ones.

We consider variants of the key indistinguishability (KI) security goal proposed by Atallah *et al.* in [8]. We consider models with both static and dynamic adversaries. It will shortly become clear, however, that these two models are polynomially equivalent. We begin with an informal statement of our security models, and then give a formal model in terms of a security experiment involving an adversary.

Static adversaries $\mathcal{A}_{stat}$, upon given an access graph $G = (V, E)$, first choose a security class $u \in V$ to attack. Using `Gen` algorithm on graph G, the experiment generates (S, k, pub). The adversary is then provided with private information S_v assigned to all classes $v \in V$ that should *not* enable the computation of key k_u, along with the set of all keys k_v associated to classes $v \in V$ such that $v > u$, and the public information pub. Precisely, the adversary gets pub and the two sets $Corrupt_{G,S,u}$ and $Keys_{G,u}$, where we define

$$Corrupt_{G,S,u} = \{S_v \in S \mid u \not\leq v\} \qquad \text{and} \qquad Keys_{G,u} = \{k_v \mid v > u\} \ .$$

Notice that, given $Corrupt_{G,S,u}$, the adversary can compute for himself all keys k_v for $v \in Corrupt_{G,S,u}$. As a challenge, the adversary additionally gets either key k_u or a random string of the same length, and it has to distinguish these two cases. We refer to Definition 2 below for the formal specification of this experiment. Observe that, from the obtained information, the adversary can gain access to k_v for any $v \in V \setminus \{u\}$.

In contrast to static adversaries, dynamic (also called adaptive) adversaries $\mathcal{A}_{dyn}$ may request keys k_v and secret information S_v in an adaptive manner before eventually committing to a security class $u \in V$ they want to attack. After receiving a challenge based on key k_u, they continue to request keys and secret information until terminating and outputting a bit. The adversary wins in the experiment if it successfully distinguishes the key k_u from random, under the restriction that $u \not\leq v$ for all classes v in the corrupted set and that key k_u has not been requested.

It is not difficult to see that the static and dynamic models are actually polynomially equivalent. Indeed, in the corresponding reduction, the static adversary simply guesses which class will be the subject of the dynamic adversary's query, and aborts if the guess turns out to be incorrect; this reduction succeeds with probability $1/|V|$. A similar proof was used in [9] (and implicitly in [8]). So schemes proven secure against static adversaries are automatically also secure against dynamic adversaries (albeit with a less tight overall security reduction). In the remainder of the paper, we focus on the static case.

We next give our definition for security in the sense of *strong key indistinguishability with respect to static adversaries (S-KI-ST-security)*, formalising the above discussion.

Definition 2 (S-KI-ST). *Let Γ be a set of access graphs and let* (`Gen`, `Derive`) *be a hierarchical key assignment scheme for Γ. Consider the following experiment (where we assume that adversary $\mathcal{A}$ keeps state between invocations):*

```
Experiment $\mathrm{Exp}_{\mathcal{A},G}^{\mathrm{S-KI-ST}}(1^\rho)$ :
    $u \leftarrow \mathcal{A}(1^\rho, G)$
    $(S, k, pub) \leftarrow \mathtt{Gen}(1^\rho, G)$
    $\beta \overset{r}{\leftarrow} \{0,1\}$
    $\textit{If } \beta = 1 \textit{ then } T \leftarrow k_u \textit{ else } T \overset{r}{\leftarrow} \{0,1\}^{p(\rho)}$
    $d \leftarrow \mathcal{A}(pub, \mathit{Corrupt}_{G,S,u}, \mathit{Keys}_{G,u}, T)$
    $\textit{return } d$
```

For any $G \in \Gamma$, the advantage of $\mathcal{A}$ in the above experiment is defined as

$$Adv_{\mathcal{A},G}^{\mathrm{S-KI-ST}}(\rho) = 2\left|\Pr\left[\mathrm{Exp}_{\mathcal{A},G}^{\mathrm{S-KI-ST}}(1^\rho) = \beta\right] - 1/2\right|.$$

Note that if we write $\mathrm{Exp}_{\mathcal{A},G}^{\mathrm{S-KI-ST},\gamma}(1^\rho)$, $\gamma \in \{0,1\}$, for the modification of $\mathrm{Exp}_{\mathcal{A},G}^{\mathrm{S-KI-ST}}(1^\rho)$ where bit β is fixed to $\beta = \gamma$, we have that

$$Adv_{\mathcal{A},G}^{\mathrm{S-KI-ST}}(\rho) = \left|\Pr[\mathrm{Exp}_{\mathcal{A},G}^{\mathrm{S-KI-ST},1}(1^\rho) = 1] - \Pr[\mathrm{Exp}_{\mathcal{A},G}^{\mathrm{S-KI-ST},0}(1^\rho) = 1]\right|.$$

The key assignment scheme is said to be secure in the sense of strong key indistinguishability with respect to static adversaries (S-KI-ST-secure) *if $Adv_{\mathcal{A},G}^{\mathrm{S-KI-ST}}(\rho)$ is negligible for every efficient adversary $\mathcal{A}$ and any graph $G \in \Gamma$.*

It will be evident that one can also define an S-KR-ST-security notion, in which the adversary is required to recover the key k_u rather than distinguish it from a random key. Clearly S-KI-ST-security implies S-KR-ST security.

We now explain why our model is stronger than the one introduced by Atallah *et al.* [8] that it is based on. While our S-KI-ST adversary receives both the set $\mathit{Corrupt}_{G,S,u} \subseteq S$ of secret information and the set $\mathit{Keys}_{G,u} \subseteq \{0,1\}^{p(\rho)}$ of computed (symmetric) keys, in the model from [8] the adversary receives only the former set when performing its attack. In the dynamic setting, our strong adversary has access to keys k_v for which $v > u$, where u is the challenge security class, whereas in the dynamic model of [8], the adversary has no access to such keys. Now in a real deployment of a scheme, some of the cryptographic keys k_v used in the scheme may leak, perhaps through cryptanalysis or misuse. In this case, we would like our selected security model to provide the strongest possible guarantees about the security of other keys that have not been leaked. But note that the previous security model from [8] provides no such guarantees, whereas our model provides the strongest possible guarantee, in that *all* keys k_v with $v > u$ are given to the adversary. Indeed, as the next example makes clear, it is quite feasible that leakage of a key k_v for which $v > u$ can damage the security of the key k_u.

A separating example: Consider a graph (V, E) having linear structure, i.e. $V = \{v_0, \ldots, v_{n-1}\}$ with $v_{i+1} \lessdot v_i$ for all i. Let H be a one-way function, which we model as a random oracle. We select S_{v_0} at random from the domain of H and set $k_{v_i} = S_{v_i}$ and $S_{v_{i+1}} = H(S_{v_i})$ for all i. It is clear how the $\texttt{Gen}$ and $\texttt{Derive}$ algorithms should be defined, and that the resulting scheme is correct. It is also easy to see that the scheme is KR-ST-secure in the random oracle model, in the sense of [8]. However, it is also clear that with knowledge of key S_{v_0}, all keys S_v in the hierarchy can be efficiently determined (including challenge key k_u) and hence the scheme is insecure in the S-KR-ST model. We note that this separation is for key recovery (KR) security notions.

3 Security Analysis of the Chain Partition Construction

We begin by reviewing the Chain Partition Construction for key assignment schemes from [15]. Given a partially ordered set $(V, \leq)$, represented by the directed acyclic graph $P = (V, E)$, Dilworth's Theorem [25] asserts that every partially ordered set $(V, \leq)$ can be partitioned into w chains, where w is the *width* of V, that is, the cardinality of the largest antichain in V. The partition need not be unique. We select a particular partition of V into chains $\{C_0, \ldots, C_{w-1}\}$. The length of C_i is denoted by l_i, for $0 \leq i \leq w-1$. We let $l_{\max}$ denote $\max_i\{l_i\}$. The maximum class of C_i is regarded as the first class in C_i and the minimum class as the last class. Since $\{C_0, \ldots, C_{w-1}\}$ is a partition of V, each $u \in V$ belongs to precisely one chain.

Let $C = u_0 \gtrdot \ldots \gtrdot u_m$ be any chain in V. Then any chain of the form $u_j \gtrdot \ldots \gtrdot u_m$, $0 < j \leq m$ is said to be a *suffix* of C. Now, for any $u \in V$, the set $\downarrow u := \{v \in V : v \leq u\}$ has non-empty intersection with one or more chains $C_0, \ldots, C_{w-1}$. It is proved in [15] that the intersection of $\downarrow u$ and the chain C_i is a suffix of C_i or the empty set. Following, [15], this will enable us to define the private information that should be given to a user with label u.

Since $\{C_0, \ldots, C_{w-1}\}$ is a partition of V into chains, $\{\downarrow u \cap C_0, \ldots, \downarrow u \cap C_{w-1}\}$ is a disjoint collection of chain suffixes. Additionally, the private information for each class in V should be chosen so that the private information for the j-th class of a chain can be used to compute keys for all lower classes in that chain. Hence, we can see that a user with label u should be given the private information for the maximal classes in the non-empty suffixes $\downarrow u \cap C_0, \ldots, \downarrow u \cap C_{w-1}$. Given $u \in V$, let $\hat{u}_0, \ldots, \hat{u}_{w-1}$ denote these maximal classes, with the convention that $\hat{u}_i = \perp$ if $\downarrow u \cap C_i = \emptyset$. Let u^i_j denote the j-th class in the chain C_i, where $0 \leq j \leq l_i - 1$.

The Chain Partition Construction: Let $(V, \leq)$ be a poset, $P = (V, E)$ the corresponding directed acyclic graph, and ρ a security parameter. Select a chain partition of V into w chains $C_0, \ldots, C_{w-1}$, so that C_i contains classes $u^i_0, u^i_1, \ldots, u^i_{l_i-1}$, with $u^i_{j+1} < u^i_j$, $0 \leq j < l_i - 1$. Let $l_{\max}$ denote $\max_i\{l_i\}$. Additionally, let $\texttt{X} = (\texttt{Gen}_\texttt{X}, \texttt{Derive}_\texttt{X})$ be a KAS scheme for the set consisting of a single chain of length exactly $l_{\max}$. Then the chain partition scheme $\texttt{KAS}_\texttt{CP}(\texttt{X}, P) = (\texttt{Gen}_\texttt{CP}, \texttt{Derive}_\texttt{CP})$ (relative to the particular partition selected) is defined as follows.

Algorithm $\mathtt{Gen}_{\mathtt{CP}}(1^\rho, P)$:

1. For $0 \leq i \leq w-1$, run $\mathtt{Gen}_{\mathtt{X}}$ on inputs 1^ρ and a chain of length $l_{\max}$ to obtain (T^i, k^i, pub_i). Discard the last $l_{\max} - l_i$ elements of T^i and k^i to obtain the secret information and keys for a chain of length l_i. Note that this chain has the same `Derive` algorithm as the starting chain. For ease of notation, we continue to denote the reduced sets by T^i and k^i, and we write $T^i = \{T_{u_0^i}, \dots, T_{u_{l_i-1}^i}\}$ and $k^i = \{k_{u_0^i}, \dots, k_{u_{l_i-1}^i}\}$. We stress here that we could run different algorithms $\mathtt{Gen}_{\mathtt{X}}$ to produce the different chains of lengths l_i, but for ease of notation we will assume they are all the same.
2. For each $u \in V$, define the private information S_u to be $\{T_{\hat{u}_i} : \hat{u}_i \neq \perp, 0 \leq i \leq w-1\}$ and the encryption key k_u to be $k_u = k_{u_j^i}$, where $u = u_j^i$.
3. Let S and k be the sets of private information and keys, respectively, in the above construction, and let $pub_{\mathtt{CP}} = (pub_0, \dots, pub_{w-1})$.
4. Output $(S, k, pub_{\mathtt{CP}})$.

Algorithm $\mathtt{Derive}_{\mathtt{CP}}(P, u_j^i, u_h^g, S_{u_j^i}, pub_{\mathtt{CP}})$:

1. For $u_j^i \geq u_h^g$, find $\hat{u}_g$, the maximal class in $\downarrow u_j^i \cap C_g$. This class is in chain C_g. We denote it by u_r^g, where $0 \leq r < l_g$. Note that, by construction, $u_r^g \leq u_j^i$ and $T_{u_r^g} \in S_{u_j^i}$.
2. Set $k_{u_h^g} \leftarrow \mathtt{Derive}_{\mathtt{X}}(C_g, u_r^g, u_h^g, T_{u_r^g}, pub_g)$.
3. Output $k_{u_h^g}$.

Theorem 1 (S-KI-ST Security of the Chain Partition Construction). *Let P be a directed acyclic graph and* X *be an S-KI-ST-secure scheme for single chains. Then scheme* $\mathtt{KAS}_{\mathtt{CP}}(\mathtt{X}, P) = (\mathtt{Gen}_{\mathtt{CP}}, \mathtt{Derive}_{\mathtt{CP}})$ *obtained from the chain partition construction is also S-KI-ST-secure.*

Proof. Assume $\mathcal{A}_{\mathtt{CP}}$ attacks a class u_j^i of graph P. If $\mathcal{A}_{\mathtt{CP}}$ is able to distinguish between the real key $k_{u_j^i}$ associated with class u_j^i, and a random string having the same length, we show that we can construct an S-KI-ST adversary $\mathcal{A}_{\mathtt{X}}$ against the scheme X that, using $\mathcal{A}_{\mathtt{CP}}$ as a black box, is able to distinguish between real or random keys. Algorithm $\mathcal{A}_{\mathtt{X}}$ plays the S-KI-ST security game described in Definition 2, receiving as initial input a security parameter 1^ρ and a chain on $l_{\max}$ classes. Adversary $\mathcal{A}_{\mathtt{X}}$ simulates the environment of $\mathcal{A}_{\mathtt{CP}}$ in such a way that $\mathcal{A}_{\mathtt{CP}}$'s view is indistinguishable from its view when playing the S-KI-ST security game.

Algorithm $\mathcal{A}_{\mathtt{X}}$:

1. Receive from the S-KI-ST experiment a chain C on $l_{\max}$ classes $v_0, \dots, v_{l_{\max}-1}$.
2. Let $P = (V, E) \in \Gamma$ and run $\mathcal{A}_{\mathtt{CP}}$ with input $(1^\rho, P)$ to get $\mathcal{A}_{\mathtt{CP}}$'s choice of target class u.
3. Generate a chain partition of P containing chains $C_0, \dots, C_{w-1}$. In this partition, class u is identified as some class u_j^i in some chain C_i of length $l_i \leq l_{\max}$. For $0 \leq t \leq w-1, t \neq i$, run `Gen` on inputs 1^ρ and a chain of length $l_{\max}$

to obtain (S^t, k^t, pub_t), the set of secret information, the set of keys and the public information for that chain. Note that, as in the construction, these sets can be truncated to obtain the set of secret information, the set of keys and the public information for a chain of length exactly l_t. By abuse of notation, we continue to use (S^t, k^t, pub_t) to denote this data.

4. Output v_j in chain C as $\mathcal{A}_{\mathtt{X}}$'s choice of target class. $\mathcal{A}_{\mathtt{X}}$ now receives as input the public information, pub, output by $\mathtt{Gen}_{\mathtt{X}}$, along with secret information S_{v_t} for all classes $v_t < v_j$ in C, and all secret keys k_{v_t} in C such that $v_t > v_j$. $\mathcal{A}_{\mathtt{X}}$ also receives as input a value T which is either the real key k_{v_j} or a random key of the same length. In what follows, $\mathcal{A}_{\mathtt{X}}$ will identify the first l_i classes in C with the chain C_i in the chain partition construction.
5. Set $pub_{\mathtt{CP}} = (pub_0, \ldots, pub_{i-1}, pub, pub_{i+1}, \ldots, pub_{w-1})$. Use the secret information S_{v_t} for classes $v_t < v_j$ in C together with the secret information in the sets S^t for $0 \leq t \leq w-1, t \neq i$ to build the set $Corrupt_{P,S,u}$. Use keys k_{v_t} in C such that $v_t > v_j$ and the keys from the sets k^t to build the set $Keys_{P,u}$.
6. Run $\mathcal{A}_{\mathtt{CP}}$ with inputs $(pub_{\mathtt{CP}}, Corrupt_{P,S,u}, Keys_{P,u}, T)$. It is easy to see that $\mathcal{A}_{\mathtt{X}}$ has the information required to properly construct the sets $Corrupt_{P,S,u}, Keys_{P,u}$ in such a way that $\mathcal{A}_{\mathtt{CP}}$'s input here is valid in $\mathcal{A}_{\mathtt{CP}}$'s experiment against the scheme $\mathtt{KAS}_{\mathtt{CP}}(\mathtt{X}, P)$, and such that T is the real key (resp. the random key) in $\mathcal{A}_{\mathtt{CP}}$'s experiment if and only if T is the real key (resp. the random key) in $\mathcal{A}_{\mathtt{X}}$'s experiment.
7. When $\mathcal{A}_{\mathtt{CP}}$ outputs a bit, output the same bit.

Now as $\mathcal{A}_{\mathtt{X}}$'s simulation is perfect, we see that the advantage of $\mathcal{A}_{\mathtt{X}}$ in winning its S-KI-ST indistinguishability game for the chain C of length $l_{\max}$ is the same as the advantage of $\mathcal{A}_{\mathtt{CP}}$ in playing the S-KI-ST indistinguishability game against $\mathtt{KAS}_{\mathtt{CP}}(\mathtt{X}, P)$. The theorem now follows. □

Note that, in the above theorem, $\mathtt{X}$ need only be an S-KI-ST-secure scheme for chains of length exactly $l_{\max}$. Because of the truncation trick, this is equivalent to $\mathtt{X}$ being an S-KI-ST-secure scheme for the set of graphs consisting of chains of lengths up to $l_{\max}$.

4 A Scheme Based on PRFs

We construct an S-KI-ST-secure key assignment scheme for totally-ordered hierarchical access structures of arbitrary depth, based on pseudorandom functions. By combining our construction with the result from Section 3, a general key assignment scheme for arbitrary posets is obtained.

We admit that also Atallah *et al.* [8] give an efficient PRF-based construction for arbitrary posets. However, their construction achieves only a security notion called 'key recovery' (where an adversary attacking a class u has to compute the challenge key k_u, instead of distinguishing it from random), which is weaker than our S-KI-ST notion. Moreover, our scheme is much simpler, and requires no public information to be stored.

We start by recalling the definition of a PRF, the central building block of our construction:

Definition 3 (Pseudorandom Function, PRF). *Let $\mathcal{K}, \mathcal{D}, \mathcal{R}$ be finite sets*[1] *and $F : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be an efficient function. For all $\kappa \in \mathcal{K}$ and $x \in \mathcal{D}$ we also write $F_\kappa(x) = F(\kappa, x)$ and call $F_\kappa : \mathcal{D} \to \mathcal{R}$ an* instance *of F. More formally, let $\mathtt{Rand} = \mathcal{R}^{\mathcal{D}} = \{g \,|\, g : \mathcal{D} \to \mathcal{R}\}$ denote the set of all functions $\mathcal{D} \to \mathcal{R}$. Let $\mathcal{A}_F$ be an algorithm that has oracle access to a function $\mathcal{D} \to \mathcal{R}$, and returns a bit. Consider the following two experiments:*

```
Experiment $\mathrm{Exp}^{\mathrm{PRF}-1}_{\mathcal{A}_F,F}(1^\rho)$ :
    $\kappa \xleftarrow{r} \mathcal{K}$
    $d \leftarrow \mathcal{A}_F^{F_\kappa}(1^\rho)$
    return $d$

Experiment $\mathrm{Exp}^{\mathrm{PRF}-0}_{\mathcal{A}_F,F}(1^\rho)$ :
    $g \xleftarrow{r} \mathtt{Rand}$
    $d \leftarrow \mathcal{A}_F^{g}(1^\rho)$
    return $d$
```

The advantage of $\mathcal{A}_F$ is defined as

$$Adv^{\mathrm{PRF}}_{\mathcal{A}_F,F}(\rho) = \left| \Pr\left[\mathrm{Exp}^{\mathrm{PRF}-1}_{\mathcal{A}_F,F}(1^\rho) = 1\right] - \Pr\left[\mathrm{Exp}^{\mathrm{PRF}-0}_{\mathcal{A}_F,F}(1^\rho) = 1\right] \right| .$$

We say that F is pseudorandom *(or: is a PRF) if $Adv^{\mathrm{PRF}}_{\mathcal{A}_F,F}(\rho)$ is negligible for every efficient adversary $\mathcal{A}_F$.*

In our following construction, we will use special PRFs where $\mathcal{K} = \mathcal{R} = \{0,1\}^\rho$ for security parameter ρ, and $\mathcal{D}$ is any set. We remark that some constructions in [8] also require PRFs with similar restrictions on keyspace and range. For concreteness, we propose to deploy the (hash-based) HMAC primitive [19] as a PRF (see also analysis in [26]). In addition, it might be possible to find suitable constructions based on number-theoretic assumptions, e.g. derived from factoring-based PRF by Naor and Reingold [20], or the PRF obtained by converting the BBS [21] PRG into a PRF via the Goldreich-Goldwasser-Micali construction [27].

4.1 A PRF-Based Key Assignment Scheme for Totally Ordered Hierarchies

We briefly recall the setting of key assignment for chains. Let Γ be the family of graphs corresponding to totally ordered hierarchies, and let $G = (V, E) \in \Gamma$ be a graph, where $V = \{u_0, \ldots, u_{n-1}\}$ for some n, and $u_{i+1} \lessdot u_i$ for all i. To each security class $u_i \in V$, private information S_i and key k_i are assigned, where S_i can be used to compute subordinated keys. Here we abuse notation, for better exposition (we can do this because we are in the linear setting), writing S_i for S_{u_i} and k_i for k_{u_i}.

[1] More precisely, we assume that $\mathcal{K}, \mathcal{D}, \mathcal{R}$ are families of finite sets, indexed by a security parameter ρ. That is, we require $\mathcal{K} = (\mathcal{K}_\rho)_{\rho \in \mathbb{N}}$, and similarly for $\mathcal{D}$ and $\mathcal{R}$. For the sake of a cleaner exposition, however, we do not write down the security parameter explicitly.

Let ρ be a security parameter and let $F : \{0,1\}^\rho \times \mathcal{D} \to \{0,1\}^\rho$ be a PRF. Let c_0 and c_1 be two different elements in $\mathcal{D}$. The `Gen` and `Derive` algorithms work as follows.

Algorithm `Gen`$(1^\rho, G)$:

```
1. Pick random $S_0 \overset{r}{\leftarrow} \{0,1\}^\rho$ and set $k_0 \leftarrow F_{S_0}(c_1)$.
2. For each class $u_i \in V, i > 0$, set $S_i \leftarrow F_{S_{i-1}}(c_0)$ and $k_i \leftarrow F_{S_i}(c_1)$.
3. Set $S \leftarrow (S_0, \ldots, S_{n-1})$, $k \leftarrow (k_0, \ldots, k_{n-1})$, and $pub \leftarrow \emptyset$.
4. Output $(S, k, pub)$.
```

Algorithm `Derive`(G, u_i, u_j, S_i, pub): (note that we may assume $j \geq i$)

```
1. If $i = j$ then return $k_j = F_{S_i}(c_1)$.
2. For $h = i + 1$ to $j$: $S_h \leftarrow F_{S_{h-1}}(c_0)$.
3. Return $k_j = F_{S_j}(c_1)$.
```

Observe that computing key k_j from secret information S_i requires exactly $j - i + 1$ evaluations of the underlying PRF.

The following theorem is proven in the full version of this paper [24].

Theorem 2 (S-KI-ST Security of the PRF-based Scheme for Totally Ordered Hierarchies). *The above PRF-based scheme is key indistinguishable, in the sense of Definition 2, for any totally ordered graph G, assuming security of pseudorandom function F.*

5 A Scheme Based on Forward-Secure PRGs

FS-PRGs, introduced by Bellare and Yee in [23], are stateful/iterated pseudorandom generators (PRGs) that deterministically derive sequences of fixed-length bit strings from an initial (random) seed. More precisely, in each iteration they output a string of bits, update their internal state, and securely erase the old state. Like in regular PRGs, the output sequences are required to be indistinguishable from sequences of random strings. The pivotal property of FS-PRGs is *forward security*, i.e. the adversary has the ability to eventually corrupt generator's internal state, but indistinguishability of output strings is guaranteed to still hold up to that point.

In this section, building on generic FS-PRGs, we construct a key assignment scheme which achieves S-KI-ST security for totally-ordered access graphs and, in combination with the results from Section 3, for arbitrary posets. It is worth pointing out that we actually widely generalize the construction from [16], which implicitly exploits the property of forward security of the BBS pseudorandom generator. As our construction generically builds on FS-PRGs, it is amenable to the efficiency gain obtained by replacing the BBS-based FS-PRG by, for instance, an HMAC-based one.

Before describing our scheme, let us first recall the definition and security notion of forward-secure pseudorandom number generators (FS-PRGs). Observe that we slightly weaken the model from [23] (considering static adversaries instead of adaptive ones), what renders our construction of a key assignment scheme more general. Clearly the FS-PRG constructions proposed and proved secure in [23] naturally remain secure in our adapted model.

Definition 4 (Forward-Secure PRG). *Let* $G_{\texttt{FS}} = (G_{\texttt{FS}}.\texttt{setup}, G_{\texttt{FS}}.\texttt{key}, G_{\texttt{FS}}.\texttt{next})$ *be a set of efficient algorithms such that* $G_{\texttt{FS}}.\texttt{setup}$ *is a probabilistic algorithm that, on input a security parameter* 1^ρ*, outputs a set of system parameters 'params';* $G_{\texttt{FS}}.\texttt{key}$ *is a probabilistic key generation algorithm that takes 'params' as input and outputs an initial state* $St_0 \in \{0,1\}^\rho$ *(the initial seed);* $G_{\texttt{FS}}.\texttt{next} : \{0,1\}^\rho \to \{0,1\}^\rho \times \{0,1\}^{p(\rho)}$ *is a deterministic algorithm that turns state* $St_{i-1} \in \{0,1\}^\rho$ *(the 'seed' at iteration* i*) into a pair* (St_i, Out_i)*, where* $St_i \in \{0,1\}^\rho$ *is the updated state, and* Out_i *is a* $p(\rho)$*-bit string.*

Let $\mathcal{D}$ *be an adversary against* $G_{\texttt{FS}}$*.* $\mathcal{D}$ *is fed with a number of output blocks,* $Out_1, Out_2, \ldots, Out_i$*, each of length* $p(\rho)$*, and is given the then current state of the generator,* St_i*. Consider the following experiments:*

```
Experiment $\mathrm{Exp}^{\mathrm{FS-PRG-1}}_{\mathcal{D},G_{\texttt{FS}}}(1^\rho)$ :
  $i \leftarrow \mathcal{D}$
  $params \overset{r}{\leftarrow} G_{\texttt{FS}}.\texttt{setup}(1^\rho)$
  $St_0 \overset{r}{\leftarrow} G_{\texttt{FS}}.\texttt{key}(params)$
  $i' \leftarrow 0$
  Repeat
    $i' \leftarrow i' + 1$
    $(St_{i'}, Out_{i'}) \leftarrow G_{\texttt{FS}}.\texttt{next}(St_{i'-1})$

  Until $i' = i$
  $Out \leftarrow Out_1, Out_2, \ldots, Out_i$
  $d \leftarrow \mathcal{D}(St_i, Out)$
  return $d$
```

```
Experiment $\mathrm{Exp}^{\mathrm{FS-PRG-0}}_{\mathcal{D},G_{\texttt{FS}}}(1^\rho)$ :
  $i \leftarrow \mathcal{D}$
  $params \overset{r}{\leftarrow} G_{\texttt{FS}}.\texttt{setup}(1^\rho)$
  $St_0 \overset{r}{\leftarrow} G_{\texttt{FS}}.\texttt{key}(params)$
  $i' \leftarrow 0$
  Repeat
    $i' \leftarrow i' + 1$
    $(St_{i'}, Out_{i'}) \leftarrow G_{\texttt{FS}}.\texttt{next}(St_{i'-1})$
    $Out_{i'} \overset{r}{\leftarrow} \{0,1\}^{p(\rho)}$
  Until $i' = i$
  $Out \leftarrow Out_1, Out_2, \ldots, Out_i$
  $d \leftarrow \mathcal{D}(St_i, Out)$
  return $d$
```

The advantage of $\mathcal{D}$ *is defined as*

$$Adv^{\mathrm{FS-PRG}}_{\mathcal{D},G_{\texttt{FS}}}(\rho) = \left| \Pr[\mathrm{Exp}^{\mathrm{FS-PRG-1}}_{\mathcal{D},G_{\texttt{FS}}}(1^\rho) = 1] - \Pr[\mathrm{Exp}^{\mathrm{FS-PRG-0}}_{\mathcal{D},G_{\texttt{FS}}}(1^\rho) = 1] \right| .$$

We say that $G_{\texttt{FS}}$ *is a forward-secure pseudorandom number generator (FS-PRG) if* $Adv^{\mathrm{FS-PRG}}_{\mathcal{D},G_{\texttt{FS}}}(\rho)$ *is negligible for every efficient adversary* $\mathcal{D}$*.*

5.1 The FS-PRG-Based Scheme for a Single Chain

Key assignment schemes for totally-ordered access graphs are readily constructed from FS-PRGs: In our construction, we identify the FS-PRG's state St_i with the private information S_i stored for class u_i, while key k_i is set to the FS-PRG's output Out_{i+1}.

More precisely, let Γ be the family of graphs corresponding to totally ordered hierarchies, let $G = (V, E) \in \Gamma$ be a graph, where $V = \{u_0, \ldots, u_{n-1}\}$ for some n, and $u_{i+1} \lessdot u_i$ for all i. As in Section 4.1, we write S_i for private information S_{u_i}, and k_i for key k_{u_i}. Let ρ be a security parameter, and let

$G_{\texttt{FS}} = (G_{\texttt{FS}}.\texttt{setup}, G_{\texttt{FS}}.\texttt{key}, G_{\texttt{FS}}.\texttt{next})$ be an FS-PRG. Then `Gen` and `Derive` algorithms work as follows.

```
Algorithm Gen($1^\rho, G$):
 1. Run $params \leftarrow G_{\texttt{FS}}.\texttt{setup}(1^\rho)$ and $S_0 \leftarrow G_{\texttt{FS}}.\texttt{key}(params)$;
 2. For all $0 \le i < n$: Compute $(S_{i+1}, k_i) \leftarrow G_{\texttt{FS}}.\texttt{next}(S_i)$;
 3. Set $S \leftarrow (S_0, \ldots, S_{n-1})$, $k \leftarrow (k_0, \ldots, k_{n-1})$, and $pub \leftarrow \emptyset$;
 4. Output $(S, k, pub)$.
```

```
Algorithm Derive($G, u_i, u_j, S_i, pub$):   (note that we may assume $j \ge i$)
 1. For $h = i$ to $j$: $(S_{h+1}, k_h) \leftarrow G_{\texttt{FS}}.\texttt{next}(S_h)$;
 2. Return $k_j$.
```

The following theorem is proven in the full version of this paper [24].

Theorem 3 (S-KI-ST Security of the FS-PRG-based Scheme for Totally Ordered Hierarchies). *The above FS-PRG-based scheme is key indistinguishable, in the sense of Definition 2, for any totally ordered graph G, assuming security of the FS-PRG, $G_{\texttt{FS}}$.*

References

1. Akl, S.G., Taylor, P.D.: Cryptographic solution to a problem of access control in a hierarchy. ACM Transactions on Computer Systems 1(3), 239–248 (1983)
2. MacKinnon, S.J., Taylor, P.D., Meijer, H., Akl, S.G.: An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. IEEE Transactions on Computers 34(9), 797–802 (1985)
3. Harn, L., Lin, H.Y.: A cryptographic key generation scheme for multilevel data security. Computers & Security 9(6), 539–546 (1990)
4. Chen, T.S., Chung, Y.F.: Hierarchical access control based on Chinese remainder theorem and symmetric algorithm. Computers & Security 21(6), 565–570 (2002)
5. Shen, V., Chen, T.S.: A novel key management scheme based on discrete logarithms and polynomial interpolations. Computers & Security 21(2), 164–171 (2002)
6. Wu, T.C., Chang, C.C.: Cryptographic key assignment scheme for hierarchical access control. Int. Journal of Computer Systems Science and Engineering 16(1), 25–28 (2001)
7. Yeh, J.H., Chow, R., Newman, R.: A key assignment for enforcing access control policy exceptions. In: Int. Symposium on Internet Technology, pp. 54–59 (1998)
8. Atallah, M.J., Blanton, M., Fazio, N., Frikken, K.B.: Dynamic and efficient key management for access hierarchies. In: ACM Conference on Computer and Communications Security, pp. 190–202 (2006)
9. Ateniese, G., De Santis, A., Ferrara, A.L., Masucci, B.: Provably-secure time-bound hierarchical key assignment schemes. In: ACM Conference on Computer and Communications Security, pp. 288–297 (2006)
10. Tzeng, W.G.: A secure system for data access based on anonymous authentication and time-dependent hierarchical keys. In: ACM Symposium on Information, Computer and Communications Security, pp. 223–230 (2006)

11. Wang, S.Y., Laih, C.S.: An efficient solution for a time-bound hierarchical key assignment scheme. IEEE Transactions on Dependable and Secure Computing 3(1), 91–100 (2006)
12. De Santis, A., Ferrara, A.L., Masucci, B.: Efficient Provably-Secure Hierarchical Key Assignment Schemes. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 371–382. Springer, Heidelberg (2007)
13. Atallah, M.J., Blanton, M., Fazio, N., Frikken, K.B.: Dynamic and efficient key management for access hierarchies. ACM Trans. Inf. Syst. Secur. 12(3) (2009)
14. D'Arco, P., De Santis, A., Ferrara, A.L., Masucci, B.: Variations on a theme by Akl and Taylor: Security and tradeoffs. Theoretical Computer Science 411(1), 213–227 (2010)
15. Crampton, J., Daud, R., Martin, K.M.: Constructing Key Assignment Schemes from Chain Partitions. In: Foresti, S., Jajodia, S. (eds.) Data and Applications Security and Privacy XXIV. LNCS, vol. 6166, pp. 130–145. Springer, Heidelberg (2010)
16. Freire, E.S.V., Paterson, K.G.: Provably Secure Key Assignment Schemes from Factoring. In: Parampalli, U., Hawkes, P. (eds.) ACISP 2011. LNCS, vol. 6812, pp. 292–309. Springer, Heidelberg (2011)
17. Ateniese, G., De Santis, A., Ferrara, A.L., Masucci, B.: Provably-secure time-bound hierarchical key assignment schemes. J. Cryptology 25(2), 243–270 (2012)
18. Crampton, J., Martin, K.M., Wild, P.R.: On key assignment for hierarchical access control. In: Computer Security Foundations Workshop, pp. 98–111 (2006)
19. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
20. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. J. ACM 51(2), 231–262 (2004)
21. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. SIAM Journal on Computing 15(2), 364–383 (1986)
22. Blum, M., Goldwasser, S.: An Efficient Probabilistic Public-Key Encryption Scheme Which Hides All Partial Information. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 289–299. Springer, Heidelberg (1985)
23. Bellare, M., Yee, B.S.: Forward-Security in Private-Key Cryptography. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 1–18. Springer, Heidelberg (2003)
24. Freire, E.S.V., Paterson, K.G., Poettering, B.: Simple, efficient and strongly KI-secure hierarchical key assignment schemes (2012), http://eprint.iacr.org/2012/645
25. Dilworth, R.P.: A decomposition theorem for partially ordered sets. Annals of Mathematics 51(1), 161–166 (1950)
26. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 494–510. Springer, Heidelberg (2004)
27. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986)

Randomized Partial Checking Revisited

Shahram Khazaei[⋆] and Douglas Wikström

KTH Royal Institute of Technology
khazaei@kth.se,
dog@csc.kth.se

Abstract. We study mix-nets with randomized partial checking (RPC) as proposed by Jakobsson, Juels, and Rivest (2002). RPC is a technique to verify the correctness of an execution both for Chaumian and homomorphic mix-nets. The idea is to relax the correctness and privacy requirements to achieve a more efficient mix-net.

We identify serious issues in the original description of mix-nets with RPC and show how to exploit these to break both correctness and privacy, both for Chaumian and homomorphic mix-nets. Our attacks are practical and applicable to real world mix-net implementations, e.g., the Civitas and the Scantegrity voting systems.

1 Introduction

A *mix-net* is a protocol that provides anonymity for a group of senders. This notion was first introduced by Chaum in 1981 [2] to implement anonymous channels in general and electronic voting schemes in particular. A voter submits an encrypted ballot and the mix-net later outputs the plaintexts in random order. Other applications of mix-nets include anonymous web browsing [8], private payment systems [16], and multiparty computation [13].

The original mix-net proposed by Chaum is a decryption mix-net that works as follows with mix-servers $\mathcal{M}_1, \ldots, \mathcal{M}_k$. The jth mix-server generates a key pair (pk_j, sk_j) and publishes the public key. To encrypt a message m_i, the ith sender forms a ciphertext

$$c_{i,0} = \mathsf{Enc}_{pk_1}(\mathsf{Enc}_{pk_2}(\cdots \mathsf{Enc}_{pk_k}(m_i) \cdots)) \enspace .$$

The mix-servers then take turns and "peel off" a layer of encryption and permute the resulting ciphertexts before publishing them. More precisely, the jth mix-server computes

$$c_{i,j} = \mathsf{Dec}_{sk_j}(c_{\pi_j(i),j-1})$$

for a random permutation π_j. Note that the output of the last mix-server is the list of randomly permuted plaintexts. Chaum's mix-net preserves the privacy of the senders as long as at least one server keeps its secret key and its random permutation secret, but a single mix-server can replace all ciphertexts with ciphertexts of his own choosing.

[⋆] The author is now at the Department of Mathematical Sciences at Sharif University of Technology, Tehran, Iran: shahram.khazaei@sharif.edu

Jakobsson, Juels, and Rivest [15] proposed *Randomized Partial Checking* (RPC) as a technique to address this problem and gave heuristic arguments showing that it should be difficult to change more than a small number of ciphertexts. The idea is strikingly simple: each mix-server is challenged to reveal how he processed a random subset of his input ciphertexts. If more than a handful of the ciphertexts are processed incorrectly, then he should get caught. In this scheme a single mix-server clearly does not provide privacy, but it seems that several mix-servers taken together still provide fairly strong privacy guarantees.

For homomorphic cryptosystems, Park, Itoh and Kurosawa introduced re-encryption mix-nets [20]. Here the mix-servers generate a single joint public key with a verifiably secret shared secret key and decryption is replaced by re-encryption, followed by a joint verifiable decryption step. This means that the size of the ciphertext is independent of the number of mix-servers. Sako and Kilian constructed the first universally verifiable mix-net [23], where senders can verify that the entire shuffle was performed correctly (and not just that their own input was included in the output). Sako and Kilian's construction was based on cut-and-choose zero-knowledge proofs; Neff [18] and Furukawa and Sako [7] gave much more efficient zero-knowledge proofs of shuffles.

Much of the work in the field aim to improve the efficiency of the mix-net, e.g., [12,11,9,14], but most mix-nets not based on proofs of shuffles have been broken or vulnerabilities have been found.

1.1 Motivation and Contribution

Random partial checking (RPC) is fast and in contrast to efficient proofs of shuffles [18,7] it is compatible with *any* cryptosystem for which a mix-server can efficiently prove that it processed a single ciphertext correctly. Thus, it is one of few mix-nets that can be used with cryptosystems conjectured to be secure against quantum computers. Currently, it is also the only viable option to construct a *universally verifiable* mix-net from any cryptosystem. Furthermore, it is perhaps the most common heuristically secure mix-net found in real implementations. Jakobsson et al. only claim a restricted form of security for their mix-net, but as such it has resisted all attacks for 10 years. This makes it an important cryptographic construction to study.

We show that the description of RPC by Jakobsson et al. [15] does not capture their ideas correctly. More precisely, we have discovered fully practical attacks on both the privacy and the correctness of their protocol and notable real world implementations of it, e.g., the Civitas voting system [4]. Using similar ideas we can also attack the correctness of related schemes, e.g., the Scantegrity voting system [3] that was used in real elections including Takoma Park City Municipal Elections 2009 and 2011. The most serious attack allows an adversary to replace the complete output of the mix-net without detection by corrupting a single mix-server, but we can also break the anonymity of targeted senders at low cost in terms of the number of corrupted senders. Furthermore, we show that even if the issues we have identified are handled correctly, an adversary can replace t ciphertexts in the homomorphic mixing without detection with probability roughly $(3/4)^t$ and not 2^{-t} as claimed. Finally, we argue informally that there is no blackbox proof of security for homomorphic mix-nets with RPC.

The proper interpretation of our results is not that the basic ideas behind RPC are flawed, but we think RPC should not be used at all with homomorphic mix-nets, and it should not be used for Chaumian mix-nets until there is a rigorous proof of security. We believe that modeling and proving the security is possible if only the issues identified in this paper are rectified and hope to provide such a proof in future work.

2 Notation

We consider a mix-net with k pairs of mix-servers $\mathcal{M}_1, \ldots, \mathcal{M}_{2k}$ that provide anonymity for a group of N senders $\mathcal{P}_1, \ldots, \mathcal{P}_N$. It is convenient to think of each pair of consecutive mix-servers as if they are executed by a single entity, i.e., $\mathcal{M}_{2j-1}, \mathcal{M}_{2j}$ is executed by the jth party.

We denote a cryptosystem by $\mathcal{CS} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where Gen, Enc, and Dec denote the key generation algorithm, the encryption algorithm, and the decryption algorithm respectively. The key generation algorithm Gen outputs a pair (pk, sk) consisting of a public key and a private key. We let $\mathcal{M}_{pk}$, $\mathcal{C}_{pk}$, and $\mathcal{R}_{pk}$ be the sets of plaintexts, ciphertexts, and randomizers, respectively, associated with the public key pk. We write $c = \mathsf{Enc}_{pk}(m, r)$ for the encryption of a plaintext m using randomness r, and $\mathsf{Dec}_{sk}(c) = m$ for the decryption of a ciphertext c. We often view Enc as a probabilistic algorithm and drop r from our notation. Recall that a cryptosystem is called *homomorphic* if for every public key pk: $\mathcal{M}_{pk}$, $\mathcal{C}_{pk}$, and $\mathcal{R}_{pk}$ are groups and for every $m_0, m_1 \in \mathcal{M}_{pk}$ and $r_0, r_1 \in \mathcal{R}_{pk}$ we have

$$\mathsf{Enc}_{pk}(m_0, r_0)\mathsf{Enc}_{pk}(m_1, r_1) = \mathsf{Enc}_{pk}(m_0 m_1, r_0 + r_1) \enspace .$$

Homomorphic cryptosystems allow ciphertexts to be *re-encrypted*. This means that anybody with access to the public key can take a ciphertext c and form $c \cdot \mathsf{Enc}_{pk}(1, r)$, for a randomly chosen $r \in \mathcal{R}_{pk}$, and the resulting ciphertext is identically, but independently, distributed to the original ciphertext.

We extend our notation to lists of keys. For a plaintext m and lists of public and private keys $pk = (pk_1, \ldots, pk_\ell)$ and $sk = (sk_1, \ldots, sk_\ell)$, we write

$$\begin{aligned} \mathsf{Enc}_{pk}(m) &= \mathsf{Enc}_{pk_1}(\mathsf{Enc}_{pk_2}(\cdots \mathsf{Enc}_{pk_\ell}(m) \cdots)) \text{ and} \\ \mathsf{Dec}_{sk}(c) &= \mathsf{Dec}_{sk_\ell}(\mathsf{Dec}_{sk_{\ell-1}}(\cdots \mathsf{Dec}_{sk_1}(c) \cdots)) \enspace . \end{aligned}$$

3 Randomized Partial Checking

In this section we provide a brief description of the mix-net with randomized partial checking (RPC) proposed by Jakobsson et al. [15] that focuses on the most common variations. We mostly borrow their notation in the following for easy reference.

The mix-net with RPC is not intended to provide full correctness or privacy; it gains in efficiency by relaxing the security requirements. The goal is to prevent mix-servers from undetectably modifying *many* inputs; it is easy to see that a malicious server can succeed in changing a small number of inputs with constant probability. Assuming that the penalty for being identified as a cheater is severe, the authors of [15] argue that this suffices. The privacy guarantees are also relaxed: while the exact correspondence between senders and their inputs is hidden, some information may still be leaked (e.g., that a specific input did not originate from a specific sender with a probability different from what is ideally expected).

RPC is proposed as a general technique to verify the correctness of both homomorphic and Chaumian mix-nets. A key requirement on the underlying cryptosystem is that it allows a party to transform a ciphertext c into a ciphertext c' using a cryptographic transformation X_j, and then prove that it did so correctly. For a homomorphic mix-net, X_j denotes random re-encryption and proving that this was done correctly amounts to revealing the randomness used. For a Chaumian mix-net X_j denotes decryption. To prove that the transformation was applied a zero-knowledge proof of correct decryption can be used. However, if the cryptosystem allows using the secret key to recover the randomness used to form a ciphertext from the ciphertext itself, then the randomness can simply be revealed. This interesting feature is not mentioned in [15]. In the following we assume for concreteness that the cryptosystem has this special feature.

3.1 Key Distribution

There is a public key pk and a corresponding secret key sk for the mix-net. For the homomorphic mix-net, the public key is jointly generated, whereas for the Chaumian mix-net it is a list of $2k$ keys. For homomorphic mixing, the joint secret key is typically verifiably secret shared among the mix-servers [6] and not known by any subset smaller than a certain threshold. For the Chaumian mix-net, $\mathcal{M}_j$ knows the jth component of the secret key, sk_j, and each such key is verifiably secret shared among all the mix-servers. The threshold used determines the privacy and robustness of the mix-net.

3.2 Ballot Preparation and Encryption

Each sender $\mathcal{P}_i$ encrypts his plaintext m_i as $c_{i,0} = \mathsf{Enc}_{pk}(m_i)$, and sends it to the bulletin board. Pfitzmann [22,21] pointed out that this ciphertext must either be non-malleable (CCA2-secure) on its own, or augmented with a non-interactive zero-knowledge proof of knowledge of the plaintext to provide this property. For concreteness we assume that the latter approach is used.

3.3 Initial Ballot Checking

When all voters have submitted their ciphertexts, all duplicates are removed (preserving a single copy) and ciphertexts with invalid proofs are discarded. Without loss of generality, we assume that this results in a list of N ciphertexts $(c_{1,0}, \ldots, c_{N,0})$.

3.4 Permutation Commitment

Each server $\mathcal{M}_j$ selects a permutation π_j on N elements uniformly at random. The server publishes on the bulletin board a commitment to π_j or π_j^{-1} depending on j being odd or even. The commitment consists of N integer commitments of the form

$$\Gamma_j^{(In)} = \left(\zeta_{w_{i,j}}[\pi_j(i)]\right)_{i=1}^{N} \text{ or } \Gamma_j^{(Out)} = \left(\zeta_{w_{i,j}}[\pi_j^{-1}(i)]\right)_{i=1}^{N} ,$$

depending on the parity of j, where $\zeta_w[i]$ denotes a commitment to integer i under randomness w. We simply let $\gamma_{i,j}$ denote the ith commitment of mix-server $\mathcal{M}_j$, i.e., $\gamma_{i,j}$ is an element of $\Gamma_j^{(In)}$ or $\Gamma_j^{(Out)}$ depending on the parity of j.

3.5 Mix-Net Processing

Each server $\mathcal{M}_j$, in turn accepts a ciphertext list $(c_{1,j-1}, \ldots, c_{N,j-1})$ from the bulletin board as input, and computes a list $(c_{1,j}, \ldots, c_{N,j})$ as output and publishes it on the bulletin board, where input ciphertext goes through the cryptographic transformation

$$c_{i,j} = X_j(c_{\pi_j(i),j-1}) \ .$$

For the homomorphic mixing, the transformation X_j re-encrypts the input ciphertext using the joint public key. For the Chaumian mix-net, X_j is the decryption algorithm executed with the secret key sk_j.

3.6 Correctness Check

Each mix-server $\mathcal{M}_j$ is verified as follows. The mix-servers jointly select a collection of ciphertexts from its input or output list, depending on j being even or odd. The selection method is explained in the next section. Mix-server $\mathcal{M}_j$ is then asked to reveal a collection of input/output correspondences related to the selected ciphertexts. Suppose that $\mathcal{M}_j$ wishes to reveal information that allows anyone to verify that an input ciphertext $c_{k,j-1}$ maps to $c_{i,j}$. The mix-server $\mathcal{M}_j$ reveals the triple $(k, i, r_{k,i,j})$ where $r_{k,i,j}$ is the information required to validate the transformation $c_{i,j} = X_j(c_{k,j-1})$. In the case of the Chaumian mix-net, $r_{k,i,j}$ is the randomness chosen by a sender when encrypting $c_{i,j}$ to obtain $c_{k,j-1}$; in the case of the homomorphic mix-net, $r_{k,i,j}$ is the randomness value chosen by $\mathcal{M}_j$ itself for re-encrypting $c_{k,j-1}$.

Additionally, $\mathcal{M}_j$ reveals his commitment to the mapping from $c_{k,j-1}$ to $c_{i,j}$. An odd-numbered mix-server $\mathcal{M}_j$, for the selected ciphertext $c_{i,j}$, decommits $\gamma_{i,j}$ (revealing $k = \pi_j(i)$) whereas an even-numbered mix-server, for the selected ciphertext $c_{k,j-1}$, decommits $\gamma_{k,j}$ (revealing $i = \pi_j^{-1}(k)$).

If all the input/output correspondences verifies correctly, then the mix-server passes the correctness check. Otherwise, he is identified as a cheater.

Inconsistent Permutation Commitments are Possible. We observe that Jakobsson et al. do not stress the importance of checking that the opened commitments of integers are *consistent with a permutation*. That is, all the revealed commitments $\gamma_{i,j}$'s (or $\gamma_{k,j}$'s) must open to *distinct* values. In fact, they emphasize that the verification of correspondences can be performed independently, i.e., it is easy to parallelize. In Section 6, we explore this issue in depth.

3.7 Selection Strategy

Jakobsson et al. propose different schemes for how to select a subset for each mix-server's inputs. The *pairwise dependent selection* scheme is favored and can be described as follows. Two adjacent mix-servers are paired to ensure that no overlapping correspondences are revealed. The output list $(c_{1,2j-1},\ldots,c_{N,2j-1})$ of an odd-numbered mix-server is divided in two groups of ciphertexts. Then $\mathcal{M}_{2j-1}$ is challenged with one group and $\mathcal{M}_{2j}$ with the other one.

The partitioning of the output list of an odd-numbered mix-servers is done as follows. Mix-servers jointly compute a random seed R. One way to do this is that each mix-server commits to a random value R_j before starting verification. Then, they open their commitments and compute R as the XOR of all R_j's. The seed R can be used to determine which challenges each mix-server needs to answer. For achieving universal verifiability, RPC combines the random seed R with the contents of the bulletin board, denoted by $\mathcal{BB}$, to compute a seed Q_{2j-1}, e.g., by computing the hash value $H(H(R,\mathcal{BB}),j)$ where H is a cryptographic hash function. The seed Q_{2j-1} is interpreted as a vector of N boolean values of which half are true which is used to divide $(c_{1,2j-1},\ldots,c_{N,2j-1})$ in two disjoint groups.

The correctness check of mix-servers can be done in two ways: *in-phase with mixing*, i.e., right after each mix-server pair has published his output list, or *after all mixing has been performed*, i.e., the last mix-server has published his output list. Both schemes are proposed for homomorphic mixing, but if cheating is detected in the second scheme, then the culprit would be kicked out and the mixing restarts. Only when the mixing proceeds without any detected cheating does joint decryption take place. For Chaumian mixing the correctness check is suggested to take place in-phase. If a mix-server is identified as a cheater, then his secret key is recovered and his input decrypted in the open.

3.8 Ballot Decryption

For the homomorphic mix-net, once the mixing operation is complete without detecting any cheating mix-server, the holders of the secret key sk (the mix-servers or some other entities) jointly decrypt all output ciphertexts, yielding the full list of plaintext ballots. This decryption operation is not needed in the case of a decryption mix-net, since the X_j transformations have already performed all necessary decryptions.

4 Pfitzmann's Attack and a Generalization

It is easy to see that for the homomorphic mix-net with RPC, the attack of Pfitzmann [22,21] can be adopted to break the privacy of any given sender with probability $1/2$. This forms the basis of our attacks on privacy, so it is worthwhile to describe it in detail.

The first mix-server knows the correspondence between voters and ciphertexts. He targets a sender with a submitted ciphertext c. Then he chooses an integer δ of suitable size randomly and replaces one of his outputs by c^δ. With probability $1/2$ this is not detected during RPC. Then he waits until the mix-net produces an output, identifies two plaintexts m and m^* that satisfy $m^* = m^\delta$, and concludes that m was submitted by the targeted sender.

With more processing, Pfitzmann's basic attack can be generalized to break the privacy of s senders while keeping the probability of detection equal to $1/2$. To target some ciphertexts $c_1, \ldots, c_s$, the first mix-server chooses random values $\delta_1, \ldots, \delta_s$ and replaces one of its outputs by the product $\prod_{i=1}^{s} c_i^{\delta_i}$. When the mix-net produces an output, the attacker identifies $s + 1$ plaintexts $m_1, \ldots, m_s, m^*$ in the final list that satisfy $m^* = \prod_{i=1}^{s} m_i^{\delta_i}$. The running time of the attack is clearly exponential in δ, so δ should be viewed as a reasonably small constant. Then it concludes that the ith targeted ciphertext is an encryption of m_i.

5 On the Need for Duplicate Removal Everywhere

Recall that duplicate ciphertexts in the list of initial ciphertexts are removed (preserving only the first posted copy). Jakobsson et al. do not emphasize that each mix-server must perform this operation before processing its input. In this section we explore the consequences of failing to do so in a Chaumian mix-net with RPC.

For simplicity, assume that the adversary targets the first s ciphertexts $c_{1,0}, \ldots, c_{s,0}$ and corrupts $s(s + 1)/2$ senders and the first and last mix-servers. For each i, the adversary removes the first layer of encryption by computing $c_{i,1} = \mathsf{Dec}_{sk_1}(c_{i,0})$, using the secret key of the first corrupted mix-server. He then makes i independent encryptions of $c_{i,1}$ under the public key of the first mix-server. This way, the adversary has prepared $1 + 2 + \ldots + s = s(s + 1)/2$ ciphertexts which will be sent by the same number of corrupted senders — each submitting one ciphertext. By construction there are $i + 1$ related ciphertexts of the ith targeted ciphertext. By looking at the input list of the last mix-server, the adversary can identify the targeted senders' ciphertexts (encrypted under the public key of the last mix-server) based on the number of duplicates. Since the last mix-server is corrupted, he learns the plaintexts of the targeted senders. As $s + s(s + 1)/2 \leq N$ must hold, this attack can break privacy of at most $\mathcal{O}(\sqrt{N})$ senders.

Jakobsson et al. do suggest to employ a CCA2 secure encryption scheme like OAEP-based RSA [1] to make the initial encryption non-malleable. The problem illustrated by the above attack is that the composition of the cryptosystems of the mix-servers only remain CCA2 secure as long as the first mix-server remain uncorrupted.

Clearly, the attack is prevented if every mix-server removes the duplicates before processing its input list. Notice that the final output list, i.e., the mixed output, can contain duplicates.

We do not see any way to extend the above attack to a homomorphic mix-net with RPC.

6 Inconsistent Commitments Are Dangerous

Jakobsson et al. [15] do not mention that it is essential to verify that those parts of the permutation commitments $\Gamma_j^{(In)}$ (or $\Gamma_j^{(Out)}$) that are opened must be consistent with a permutation. Unfortunately, this also turns out to be the way that implementors have interpreted the paper. A prominent research group in electronic voting generously gave us access to their private source code of their homomorphic mix-net with RPC and it suffered from this flaw. Another notable and publicly available example of an implementation with this flaw is the Civitas [4] election system (version 0.7.1), implemented by Clarkson et al. The Scantegrity scheme [3] employs a checking approach that resembles random partial checking and suffers from this flaw.

In this section we show how this flaw can be exploited to break either the correctness or the privacy, or a little bit of both, without detection.

6.1 Breaking Privacy without Detection

This attack applies only to the homomorphic mix-net with RPC. Recall our generalization of Pfitzmann's attack from Section 4 that breaks the privacy of s senders with detection probability $1/2$. We show how to mount a variation of this attack without being detected. The adversary targets some s ciphertexts $c_1, \ldots, c_s$. It chooses random values $\delta_1, \ldots, \delta_s$ and computes the product $c = \prod_{i=1}^{s} c_i^{\delta_i}$. Then it corrupts two senders and the first pair of mix-servers (here we assume that they are operated by a single entity). The two corrupt senders are asked to submit two ciphertexts that are re-encryptions of one another. The first mix-server behaves honestly and it is corrupted only in that it keeps track of the ciphertexts submitted by the corrupted senders and how they are re-encrypted. Let $c_{i_1,1}$ and $c_{i_2,1}$ be these ciphertexts in its output list and note that the adversary knows how to transform $c_{i_2,1}$ into $c_{i_1,1}$ by re-encryption.

Recall that $\gamma_{i,2}$ denotes a commitment to $\pi_2^{-1}(i)$ if $\mathcal{M}_2$ behaves honestly. The second mix-server $\mathcal{M}_2$ behaves honestly except that:

1. It defines $\gamma_{i_2,2}$ to be a commitment to $\pi_2^{-1}(i_1)$. The remaining $N-1$ commitments, including $\gamma_{i_1,2}$, a commitment to $\pi_2^{-1}(i_1)$, are computed in the usual way.
   Note that $\mathcal{M}_2$ can not open both $\gamma_{i_1,2}$ and $\gamma_{i_2,2}$ in a way that is consistent with a permutation since they are commitments to the same index i_1.
2. It replaces $c_{i_2,1}$ with the maliciously constructed ciphertext c. The modified list is then re-encrypted and shuffled to give the output list $(c_{1,2}, \ldots, c_{N,2})$.

The attacker has replaced the ciphertext of one of the corrupted senders with a re-encryption of c. If the attack goes undetected, the adversary can clearly identify the targeted senders' plaintexts as in the generalization of Pfitzmann's attack.

To see that the attack is not detected by RPC, first note that both commitments $\gamma_{i_1,2}$ and $\gamma_{i_2,2}$ verify correctly. Then observe that $c_{\pi_2(i_1),2}$ is in fact a re-encryption of both $c_{i_1,1}$ and $c_{i_2,1}$ and $\mathcal{M}_2$ can provide the randomness needed to verify this. The attack can be extended to break the privacy of rs senders without detection by using $r+1$ commitments of i_1 and introducing $r+1$ ciphertexts submitted by corrupted senders that are re-encryptions of each other.

Depending on the method used to decode messages, the attack will be detected when the output plaintexts are interpreted. Thus, in practice, the above attack could probably only be executed once before implementors identified the issue.

Interestingly, even if decommitted values are verified to be distinct, the above attack with two ciphertexts originating from a single ciphertext input by a corrupted party performs better than the attacks considered by Jakobsson et al. We discuss this is Section 7.

6.2 Rigging an Election without Detection

This attack applies to the homomorphic mix-net with RPC, and if the duplicates are not removed (see Section 5), it is also applicable to the Chaumian mix-net. A straightforward adaptation of the attack applies to the Scantegrity voting system [3].

The adversary corrupts the first sender and the first mix-server. The first sender is asked to use m as his plaintext. During the attack all other encrypted votes are replaced by m. The first mix-server simply replaces all the ciphertexts by $c_{1,0}$, i.e., the submitted ciphertext by the first sender which is an encryption of m. The modified list is then correctly transformed and shuffled to produce the output list $(c_{1,2}, \ldots, c_{N,2})$. To avoid being detected, the first mix-server chooses all the $\gamma_{i,1}$'s as commitments to 1, i.e., all of them are opened to 1. Clearly, the first mix-server can then provide the evidence that $c_{1,0}$ maps to all $c_{i,2}$'s. Therefore, the output list of the last mix-server is N encryptions of m and the attack is not detected.

To make the resulting output look less unrealistic, the attacker can of course submit encryptions of several different plaintexts and choose a suitable distribution over these and apply the attack for each such plaintext.

We stress that there is no way to notice this attack except manually inspecting the list of decommitted integers. Thus, this attack could in fact already have been exploited in executions of mix-nets with RPC, so we suggest that transcripts of old executions of such implementations are inspected manually.

7 What Is the Best We Can Hope for?

Note that even if duplicates are removed everywhere and it is verified that all *opened* commitments contain distinct integers to prevent the attacks of the previous sections, then this does not guarantee that all the unopened commitments are consistent with a permutation. We show that we can still replace ciphertexts in the homomorphic mix-net or eliminate in Chaumian mix-net with notably better probability than the attack considered optimal by Jakobsson et al.

The attack is essentially the same as in Section 6.2 except that only *one* ciphertext submitted by an honest sender is replaced by a copy of the ciphertext of the corrupted sender. Then the probability of detection is $1/4$, since the attack is only detected if the two commitments containing the same integer are opened. The attack can be repeated independently t times to replace t ciphertexts with probability $(3/4)^t$.

For the homomorphic mix-net, replacing ciphertexts translates to replacing the final plaintexts, i.e., the output of the mix-net. For Chaumian mix-net our replacements corresponds to replacing the final plaintext if the last mix-server in the chain makes the replacements; otherwise, since the duplicates are removed, replacing ciphertexts results in eliminating plaintexts from the final mixed output.

8 On the Universal Verifiability of RPC

Recall that a mix-net is called universally verifiable if it ensures correctness even if the adversary corrupts all parties, and consider the case where checking takes place at the end of the mixing. If all mix-servers are corrupted, then they can try to cheat by repeatedly: replacing t ciphertexts in their output, picking a random seed as in the protocol, and checking if the resulting challenges can be answered. The attack is successful if a replacement is found that passes the correctness check. Assuming the adversary repeats the procedure q times, Jakobsson et al. argue that the success probability in each attempt is bounded by 2^{-t} and then apply the union bound to conclude that the probability of success is roughly $1 - (1 - 2^{-t})^q \leq q2^{-t}$.

8.1 An Improved Attack

There is a more clever attack on the public verifiability of homomorphic mixing with RPC. A straightforward application of the idea of Section 7 shows that the success probability can be increased to roughly $1 - (1 - (3/4)^t)^q$. In other words, the adversary can change $1/\log(4/3) \approx 2.4$ times as many ciphertexts as claimed for a given computational complexity. For example, to change $t = 192$ ciphertexts the adversary has to prepare and do 2^{80} hash queries. This should be compared with the claimed bound of 80 ciphertexts for corresponding complexity. This attack also applies to the Chaumian mixing even if the duplicates are removed. This requires that the replacement is done by the last mix-server.

8.2 When Checking Is Performed at the End of the Mixing

Universal verifiability is considerably weaker when in-phase checking is used. The problem is that in-phase checking allows the adversary to replace a few ciphertexts in each each mix-server. Replacing kt ciphertexts in this way goes undetected with probability roughly $\left(1 - (1 - (3/4)^t)^q\right)^k$ (instead of $1 - (1 - (3/4)^{tk})^q$) for homomorphic mixing. For example, with $k = 5$, $t = 192$, and roughly 2^{80} queries to the hash function, the adversary can replace almost one thousand ciphertexts. If we really care about public verifiability, then this is unlikely to be acceptable. This also holds for Chaumian mixing even if the duplicates are removed everywhere but the replaced ciphertexts are eliminated except for the last mix-server.

This leaves us with two options for Chaumian mix-nets: either we adopt checking in-phase with mixing and accept the weaker guarantee on universal verifiability, or we use checking after mixing.

There is a fairly obvious attack on a Chaumian mix-net with RPC if the checking is performed at the end of the mixing and this is why Jakobsson et al. do not suggest this type of checking. The problem is that the adversary can corrupt the first mix-server, replace some ciphertexts, and simply wait for the output of the mix-net. Then he will be caught, but before this happens the votes of the targeted voters have already been revealed.

To prevent this attack and still use checking after mixing, we propose to protect senders plaintexts with an innermost cryptosystem whose secret is shared between the mix-servers and only recovered after the checking has been successfully performed. This can also be implemented by letting each server generate an additional key pair and letting the joint key be the list of all the additional public keys. This avoids the need for a costly distributed key generation protocol, but increases the size of ciphertexts. Another problem with this scheme is that the execution can only be aborted if cheating is detected.

Note that the problem with checking at the end of the mixing does not appear in a homomorphic mix-net with RPC since nothing is revealed about the plaintexts until after the checking is completed successfully.

9 On the Provable Security of RPC

Even cryptographic protocols proposed without a proof of security by experienced cryptographers can often be broken, and this seems to be particularly true for mix-nets. Historically the proposals of heuristically secure mix-nets [20,11,12,14,9] have been followed by discovery of security flaws [22,21,5,17,24].

A formal proof of security does not guarantee that no attack will ever be found (proofs can have subtle errors, assumptions can be wrong, and the adversarial model can be unrealistic), but it increases the confidence in the security of the scheme significantly.

9.1 Homomorphic Mix-Net with RPC

We argue informally that homomorphic mix-nets with RPC, e.g., based on El Gamal cannot be proven secure using a blackbox reduction in the simulation paradigm, even if the issues explored in this paper are handled correctly.

A definition of security in the simulation paradigm requires that no efficient distinguisher can tell a suitable ideal model with a simulator from a real model with a real adversary. Suppose that there is a real adversary and a distinguisher that contradicts this claim. We must use them to break the semantic security of the cryptosystem. The first step would be to do a hybrid argument such that two hybrids only differ in that in one of the hybrids the ith voter encrypts his true message and in the other he encrypts some bogus message, e.g. zero. Then to exploit the adversary in a blackbox way we would:

1. Accept a public key from the semantic security experiment as input and somehow embed this into the public key used by voters. We could for example pretend that the public key belongs to one of the honest senders and simulate the verifiable secret sharing of the secret key without knowing the secret key at all.
2. Simulate the execution until the ith voter prepares its ciphertext and interrupt the execution at this point. Then we hand the true plaintext and the bogus plaintext to the experiment and wait for a ciphertext in return which is used as the ciphertext of the ith sender in the continued simulation.
3. Simulate the decryption of the given ciphertext to the true plaintext, and output the output of the distinguisher. (In a homomorphic mix-net we must keep track of how it is permuted through the mix-net to be able to do this.)

The problem with the homomorphic mix-net with RPC is that a corrupted mix-server with probability $3/4$ can replace a ciphertext by *any* ciphertext and before the distinguisher outputs its result, the adversary expects to see the plaintext of this ciphertext in the output of the mix-net. Thus, the adversary is given access to a restricted decryption oracle *before* the distinguisher guesses which model it is interacting with. In other words, for the adversary to be useful to the reduction, the simulator must solve almost the same problem as the reduction is intended to do.

9.2 Chaumian Mix-Net with RPC

Proving the security of the Chaumian mix-net with RPC based on a CCA2 secure cryptosystem where the vulnerabilities explored in this paper are resolved seems hard, but not impossible. The challenge is to capture the restricted forms of privacy and soundness that it exhibits. The obvious way to resolve the problem with limited privacy is to assume that there are sufficiently many mix-servers to get full privacy, but it turns out to be non-trivial to determine how many mix-servers are needed.

Gomulkiewicz et al. [10] study the probability distribution of the permutations linking the input and outputs of a mix-net with RPC given the information revealed. They show that the distance between this distribution and the uniform distribution is $\mathcal{O}(\frac{1}{N})$ even when there is only a constant number of mix-servers. Contrary to what is claimed, this result does not capture the privacy of a mix-net with RPC, but it may well be a useful result.

10 Interpretation and Discussion

It is easy to add consistency checks to the random partial checking protocol, but such consistency checks are missing in the description of Jakobsson et al. [15]. Implementers should of course follow the description of a cryptographic protocol strictly to make sure that their implementation capture the intentions of the protocol designers, so it is not surprising that the consistency checks are missing in all implementations we have considered. Furthermore, even if the needed consistency checks are added the protocol still does not achieve the claimed security guarantees. Thus, we think it is fair to consider the issues we have identified as *flaws in the protocol* and not as mere implementation bugs, although we realize that different interpretations are possible. Both the Civitas [4] and the Scantegrity [3] teams have reported that they have already mended, or are about to mend, their implementations based on our insights.

Our attack on correctness is easily generalized to be very difficult for a human to discover by a manual sanity check of the values revealed during random partial checking. Thus, we choose to view our attack as *undetectable* in practice, despite that it can be detected using an algorithm that performs the needed consistency checks.

It is hard to exaggerate how fortunate we are to be able to *retroactively verify* that the attack on correctness did not take place in a given execution. We strongly suggest that all implementers of random partial checking (or similar schemes) perform the needed verifications for all conducted elections. The Scantegrity team has already reported that no tampering took place in elections using their scheme.

We stress that all the issues described in this paper were found in an attempt to prove the security of random partial checking and we are convinced that any attempt to do so would have revealed the same issues. Thus, we think that our work illustrates the importance of precise descriptions and rigorous proofs of security. Proofs of security can have subtle bugs and models can be unrealistic, but we think that protocols without proofs of security should not be trusted.

Acknowledgments. Tal Moran contributed to our initial discussions on possible approaches to prove the security of random partial checking. Johan Håstad gave helpful comments. Members of the Civitas and the Scantegrity teams quickly confirmed our findings and provided valuable feedback on a draft of this paper.

References

1. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
2. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–88 (1981)
3. Chaum, D., Essex, A., Carback, R., Clark, J., Popoveniuc, S., Sherman, A., Vora, P.: Scantegrity: End-to-end voter-verifiable optical- scan voting. In: IEEE Security and Privacy, vol. 6, pp. 40–46 (2008)
4. Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. In: IEEE Symposium on Security and Privacy, pp. 354–368. IEEE Computer Society (2008)
5. Desmedt, Y., Kurosawa, K.: How to Break a Practical MIX and Design a New One. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 557–572. Springer, Heidelberg (2000)
6. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: FOCS, pp. 427–437. IEEE Computer Society (1987)
7. Furukawa, J., Sako, K.: An Efficient Scheme for Proving a Shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
8. Gabber, E., Gibbons, P.B., Matias, Y., Mayer, A.J.: How to Make Personalized Web Browsing Simple, Secure, and Anonymous. In: Hirschfeld, R. (ed.) FC 1997. LNCS, vol. 1318, pp. 17–32. Springer, Heidelberg (1997)
9. Golle, P., Zhong, S., Boneh, D., Jakobsson, M., Juels, A.: Optimistic Mixing for Exit-Polls. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 451–465. Springer, Heidelberg (2002)
10. Gomułkiewicz, M., Klonowski, M., Kutyłowski, M.: Rapid Mixing and Security of Chaum's Visual Electronic Voting. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 132–145. Springer, Heidelberg (2003)
11. Jakobsson, M.: A Practical Mix. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 448–461. Springer, Heidelberg (1998)
12. Jakobsson, M.: Flash mixing. In: PODC, pp. 83–89 (1999)
13. Jakobsson, M., Juels, A.: Mix and match: Secure function evaluation via ciphertexts. In: Okamoto [19], pp. 162–177
14. Jakobsson, M., Juels, A.: An optimally robust hybrid mix network. In: PODC, pp. 284–292. ACM Press, New York (2001)
15. Jakobsson, M., Juels, A., Rivest, R.L.: Making mix nets robust for electronic voting by randomized partial checking. In: Boneh, D. (ed.) USENIX Security Symposium, pp. 339–353. USENIX (2002)
16. Jakobsson, M.: Mix-Based Electronic Payments. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 157–173. Springer, Heidelberg (1999)

17. Mitomo, M., Kurosawa, K.: Attack for flash mix. In: Okamoto [19], pp. 192–204
18. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: CCS 2001: Proc. of the 8th ACM Conference on Computer and Communications Security, pp. 116–125. ACM, New York (2001)
19. Okamoto, T. (ed.): ASIACRYPT 2000. LNCS, vol. 1976. Springer, Heidelberg (2000)
20. Park, C., Itoh, K., Kurosawa, K.: Efficient Anonymous Channel and All/Nothing Election Scheme. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
21. Pfitzmann, B.: Breaking an Efficient Anonymous Channel. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 332–340. Springer, Heidelberg (1995)
22. Pfitzmann, B., Pfitzmann, A.: How to Break the Direct RSA-Implementation of Mixes. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 373–381. Springer, Heidelberg (1990)
23. Sako, K., Kilian, J.: Receipt-Free Mix-Type Voting scheme — A Practical Solution to the Implementation of a Voting Booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
24. Wikström, D.: Five Practical Attacks for "Optimistic Mixing for Exit-Polls". In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 160–175. Springer, Heidelberg (2004)

Randomly Failed! The State of Randomness in Current Java Implementations

Kai Michaelis, Christopher Meyer, and Jörg Schwenk

Horst Görtz Institute for IT-Security, Ruhr-University Bochum
{kai.michaelis,christopher.meyer,joerg.schwenk}@rub.de

Abstract. This paper investigates the Randomness of several Java Runtime Libraries by inspecting the integrated Pseudo Random Number Generators. Significant weaknesses in different libraries including Android, are uncovered.

1 Introduction

With a market share of 33-50% [1], [2], [3] Android is currently the most popular mobile OS. Each of the 331-400 million sold devices (cf. [3]) is able to run Java applications written in Java. Java provides interfaces for different Pseudo Random Number Generators (PRNGs), such as `SecureRandom`, which is intended for use by cryptographic schemes. But, only the API is specified - each library implements own algorithms. Designing secure and reliable PRNGs is a hard and complicated task [4], as well as implementing these algorithms correctly. One of the worst ideas is to implement own unproved PRNG constructs.

This paper examines the quality of the random numbers generated by common Java libraries. In detail, the PRNGs, intended for use in cryptographic environments, of Apache Harmony[1], GNU Classpath[2], OpenJDK[3] and BouncyCastle[4] are inspected. It is shown that the over-all entropy of the Android PRNG can be reduced to 64 bits. Beyond this, multiple weaknesses of entropy collectors are revealed. However, some of these weaknesses only occur under special conditions (e.g., unavailable `/dev/{u}random` device). We clearly point out that we are not going to discuss the quality of random numbers generated by PRNGs shipped with Operating Systems (OS). Discussions of OS provided (P)RNG facilities can be found at e.g., [5], [6], [7].

2 Related Work

Problems related to weak pseudo random number generators (PRNGs) have been topic of several previously published papers. In 2012 Argyros and Kiayias

[1] `http://harmony.apache.org/`
[2] `http://www.gnu.org/software/classpath/`
[3] `http://openjdk.java.net/`
[4] `http://www.bouncycastle.org/`

investigated in [8] the state of PRNGs in PHP[5] and outlined flaws leading to attack vectors. Their results are based on insecure constructions of PRNGs introduced by custom algorithms.

Kopf outlined in [9] multiple cryptographic and implementational flaws in widespread Content Management Systems (CMS). These observations focused weak cryptographic constructs and pecularities of PHP. The resulting bugs are not caused by weak PRNGs, but by vulnerable custom algorithms in combination with implementational flaws.

Meyer and Somorovsky [10] uncovered a vulnerable use of `SecureRandom` in WSS4J[6]. A function responsible for nonce generation, used at various places in the framework, suffered from weak PRNG seeding.

Problems related to PRNGs are also topic of multiple CWEs (Common Weakness Enumeration)[7] that deal with the misuse or use of weak pseudo random number generators (cf. CWE 330-343).

In [11]) Lenstra et al. inspected millions of public keys of different types (RSA, DSA, ElGamal and ECDSA) and found keys violating basic principles for secure cryptographic parameters. According to the authors these weak keys could be a result of poorly seeded PRNGs.

More alarming results concerning key quality are presented by Heninger et al. in [7] pointing out that *"Randomness is essential for modern cryptography"*. Missing entropy was identified as a root cause for many weak and vulnerable keys. Additionally, the authors identified weak entropy problems under certain conditions in the Linux RNG.

A more theory based cryptanalytic set of attacks on PRNGs can be found at e.g., [12].

2.1 Contribution

The results of this paper focus on PRNG implementations of Java core libraries. Even thus all libraries implement the same functionality - generating (pseudo) *random* numbers - based on the same API, the algorithms for random number generation differ from library to library.

The main contribution is an analysis of algorithms implemented in most commonly used `SecureRandom` generators. For this analysis tests provided by the Dieharder [13] and STS [14] testsuites are used to check for cryptographic weaknesses. Additionally manual code analysis uncovered algorithmical and implementational vulnerabilities.

3 Implementations and Algorithms

In Java, random numbers are derived from an initial seed. Two instances of a PRNG seeded with equal values always generate equal *random* sequences. The

[5] http://www.php.net
[6] http://ws.apache.org/wss4j/
[7] http://cwe.mitre.org

cryptographic strong PRNGs are accessed via the `SecureRandom` interface which is part of the Java Cryptography Architecture.

3.1 Apache Harmony - Android's Version of `SecureRandom`

Apache Harmony, introduced in 2005 as an open source implementation of the Java Core Libraries published under the Apache License[8], became obsolete with the publication of SUN Microsystems' reference implementation in 2006. Although discontinued since 2011, the project is further devleoped as part of Google's Android platform.

```
// require cnt: counter >= 0, state: seed bytes and iv: previous output
iv = sha1(iv,concat(state,cnt));
cnt = cnt + 1;
return iv;
```

Listing 1.1. Apache Harmony's SecureRandom

The PRNG shipped with Android uses the SHA-1 [15] hash algorithm to generate pseudo random sequences and is ideally seeded by the random devices provided by the OS. Random numbers are generated by calculating hash sums of an internal state concatenated with a 64 bit integer counter and a padding (algorithm in Listing 1.1). The counter is, starting at zero, incremented each run of the algorithm. Additionally, a padding is required to fill remaining bits of a 85 bytes buffer. This padding follows the SHA-1 padding scheme: The last 64 bits hold the length *len* of the values to be hashed in bits. Any space between the end of the values and the length field is filled with a single '1' bit followed by zeros. The resulting hash sums are returned as pseudo random sequence. Figure 1 illustrates the state buffer.

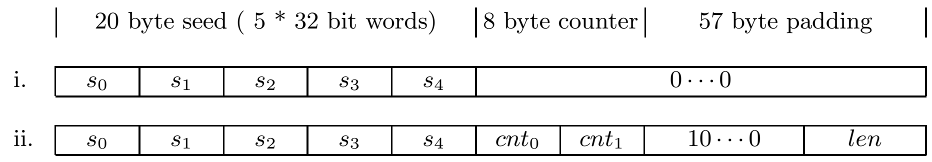


Fig. 1. The seed bytes $s_0, .., s_4$ in row i concatenated with a 64 bit counter c_0, c_1 (two 32bit words), padding bits and the length *len* as in row ii are hashed repeatedly to generate a stream of pseudorandom bytes

3.2 GNU Classpath

The GNU Classpath project started in 1998 as the first open source implementation of Java's core libraries. The library is licensed under GPL [9] and is e.g., partly used by the IcedTea [10] project.

[8] `http://www.apache.org/licenses/`
[9] `http://www.gnu.org/licenses/`
[10] `http://icedtea.classpath.org/wiki/Main_Page`

```
// require state: <= 512 bit buffer, iv: current Initialzation Vector
byte[] output = sha1(iv,state);
state = concat(state,output);
if(state.length > 512) {          // in bits
    iv = sha(iv,state [0:512]) ;  // first 512 bits
    state = state[512:-1];        // rest
    output = sha1(iv,state);
}
return output;
```

Listing 1.2. GNU Classpath's SecureRandom

The SecureRandom implementation (algorithm in Listing 1.2) of GNU Classpath is powered by a class MDGenerator that is parameterized with an arbitrary hash algorithm. A new SecureRandom instance is seeded with 32 bytes yielding an internal state as shown in row i of Figure 2. Based on this start value a digest is computed. The resulting hash (e.g. 160 bit in case of SHA-1) is returned as pseudo random value r_0. r_0 in turn is concatenated with the former seed forming the new state in row ii.

These bytes are hashed again yielding the second output r_1. Finally, the seed concatenated with the previous two hash values form the new state (c.f. row iii) whose digest is the third output value r_3. r_3 is again appended to the previous state resulting in the state illustrated in row iv. Each fourth r_i values a block overflow happens causing the implementation to hash the full block and use this hash as initialization vector (IV) for the new block. The only unknown value is the 32 byte long initial seed. All other information are known (as they are parts of former r_i value).

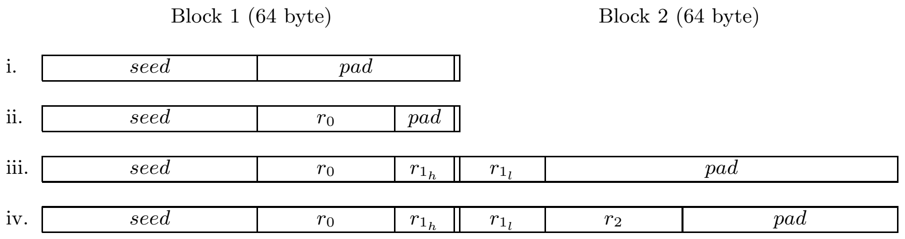


Fig. 2. Hashing the previous pseudo random bytes concatenated with the fomer seed produces the next output value

GNU Classpath includes a "backup" seeding facility (algorithm in Listing 1.3) for Unix-like operating systems. The VMSecureRandom class is able to harvest entropy from the OS' process scheduler. Every call to the seeding facility starts 8 threads, each one incrementing a looped one byte counter. The parent thread waits until at least one of the 8 threads is picked up by the scheduler. The seed value is calculated by XORing each of the one byte counters. This forms the first seed byte. Accordingly, the next seed byte is generated the same way. When the requested amount of seeding material is gathered the threads are stopped.

```
int n = 0
byte[] S = new byte[8];
byte[] output = new byte[32];

for(int i = 0; i < 8; i++) {
    S[i] = start_thread(n); // ''spinner'' incrementing a counter starting at n
    n = (2*n) % pow(2,32);
}

while(!spinners_running())
    wait();

for(int i = 0; i < 32; i++) {
    output[i] = 0;

    for(int j = 0; j < 8; j++)
        output[i] = output[i] ^ counter(S[j]);
}

for(int i = 0; i < 8; i++)
    stop_thread(S[i]);

return output;
```

Listing 1.3. GNU Classpath's entropy collector

3.3 OpenJDK

As the direct successor of the Java Development Kit (JDK), OpenJDK [11] provides not only Java core libraries, but additionally a Java compiler and a virtual machine. OpenJDK is the official reference implementation of Java and open source licensed under GPL. The project is supported by major vendors such as IBM or SAP.

```
// require state: seed bytes, iv: SHA-1 standard IV
byte[] output = sha1(iv,state);
byte[] new_state = (state + output + 1) % pow(2,160);

if(state == new_state)
    state = (new_state + 1) % pow(2,160);
else
    state = new_state;

return output;
```

Listing 1.4. OpenJDK's SecureRandom

OpenJDK uses the OS' PRNG in conjunction with a similar scheme as the previously mentioned libraries (algorithm in Listing 1.4). An initial 160 bit seed

[11] http://openjdk.java.net/

value is hashed using SHA-1. The result is XORed with the output of the OS specific PRNG and returned as pseudo random bytes. This output is added to the initial seed plus 1 modulo 2^{160}. An additional check compares the new seed to the old one preventing the function from getting trapped in a fixed state where $s_i + \mathit{SHA\text{-}1}(s_i) + 1 \equiv s_i \bmod 2^{160}$.

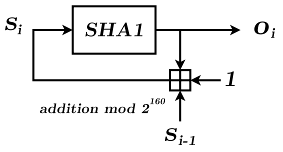


Fig. 3. SecureRandom of OpenJDK. In each step the current state (seed) s_i is compressed yielding output o_i. The sum of o_i, s_{i-1} and 1 give new state s_{i+1}.

The integrated entropy collector (cf. Listing 1.5) uses multiple threads to gather randomness. In contrast to GNU Classpath, only one thread increments a counter. Subsequently, new threads are started suspending five times for 50ms to keep the scheduler busy. Before continuing the lower 8 bits of the current counter value pass an S-Box. The XOR sum of all 5 counters is returned as random byte. The entropy collector enforces mandatory runtime (250ms) and counter value (64000). Even after enough seed is produced the entropy collector continues to run. The seed bytes are hashed together with entries of `System.properties` and the result is used as seed.

```
counter = 0;
quanta = 0;
v = 0;

while(counter < 64000 && quanta < 6) {
    start_thread()  // loops 5 times, sleeping 50ms each
    latch = 0;
    t = time();

    while(time() − t < 250ms) // repeat for 250ms
        latch = latch + 1;

    counter = counter + latch;
    v = v ^ SBox[latch % 255];
    quanta = quanta + 1;
}

return v;
```

Listing 1.5. OpenJDK's entropy collector

3.4 The Legion of Bouncy Castle

Bouncy Castle is not a complete core library, but a security framework for the Java platform. This includes a `SecurityProvider` as well as cryptographic APIs for standalone usage. The project provides enhanced functionality, as well as support for a broader range of cryptographic algorithms compared to default OpenJDK. Bouncy Castle implements various pseudo random generators as well as a threaded entropy collector akin of the one in OpenJDK.

The `DigestRandomGenerator` (cf. Listing 1.6) uses a cryptographic hash algorithm to generate a pseudorandom sequence by hashing an internal secret state of 160 bits with a 64 bit *state counter* producing 160 bits of pseudorandom output at once. After each hash operation the state counter is incremented. The initial secret state is received from a seeding facility. Every tenth hash operation on the state, a 64 bit *seed counter* is incremented and a new secret state is generated by hashing the current state concatenated with the seed counter. This new secret state replaces the old one, where the state counter remains at its previous value. When another pseudorandom value is requested from the `DigestRandomGenerator` instance, this new secret state is hashed with the state counter, producing a cryptographic checksum to be returned to the caller as random byte array.

```
// require seedBuffer: 160bit seed, stateBuffer: 160bit array, seedCounter and
      stateCounter: 64bit integers

if(stateCounter % 10 == 0) {
    stateBuffer = sha1(iv,concat(seedBuffer,seedCounter));
    seedCounter += 1;
}

byte[] output = sha1(iv,concat(stateBuffer,stateCounter));
stateCounter++;

return output;
```

Listing 1.6. DigestRandomGenerator

The `VMPCRandomGenerator` is based on Bartosz Zoltak's Variable Modified Permutation Composition one-way function [16].

The `ThreadedSeedGenerator` implements a threaded entropy collector scheme. Only two threads are used: one thread increments a counter in a loop, whereas the other waits 1ms until the counter has changed. The new value is appended to an output array. The incrementing thread is teared down after all random bytes are collected. The generator offers two modes of operation: a) "slow" mode where only the least significant bit of every byte is used and b) "fast" mode where the whole byte is used.

```
// require count: number of seed bytes needed, fast: enable "fast" mode
byte[] output = byte[count];
t = start_thread() // increments a counter in a loop
int last = 0;

// use bits in "slow" mode
if(!fast)
    count *= 8;

for(int i = 0; i < count; i++) {
    while(counter(t) == last)
        sleep(1);

    last = counter(t);
    if(fast)
        output[i] = (byte)last;
    else
        output[i/8] = (last % 2) | (output[i/8] << 1);

}
stop_thread(t);
return output;
```

Listing 1.7. ThreadedSeedGenerator

4 Methodology

Manual code review was performed for each of the introduced PRNGs. During code review the code was checked for implementation flaws and obvious bugs. Aside from code review blackbox tests on the output were preformed to grade the entropy. For this the Dieharder test suite [13] for (pseudo) random number generators was used, as well as Monobit, Runs, and Serial tests from the STS [14] suite.

While these tests can not replace cryptanalysis they still uncover bias and dependency in the pseudo random sequence. For every test exists an expected distribution of outcomes. Test runs produce a value that is compared to the theoretical outcome. A p-value, describing the probability that a real RNG would produce this outcome, between 0 and 1 is computed. A p-value below a fixed significance level $\alpha = 0.001$ indicates a failure of the PRNG with probability $1 - \alpha$. Dieharder differs from this methodology as it relies on multiple p-values to evaluate the quality of a PRNG. It is possible (and expected) for a good RNG to produce "failing" p-values. Instead of grading a single outcome, 100 p-values are computed and the distribution of these values is compared to an uniform one. This generates a final p-value grading the quality of the PRNG.

No cryptanalysis was performed - we analyzed the algorithms and evaluated the quality of randomness by using special purpose testsuites.

5 Results

This section highlights prospective weaknesses of the implementations and evaluates the quality of generated randomness. Due to space limitations, only conditions targeting the observed weaknesses are regarded - the statistical graphs can be found in the Appendix in Section 6.

Striking about all implementations of `SecureRandom` is the limited state size. In OpenJDK and GNU Classpath adding more entropy (> 160bit) to an instance will not enhance security. This limitation is alarming, since it renders the PRNGs useless for key generation > 160 bit (as e.g., in AES' case). Only Apache Harmony relies on a 512 bit buffer.

5.1 Apache Harmony

Apache Harmony revealed multiple weaknesses caused by implementation bugs. As a part of Android a plethora of cryptographic functions [17] rely on this PRNG. One of the bugs addresses directly the Android platform, where as the second one only targets Apache Harmony.

Weaknesses. *FIRST* - When creating a *self seeding* `SecureRandom` instance (by calling the constructor without arguments and subsequent `setSeed()` call), the code fails to adjust the byte offset (a pointer into the state buffer) after inserting a start value. This causes the 64 bit counter and the beginning of the padding (a 32 bit word) to overwrite parts of the seed instead of appending to it. The remaining 64 bits of entropy render the PRNG useless for cryptographic applications[12].

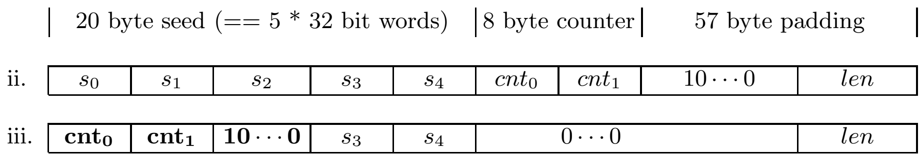

20 byte seed (== 5 * 32 bit words)
8 byte counter
57 byte padding
ii.
s_0 s_1 s_2 s_3 s_4 cnt_0 cnt_1 $10\cdots0$ len
iii.
$\mathbf{cnt_0}$ $\mathbf{cnt_1}$ $\mathbf{10\cdots0}$ s_3 s_4 $0\cdots0$ len

Fig. 4. Instead of appending (c.f. row ii), the counter and the succeeding padding overwrite a portion of the seed, yielding row iii

SECOND[13] - When running under a Unix-like OS a new `SecureRandom` instance is seeded with 20 bytes from the *urandom* or *random* device. If both are unaccessible the implementation provides a fall-back seeding facility (cf. Listing 1.8): seed is gathered from the `random()` PRNG of the GNU C library, which is seeded it via `srandom()` with the UNIX-time, processor time and the pointer value of a heap-allocated buffer. After seeding, `random()` is used to generate seed bytes for `SecureRandom`. Before these bytes are returned the most significant bit is set to zero (mod 128) - this behavior is neither documented, expected nor explained.

[12] The bug was communicated to the Google Security Team.
[13] This bug is not part of the Android Source.

```
char *seed = malloc(20);
srandom(clock() * time() * malloc()) % pow(2,31));
for(int i = 0; i < 20; i++)
    seed[i] = random() % 128;

return seed;
```

Listing 1.8. Apache Harmony's `getUnixSystemRandom`

The missing entropy is not compensated (e.g., by requesting > 20 bytes). As a consequence, the effective seed of a `SecureRandom` instance is only 7/8 for each requested byte, degrading security (of only 64 bits due to the *first* bug) by another 8 bits to 56 bits (s_3 and s_4 are 2 * 32 bit words == 8 byte). Even worse, the argument of `srandom()` in the GNU C library is of type `unsigned int` while Harmony reduces the argument modulo `INT_MAX` (defined in *limits.h*) - the maximum value for *signed int*s. This limits the entropy of a single call to the seeding facility to 31 bits.

Quality of Entropy Collectors. Generating 10MiB of seed - two consecutive bytes are interpreted as a single point - lead to the chart in Figure 9. It shows a lack of values above 127 in each direction. This test targets the *second* bug. The *first* bug limits the security to 64/56 bit, depending on the seeding source. This *seed space* of only 2^{64} elements is within reach of Brute-Force attacks utilizing GPGPUs/FPGAs (cf. [15]).

5.2 GNU Classpath

The library's entropy collector revealed inconsistencies regarding normal distribution of the output bits which could result in vulnerabilities.

Weaknesses. - The implementation of GNU Classpath contains a significant weakness related to internal states. As long as a new generated random value concatenated with the previous state does not overflow the current block, all hash computations are done with the same IV. States in rows i and ii from Figure 2 are both hashed with the SHA-1 standard IV. The obtained state in row iii overflows the first block - the hash value of this first block is used as input for the hash of the second block in row iii and as IV for the succeeding computation of $r_{1_l}|r_2$ concatenated with r_3. The state is reduced from 32 bytes seed to only 20 unkown IV bytes.

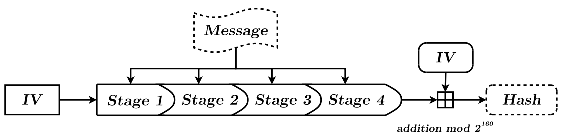


Fig. 5. Schematic view of a single SHA-1 iteration. Values in the dotted boxes are known when used in the SecureRandom class from GNU Classpath.

The IV is the only unknown value. To break the algorithm an attacker has to discover the IV for a known message producing a known checksum. While no such attack on SHA-1 has been published yet, this scenario is different from a preimage attack (c.f., [18], [19]). The compression function in SHA-1, ignoring the final 32 bit additions is invertible(c.f., [20]) for a given message. Thus, the addition of the secret IV (see Figure 5) remains the only hurdle in breaking the implementation.

Quality of Entropy Collectors. The seeding facility harvests entropy from multiple threads competing for CPU time. While the behaviour of the scheduler is difficult to predict it is possible to influence it. Only one of eight threads is expected to be scheduled. During seed extraction this precondition is not checked, enabling an attacker to fill (parts) of the output array with identical values by preventing threads to run (e.g., by creating high process load).

To test this construction under worst conditions, 11GiB of seed values were generated. To simulate high load 8 processes were run simultaneously. Each process queried for 16384 bytes while iterating in a loop. At first inspection, the resulting random seed revealed large (up to 2800 bytes) "holes" where random bytes were equal. As a result, the algorithm did not pass any blackbox tests. While the test conditions were extreme, they still expose a weakness in this entropy harvester.

The first 10MiB of seed are sampled on a graph (c.f., Figures 10). As can be seen, Classpath was unable to fill the whole space, leaving 64 by 64 large patches in the second and forth quadrant, as well as 32 by 32 along the diagonal when running under heavy load. In contrast, under normal conditions the entropy collector produced a well-balanced pane.

5.3 OpenJDK

The overall impression of `SecureRandom`'s reference implementation suggests a thoughtful and mature implementation.

Weaknesses. Code review bared no obvious weaknesses. Still, if an attacker is able to learn any internal state (s_i in Figure 3) all following states $s_j \quad \forall j > i$ and outputs $o_j = \mathit{SHA\text{-}1}(s_j)$ can be predicted if the OS PRNG (*/dev/*{*u,*}*random*) is unavailable[14].

Quality of Entropy Collectors. OpenJDK's strategy for seed generation in abstance of OS support is similar to `VMSecureRandom` of GNU Classpath. OpenJDK supplements the threaded entropy collector by enforcing minimal limits on runtime before extracting bytes and adding a substitution box. The unhashed[15] seed bytes were evaluated. The implementation revealed to be magnitudes slower, resulting in only 20MiB of seed generated by 8 processes running simultaneously. From the 114 blackbox tests only 30 were passed, whereas 12 failed with a p-value < 0.05 and 72 with p-value $< 10^{-6}$. The random bytes had grave difficulties with the STS tests, failing Monobit, Runs and the first eight Serial tests. This indicates poor variance in single bits and tuples up to eight bits. Nevermind, the resulting graph is filled very balanced (cf. Figure 11).

[14] Until the user manually reseeds the SecureRandom instance.

[15] Before hashing with additional `System.properties` values.

5.4 The Legion of Bouncy Castle

Bouncy Castle's implementation is also a hash-based algorithm (`DigestRandomGenerator`). No obvious bugs were found during code review. In contrast, the entropy collector `ThreadedSeedGenerator` revealed difficulties to generate sufficient random bytes.

Weaknesses. In `DigestRandomGenerator` the seed is modified every tenth call (cf. Section 3.4). This may increase the period of the PRNG and hinders an attacker aware of the secret state to calculate previous outputs. Predicting all succeeding outputs is still possible as the counter values can be guessed by observing the amount of values produced yet.

The VMPC function used in `VMPCRandomGenerator` is known to be vulnerable to multiple distinguishing attacks (cf. [21], [22], [23]).

Quality of Entropy Collectors. The entropy collector checks if a counter incremented in another thread has changed. Under heavy load the counter often differs only about 1 incrementation.

The seed generator running in "fast" mode passed most of the STS tests including Monobit, Runs and Serial up to eight bit tuple size. In "slow" mode, long sequences of ones and zeros caused to fail the Runs test, as well as the Serial test for 2-bit blocks. Both modes were still able to fill the pane after 10MiB of one byte samples (c.f., Figure 12).

5.5 Vulnerabilities Summary

The Tables depitcted in Figures 6, 7, 8 finally summarize our results[16].

Library	Security when seeded from	
	OS PRNG	integrated source
Apache Harmony	64 bit	31 bit
GNU Classpath	160 bit	dubious$^{(\text{see text})}$
OpenJDK	160 bit	dubious$^{(\text{see text})}$

Fig. 6. Summary of the results from SecureRandom audits

Library	Source of entropy	Passed Blackbox tests
Apache Harmony	UNIX-time, processor time, heap pointer	0/114
GNU Classpath	Thread scheduler	0/114$^{(\text{see text})}$
OpenJDK	Thread scheduler, `System.properties`	30/114

Fig. 7. Summary of the integrated seed generators

[16] Bouncy Castle is partly missing since it ships with replacements for `SecureRandom` and configurable entropy collectors. And is thus not directly comparable.

Library	**Component**	**Vulnerability**	**only if OS PRNG unavail.**
Apache Harmony	`SecureRandom`	Entropy limited to 64bit.	
Apache Harmony	`SecureRandom`	Entropy limited to 31bit.	X
Gnu Classpath	`SecureRandom`	Possibly predictable if later IVs are known.	
Gnu Classpath	Entropy Collector	Suggestible by other threads.	X
Bouncy Castle	`VMPCRandomGenerator`	Vulnerable to distinguishing attacks.	

Fig. 8. Overall summary of all uncovered vulnerabilities

6 Conclusion

The `SecureRandom` PRNG is the primary source of randomness for Java and is used e.g., by cryptographic operations. This underlines its importance regarding security. Some of fallback solutions of the investigated implementations revealed to be weak and predict- or capable of being influenced. Very alarming are the defects found in Apache Harmony, since it is partly used by Android. Long update intervals and missing pre-OS-release patching of device manufacturers in the past (c.f., [24]) may cause this bug to remain in the Android ecosystem for months or even years.

Although, some libraries provide acceptable randomness, the use of hardware RNGs instead of PRNGs is recommended for critical purposes.

References

1. Gupta, A., Cozza, R., Nguyen, T.H., Milanesi, C., Shen, S., Vergne, H.J.D.L., Zimmermann, A., Lu, C., Sato, A., Glenn, D.: Market Share: Mobile Devices, Worldwide, 1Q12. Technical report, Garnter, Inc. (May 2012)
2. Nielsen: Two Thirds of New Mobile Buyers Now Opting For Smartphones. Technical report, The Nielsen Company (June 2012)
3. Bennett, J.: Android Smartphone Activations Reached 331 Million in Q1 2012 Reveals New Device Tracking Database from Signals and Systems Telecom. Technical report, Signals and Systems Telecom (May 2012)
4. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM (August 1986)
5. Gutterman, Z., Pinkas, B., Reinman, T.: Analysis of the Linux Random Number Generator. In: IEEE Symposium on Security and Privacy (2006)
6. Dorrendorf, L., Gutterman, Z., Pinkas, B.: Cryptanalysis of the windows random number generator. In: ACM Conference on Computer and Communications Security (2007)

7. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. In: Proceedings of the 21st USENIX Security Symposium (August 2012)
8. Agyros, G., Kiayias, A.: I forgot your password: Randomness attacks against PHP applications. In: Proceedings of the 21st USENIX Security Symposium. USENIX Association (2012)
9. Kopf, G.: Non-Obvious Bugs by Example (2010), http://gregorkopf.de/slides_berlinsides_2010.pdf
10. Meyer, C., Somorovsky, J.: Why seeding with System.currentTimeMillis() is not a good idea (January 2012), http://armoredbarista.blogspot.de/2012/01/why-seeding-with-systemcurrenttimemilli.html
11. Lenstra, A., Hughes, J., Augier, M., Bos, J., Kleinjung, T., Wachter, C.: Public Keys. In: Advances in Cryptology CRYPTO 2012. LNCS. Springer, Heidelberg (2012)
12. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Cryptanalytic Attacks on Pseudorandom Number Generators. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 168–188. Springer, Heidelberg (1998)
13. Brown, R.G., Eddelbuettel, D., Bauer, D.: Dieharder: A Random Number Test Suite. Technical report, Duke University (2012)
14. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S., Bassham III, L.E.: A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications. Technical report, National Institute of Standards and Technology (NIST) (April 2010)
15. Lee, E.H., Lee, J.H., Park, I.H., Cho, K.R.: Implementation of high-speed SHA-1 architecture. IEICE Electronics Express (2009)
16. Zoltak, B.: VMPC One-Way Function and Stream Cipher. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 210–225. Springer, Heidelberg (2004)
17. Google Inc.: javax.crypto — Android Developers (July 2012)
18. McDonald, C., Hawkes, P., Pieprzyk, J.: Differential Path for SHA-1 with complexity O(2ˆ52). IACR Cryptology ePrint Archive (2009)
19. Wikipedia: Preimage attack — Wikipedia, The Free Encyclopedia (accessed August 24, 2012)
20. Handschuh, H., Knudsen, L.R., Robshaw, M.: Analysis of SHA-1 in Encryption Mode. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 70–83. Springer, Heidelberg (2001)
21. Tsunoo, Y., Saito, T., Kubo, H., Shigeri, M., Suzaki, T., Kawabata, T.: The Most Efficient Distinguishing Attack on VMPC and RC4A (2005)
22. Maximov, A.: Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness of RC4 Family of Stream Ciphers (2007) (corrected)
23. Li, S., Hu, Y., Zhao, Y., Wang, Y.: Improved cryptanalysis of the vmpc stream cipher. Journal of Computational Information Systems (2012)
24. Sverdlove, H., Brown, D., Cilley, J., Munro, K.: Orphan Android: Top Vulnerable Smartphones 2011. Technical report, Bit9, Inc. (November 2011)

Appendix

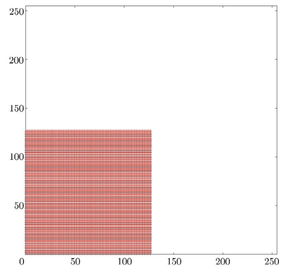


Fig. 9. Distribution of 2-tuples from Apache Harmonys integrated seeding facility

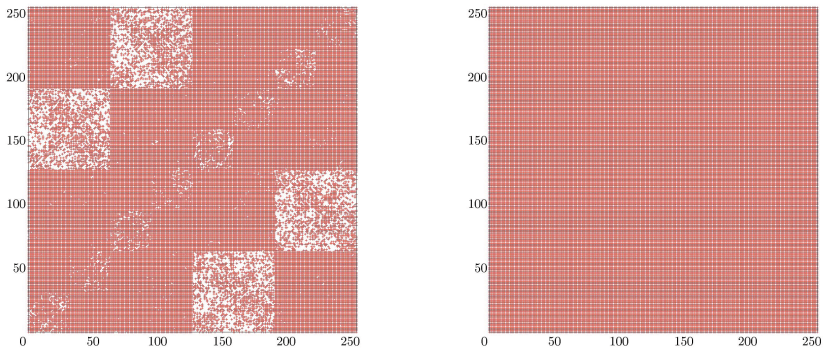


Fig. 10. Distribution of 2-tuples from `VMSecureRandom` under heavy (a) and normal (b) workload as implemented in GNU Classpath

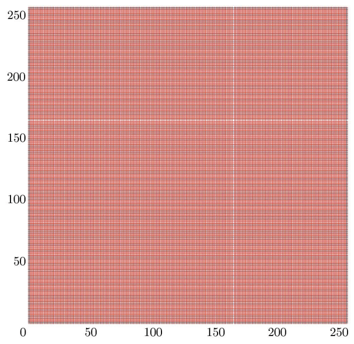


Fig. 11. Distribution of 2-tuples from the entropy collector in OpenJDK

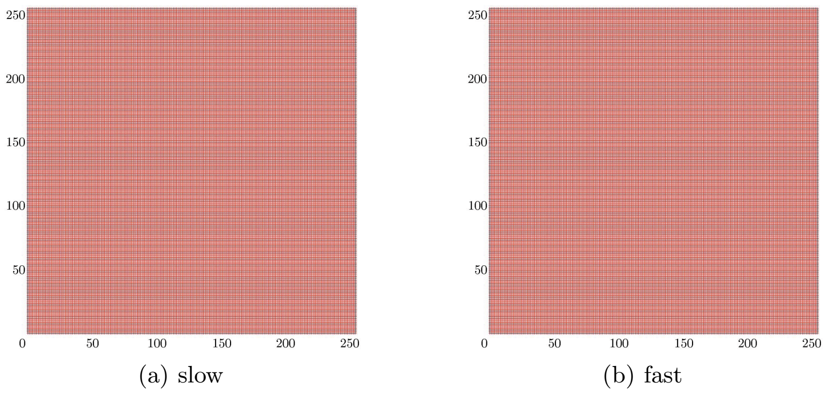


Fig. 12. Distribution of 2-tuples from the `ThreadedSeedGenerator` in Bouncy Castle's lightweight crypto library for Java running on "slow" (a) and "fast" (b)

Efficient Vector Implementations of AES-Based Designs: A Case Study and New Implemenations for Grøstl[⋆]

Severin Holzer-Graf, Thomas Krinninger, Martin Pernull, Martin Schläffer[1], Peter Schwabe[2], David Seywald, and Wolfgang Wieser

[1] IAIK, Graz University of Technology, Austria
martin.schlaeffer@iaik.tugraz.at
[2] Digital Security Group, Radboud University Nijmegen, The Netherlands
peter@cryptojedi.org

Abstract. In this paper we evaluate and improve different vector implementation techniques of AES-based designs. We analyze how well the T-table, bitsliced and bytesliced implementation techniques apply to the SHA-3 finalist Grøstl. We present a number of new Grøstl implementations that improve upon many previous results. For example, our fastest ARM NEON implementation of Grøstl is 40% faster than the previously fastest ARM implementation. We present the first Intel AVX2 implementations of Grøstl, which require 40% less instructions than previous implementations. Furthermore, we present ARM Cortex-M0 implementations of Grøstl that improve the speed by 55% or the memory requirements by 15%.

1 Introduction

Since the Advanced Encryption Standard (AES) was chosen by NIST in October 2000 [24], it has been used in innumerable applications. Apart from those applications, the components of AES or its design principles are also used as the basis for many new cryptographic algorithms. Especially the announcement of Intel to add an AES instruction (AES-NI) to its future processors [21] has caused an increasing amount of new AES-based designs. As a consequence, many AES-based designs and a few more AES-inspired designs have been submitted to the SHA-3 competition [25] initiated by NIST.

Building an AES-based design has several advantages. From a security point of view, AES-based designs can benefit from proofs against a large class of attacks.

[⋆] The work presented in this paper was carried out while Peter Schwabe was employed by Research Center for Information Technology Innovation, Academia Sinica, Taiwan. This work was funded in part by the National Science Council under Grant 100-2628-E-001-004-MY3, the European Commission through the ICT programme under contract ICT-SEC-2009-5-258754 (TAMPRES), and the Austrian Science Fund (FWF project P21936 and TRP 251-N23). Permanent ID of this document: de28943b229dbbf9d523fba01c2b028f. Date: Nov. 19, 2012.

Additionally, the design and security analysis of AES is kept particularly simple to provide security assurance within a short amount of time. As a consequence, the first single-key attack on 7 rounds of AES-128 [14] has been found before the AES competition was finished and the number of rounds (of non-marginal attacks) did not improve since then [13].

Aside from security analysis one of the most important criteria for the evaluation of cryptographic algorithms is software performance. Various implementation techniques have been proposed for AES and AES-based designs. The performance of AES-based designs and the best choice of implementation techniques highly depends on the target microarchitecture. If AES-NI is available, a design may be remarkably fast, while without AES-NI at can be quite slow. This is especially true for AES-based hash functions which consist of a large state with additional operations for mixing more than one AES state. This effect can be observed for many AES-based designs submitted to the SHA-3 competition.

In this work we focus on the three main software implementation techniques of AES-based designs: T-tables [12, Sect. 5.2], bitslicing [7] and byteslicing [1], which are discussed in detail in Section 3. We apply all techniques to the AES-based SHA-3 finalist `Grøstl` [16] and provide a number of new and improved results. We focus on implementations using vector-instruction sets.

In Section 4 we propose the first 256-bit vector implementation of `Grøstl` using the Intel AVX2 instructions [11]. Since no processor using AVX2 is available, we compare the number of instructions instead of performing a proper benchmark. The first AVX2 implementation is a bytesliced implementation of `Grøstl`-512 which improves the number of instructions by 40% compared to the AVX implementation. The second implementation uses the new AVX2 `vpgatherqq` instructions which allows to perform parallel table lookups.

In Section 5, we present the first ARM NEON [3, Chapter A7] implementations of `Grøstl` by applying all three techniques of Section 3. We show that the T-table and bitslicing approach result in equally fast implementations, while byteslicing is slower. However, we expect that byteslicing will outperform the other implementation techniques with the future AES instructions of ARMv8.

Finally, in Section 6 we show that vector implementations using byteslicing can even be used efficiently in low-memory environments. We present 32-bit bytesliced implementations of `Grøstl` which consume much less memory than T-table implementations at almost the same speed.

We have submitted all software presented in this paper to eBASH [4] or XBX [28] for public benchmarking and have put it into the public domain to maximize reusability of our results.

2 Description of `Grøstl`

The hash function `Grøstl` [15] was designed as a candidate for the SHA-3 competition [26]. For the final round of the competition, `Grøstl` was tweaked in order to increase its security margin. It is an iterated hash function with a compression function built from two distinct permutations P and Q, which are based on the same principles as the AES round transformation. In the following, we describe the components of the `Grøstl` hash function in more detail.

2.1 The Hash Function

`Grøstl` comes in two main variants, `Grøstl`-256 and `Grøstl`-512 which are used for different hash-value sizes of $n = 256$ and $n = 512$ bits. The hash function first pads the input message M and splits the message into blocks $M_1, M_2, \ldots, M_t$ of ℓ bits with $\ell = 512$ for `Grøstl`-256, and $\ell = 1024$ for `Grøstl`-512. The message blocks are processed via the compression function $f(H_{i-1}, M_i)$ and output transformation $\Omega(H_t)$. The size of the chaining value H_i is ℓ bits as well.

$$\begin{aligned} H_0 &= IV \\ H_i &= f(H_{i-1}, M_i) \quad \text{for } 1 \leq i \leq t \\ h &= \Omega(H_t). \end{aligned}$$

The compression function f is based on two ℓ-bit permutations P and Q (sometimes denoted by P_ℓ and Q_ℓ) and is defined as follows:

$$f(H_{i-1}, M_i) = P(H_{i-1} \oplus M_i) \oplus Q(M_i) \oplus H_{i-1}.$$

The output transformation Ω is applied to H_t to give the final hash value of size n, where $\text{trunc}_n(x)$ discards all but the least significant n bits of x:

$$\Omega(H_t) = \text{trunc}_n(P(H_t) \oplus H_t),$$

2.2 The Permutations

In each permutation, the four AES-like round transformations AddRoundConstant (AC), SubBytes (SB), ShiftBytes (SH), and MixBytes (MB) are applied to the state in the given order. The two permutations P and Q differ only the constants used in AC and SH. `Grøstl`-256 has 10 rounds and the 512-bit state of permutation P_{512} and Q_{512} is viewed as an 8×8 matrix of bytes. For `Grøstl`-512, 14 rounds are used and the 1024-bit state of the two permutations P_{1024} and Q_{1024} is viewed as an 8×16 matrix of bytes.

AddRoundConstant (AC) xors a round-dependent constant to one row of the state. The constant and the row is different for P and Q. Additionally, a round-independent constant 0xff is xored to every byte in Q. SubBytes (SB) applies the AES S-box to each byte of the state and the definition of the S-box can be found in [16]. ShiftBytes (SH) cyclically rotates the bytes of row r to the left by $\sigma[r]$ positions with different values for P and Q in `Grøstl`-256 and `Grøstl`-512. The rotation values are as follows:

$$\begin{aligned} \sigma &= \{0, 1, 2, 3, 4, 5, 6, 7\} \text{ for } P_{512}, \\ \sigma &= \{1, 3, 5, 7, 0, 2, 4, 6\} \text{ for } Q_{512}, \\ \sigma &= \{0, 1, 2, 3, 4, 5, 6, 11\} \text{ for } P_{1024}, \\ \sigma &= \{1, 3, 5, 11, 0, 2, 4, 6\} \text{ for } Q_{1024}. \end{aligned}$$

MixBytes (MB) is a linear diffusion layer, which multiplies each column A of the state with a constant, circulant 8×8 matrix M by computing $A \leftarrow M \cdot A$. The multiplication is performed in the finite field $GF(2^8)$ using the irreducible polynomial $x^8 \oplus x^4 \oplus x^3 \oplus x \oplus 1$ (0x11B). Since the multiplication by 2 can be carried out very efficiently using a single shift operation and a conditional xor, we will calculate all multiplications by combining multiplications by 2 and additions (xor). Moreover, optimized formulas for computing MixBytes have been published in [1]. Using these formulas, only 48 xors and 16 multiplications by 2 are needed to compute MixBytes:

$$M = \begin{bmatrix} 02 & 02 & 03 & 04 & 05 & 03 & 05 & 07 \\ 07 & 02 & 02 & 03 & 04 & 05 & 03 & 05 \\ 05 & 07 & 02 & 02 & 03 & 04 & 05 & 03 \\ 03 & 05 & 07 & 02 & 02 & 03 & 04 & 05 \\ 05 & 03 & 05 & 07 & 02 & 02 & 03 & 04 \\ 04 & 05 & 03 & 05 & 07 & 02 & 02 & 03 \\ 03 & 04 & 05 & 03 & 05 & 07 & 02 & 02 \\ 02 & 03 & 04 & 05 & 03 & 05 & 07 & 02 \end{bmatrix} \qquad \begin{aligned} b_i &= a_i + a_{i+1}, \\ a_i &= b_i + a_{i+6}, \\ a_i &= a_i + b_{i+2}, \\ b_i &= b_i + b_{i+3}, \\ b_i &= 02 \cdot b_i, \\ b_i &= b_i + a_{i+4}, \\ b_i &= 02 \cdot b_i, \\ a_i &= b_{i+3} + a_{i+4}. \end{aligned}$$

3 Implementation Methods for AES-Based Designs

In this section we give a high-level overview on common implementation techniques for AES-based designs using Grøstl-256 as an example. The main implementation techniques for AES-based designs are the T-table approach [12, Sect. 5.2], bitslicing [7], and byteslicing [1].

3.1 T-Table Approach

Daemen and Rijmen have presented a lookup-table-based approach for implementing AES on 32-bit processors in [12, Sect. 5.2]. This approach is known as the T-table approach and it can be generalized to other AES-based designs. The idea is to combine the SubBytes, MixColumns, and ShiftRows operations into table lookups. The size of the entries of the lookup tables matches the size of the state columns, 32 bits for AES and 64 bits for Grøstl.

Since many current and future small-scale 32-bit processors also provide 64-bit instructions (MMX, NEON), Grøstl can also be implemented efficiently on these platforms using the T-table approach. Even the 32-bit ARMv6 instruction set supports 64-bit loads which can be used for a T-table based implementation of Grøstl as shown in [27].

In T-table implementations, each column of the Grøstl state of is stored in a 64-bit registers. The AddRoundConstant transformation is computed through 8 xors on 64-bit registers. The SubBytes, ShiftBytes, and MixBytes operations are computed through 8 table lookups from tables $T_0, \ldots, T_7$ and 7 xors per column; for example, for column 0:

$$\begin{aligned} b_0 = & T_0(a_{00}) \oplus T_1(a_{11}) \oplus T_2(a_{22}) \oplus T_3(a_{33}) \oplus \\ & T_4(a_{44}) \oplus T_5(a_{55}) \oplus T_6(a_{66}) \oplus T_7(a_{77}). \end{aligned}$$

The values a_{ij} are bytes of the input state; the tables $T_0, \ldots, T_7$ contain 8-to-64-bit lookups of the S-box together with the 8 multipliers of MixBytes. For example, for the first table T_0 we get:

$$\begin{aligned} T_0(x) = & \mathsf{02} \cdot S(x) \parallel \mathsf{07} \cdot S(x) \parallel \mathsf{05} \cdot S(x) \parallel \mathsf{03} \cdot S(x) \parallel \\ & \mathsf{05} \cdot S(x) \parallel \mathsf{04} \cdot S(x) \parallel \mathsf{03} \cdot S(x) \parallel \mathsf{02} \cdot S(x) \end{aligned}$$

Extracting a single byte from a word can be implemented using a bit-shift and a logical and. Then, the computation of one column consists of only 8 table lookups, 8 xor (7 xor for MB, 1 xor for AC), 8 shift and 8 and instructions. On some platforms, single bytes a_{ij} can be extracted from 64-bit column words $a_j = [a_{00}, a_{10}, \ldots, a_{70}]^T$ at no cost. In this case, we can save (some of) the shift and and instructions.

3.2 Bytesliced Implementation

Another option to implement AES-based designs is a byte-wise parallel computation of columns [1]. This works especially well for large states and on platforms with large registers. In `Grøstl`, all round transformations except ShiftBytes and AddRoundConstant apply exactly the same computation to each column of the `Grøstl` state independently. Therefore, we can use a single-instruction-multiple-data (SIMD) approach to compute these identical operations on more than one column at the same time. If the state is stored in row ordering using w-bit registers, $w/8$ columns can be computed in parallel.

A requirement for this approach to be efficient is that all round transformations of `Grøstl` can be parallelized using only a few w-bit SIMD instructions. AddRoundConstant and MixBytes can be computed in parallel simply using basic ALU instructions. For ShiftBytes we need a byte-shuffling instruction or some mask-and-rotate instructions. The most difficult round transformation to parallelize is the 8-bit table lookup of SubBytes. However, using the Intel AES New Instructions extension (AES-NI) [21] or the vector-permute (vperm) approach by Hamburg [19], parallel AES S-box table lookups can be performed efficiently. Moreover, the fastest `Grøstl` implementation [4] is a bytesliced implementation using AES-NI.

In a bytesliced implementation, we need to use a row-ordering of the `Grøstl` state. However, the input bytes of the message are mapped to the `Grøstl` state in column-ordering. The column-ordering is a benefit for T-table based implementations but a drawback for bytesliced implementations. To reduce the state-transformation cost, the internal state is kept in row-ordering throughout the whole computation. Then, we only need to transform each input message block

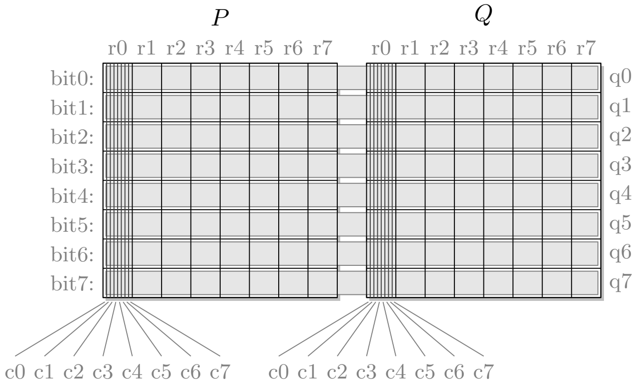


Fig. 1. Data organization of the bitsliced state for `Grøstl`-256 in 128-bit NEON `q` registers. Bits of equal rows (r0, . . . , r7) but different columns (c0, . . . , c7) are stored within the same byte.

and the hash-function output at the very end (the IV can be stored already in row-ordering). Transforming the input message from column-ordering to row-ordering corresponds to transposing the state matrix of the input message block.

3.3 Bitsliced Implementation

Bitslicing is an implementation technique proposed by Biham to improve the performance of DES [7]. Currently, the fastest software implementation of AES (without AES-NI) uses bitslicing [23]. Therefore, bitslicing is also a promising approach for other AES-based designs. Bitslicing works particularly well if the same operations can be performed many times in parallel. In AES, this is the case if multiple blocks are encrypted using a parallel mode of operation. Since the hash function `Grøstl` has a large state with many independent columns, bitslicing can be applied efficiently as well.

In general, bitslicing mimics hardware implementations in software. The data is transposed and, for example, a 32-bit value is stored in 32 registers, one bit per register. With this bitsliced representation of data we can simulate hardware gates with the corresponding bit-logical instructions. To use all m bits of a register, the same stream of operations is computed on m independent data streams in parallel. Registers of width m are used as vector registers with m 1-bit entries.

The AES S-boxes are computed using their algebraic structure (inversion in $\mathbb{F}_2$) as it is also done in efficient hardware implementations [8,10]. With minor modifications, the formulas underlying these hardware implementations are also used for bitslicing. More specifically, Käsper and Schwabe use 128 xor/and/or instructions and 35 mov (register to register) instructions in [23]. The mov instructions are required because in the SSE instruction set the output of an instruction has to overwrite one of the inputs. With 3-operand instructions (as provided by

AVX and NEON) and 16 registers, the AES S-box can be implemented using only 128 Boolean instructions. Although this is much slower than a table lookup for a single AES S-box computation, the high degree of parallelism (128 independent computations) often lets bitsliced implementations achieve higher speeds than table lookups.

The AES implementation by Käsper and Schwabe needs to process 8 blocks in parallel to achieve the required level of parallelism. In `Grøstl`-256 we can compute all 128 AES S-boxes of P and Q in parallel without the need for multiple blocks. However, ShiftBytes and MixBytes are more difficult to implement in this case. Note that for the S-box, it does not matter in which order the bytes are stored in registers. Therefore, we can choose a bitsliced state which fits the operations ShiftBytes and MixBytes best. By storing the `Grøstl`-256 bitsliced state as shown in Fig.1, we get an efficient implementation using 128-bit ARM NEON instructions (see Sect. 5.2).

4 Implementing `Grøstl` Using AVX2

The Intel AVX2 instruction set is an extension of the AVX instruction set and will be released by Intel for new processors in 2013 [22]. AVX2 provides a number of additions which can improve the efficiency of AES-based designs. AVX2 extends the functionality of integer-vector instructions to 256 bits. Furthermore, new gather instructions have been added, which provide new possibilities to implement parallel T-table lookups in AES-based designs.

Since no processors supporting AVX2 are available yet, all our AVX2 implementations have been tested using the Intel Software Development Emulator [20]. Because benchmarking of those implementations is not yet possible, we instead compare the number of instructions. Using AVX2, we show how to reduce the number of instructions for `Grøstl` by up to 40%, compared to previous AVX or AES-NI implementations [1]. Note that a similar comparison has been made by Gueron and Krasnov for their new AVX2 SHA-2 implementations using parallelized message schedules [18].

4.1 Byteslicing `Grøstl`-512 Using AVX2 and AES-NI Instructions

Using 256-bit registers of AVX2, P and Q of `Grøstl`-512 can be computed completely in parallel, except for the `aesenclast` instruction. Note that using AES-NI with SSE, P and Q have to be computed after each other. AVX2 also brings another major improvement compared to AVX. Many AVX instructions used by `Grøstl`-512 were only working on 128-bits (`vaesenclast`, `vpshufb`, `vpcmpgtb`, `vpaddb`). Especially `vpcmpgtb` and `vpaddb` are used very often in the multiplication by 2 of MixBytes. Hence, also many insertion and extraction instructions were needed to process the upper 128 bits of a 256-bit register separately.

Additionally, we have replaced the floating-point AVX instructions (`vxorps`, `vxorpd`) by their integer AVX2 instructions (`vpxor`). This avoids possible penalties caused by switching between integer and floating point domains [11].

To summarize our implementation, AddRoundConstant and ShiftBytes both can be fully parallelized and need only 8 instructions each. Note that `vpshufb` treats both 128-bit lanes separately. However, when storing P and Q in separate 128-bit lanes, we avoid all lane-switching penalties. In SubBytes, we need to use two 128-bit `vaesenclast` instructions for each row of the state. Together with the necessary `vinserti128` and `vextracti128` instructions, SubBytes of `Grøstl`-512 needs 32 instructions per round.

The most expensive round transformation is MixBytes. As shown in [1], MixBytes can be implemented using 48 xors and 16 multiplications by 2 (MUL2). Using the 256-bit `vpblendvb` instruction of AVX2, a single MUL2 computation can be implemented using only three 256-bit instructions. Together with 16 mov/xor instructions to load, store, copy, or clear temporary values, we get $48 + 3 \cdot 16 + 16 = 112$ instructions for MixBytes. Note that other variants to create the reduction mask in MUL2 are possible. For example, we may get a better throughput using `vpcmpgtb` with `vpand` instead of `vpblendvb` once AVX2 is available:

```
// ymm0 will be multiplied by 2
// ymm1 has to be all 0x1b
// ymm2 has to be all zero
// ymm3 will be lost
vpblendvb ymm3, ymm2, ymm1, ymm0
vpaddb ymm0, ymm0, ymm0
vpxor ymm0, ymm0, ymm3
```

```
// ymm0 will be multiplied by 2
// ymm1 has to be all 0x1b
// ymm2 has to be all zero
// ymm3 will be lost
vpcmpgtb ymm3, ymm2, ymm0
vpaddb ymm0, ymm0, ymm0
vandpd ymm3, ymm3, ymm1
vxorpd ymm0, ymm0, ymm3
```

Including an overhead of 5 instructions, we get a total of $8 + 8 + 32 + 112 = 165$ instructions for one round of `Grøstl`-512. Note that previously published AVX implementations need 271 instructions per round and the 128-bit AES-NI implementation needs 338 instructions [1]. Hence, using our new AVX2 implementation of `Grøstl`-512 we are able to save 40% of the instructions. Furthermore, using AVX2 instructions, we were also able to reduce the number of instructions to transpose the input message block into bytesliced representation.

4.2 Parallel T-Table Lookups for `Grøstl`-256 Using VPGATHERQQ

The new AVX2 instruction `vpgatherqq` allows to load four independent 64-bit values from memory into one 256-bit register. Using this instruction, we have implemented a fourfold parallel T-table implementation of `Grøstl`-256. We store the `Grøstl`-256 state column-wise and need two 256-bit registers for each of P and Q.

To perform the i-th T-table lookup for SubBytes and MixBytes, we first need a `vpshufb` instruction to extract the i-th byte of each 64-bit word. Note that we also use `vpshufb` to clear the unused bytes. To perform the actual lookups, `vpgatherqq` scales the extracted byte by a factor of 8 and adds the table address. The scaling takes into account that we actually perform 8-to-64 bit lookups. The table addresses are stored in 8 general-purpose registers.

The vpgatherqq instruction uses a mask to determine for which 64-bit words the lookup is performed. If the MSB of the corresponding 64-bit word is not set, this word is left unchanged. However, vpgatherqq clears the mask after each invocation and we have to restore the mask each time, e.g. using a vpcmpeqq instruction. Additionally, all registers used by vpgatherqq have to be distinct. Hence, we need 8 instructions for each of the 8 T-table lookups together with xoring the results. Since we can save two initial xors, we get 62 instructions for SubBytes and MixBytes per permutation and round. The code to compute the lookup of table i of one round is given below:

```
// SubBytes+MixBytes (Table i)
// byte extraction
vpshufb tmp0, ymm0, [EXTR+i*256]
vpshufb tmp1, ymm1, [EXTR+i*256]
// restore gather mask
vpcmpeqq mask, mask, mask
vpcmpeqq mask, mask, mask
```

```
// 4 parallel T-table lookups
// address of table i is in ri
vpgatherqq tmp2, [8*tmp0+ri], mask
vpgatherqq tmp3, [8*tmp1+ri], mask
// xor table lookup results
vpxor ymm2, ymm2, tmp2
vpxor ymm3, ymm3, tmp3
```

If table lookups can be performed in parallel, ShiftBytes together with the byte extractions can become the most costly operations in T-table implementations. Since most processors do not offer byte extraction instructions, a couple of ALU instructions are needed. In the case of AVX2, we can us a number of byte shuffles to compute ShiftBytes and to extract the bytes needed for the lookup. Since the vpshufb instruction can not move bytes across 128-bit lanes, we need additional vpermq instructions to cross lanes. To swap bytes between the two 256-bit registers storing the state, we use vpblendd which merges two vectors at 32-bit word granularity. To compute ShiftBytes, we need 8 instructions per permutation and round. The instructions for ShiftBytes are given below:

```
// pre-shuffle
vpshufb ymm0, ymm0, [SHIFT_P0]
vpshufb ymm1, ymm1, [SHIFT_P0]
// cross lanes
vpermq ymm2, ymm0, 0xd8
vpermq ymm3, ymm1, 0xd8
```

```
// combine registers
vpblendd ymm0, ymm2, ymm3, 0xaa
vpblendd ymm1, ymm3, ymm2, 0xaa
// final shuffle
vpshufb ymm0, ymm0, [SHIFT_P1]
vpshufb ymm1, ymm1, [SHIFT_P1]
```

Together with two instructions for AddRoundConstant we get in total, $(2 + 8 + 62) \cdot 2 = 144$ instructions per round of Grøstl-256. The currently fastest Grøstl-256 implementation uses AES-NI and needs 169 instructions per round. However, since it is still unknown how many cycles the vpgatherqq instruction will need to compute 4 lookups, we cannot conjecture any speed improvement.

5 ARM NEON Implementations of Grøstl

In this section, we present three new Grøstl implementations using ARM NEON instructions. We are focusing on the ARM Cortex A8 processor. The NEON vector instruction set is available also on other processors and the implementations

Table 1. Benchmark results of our NEON `Grøstl` implementations in cycles/byte for long messages. We used the SUPERCOP benchmarking suite [5] and performed the measurements using an ARM Cortex-A8 (Hercules eCafe).

hash function	T-table 5.1	bitsliced 5.2	vperm 5.3	arm32 [29]	arm11 [27]
`Grøstl`-256	45.8	48.5	92.0	76.9	99.4
`Grøstl`-512	67.0	-	-	103.2	-

presented here will work on them as well. However, the performance may be different from what we describe here. Each implementation corresponds to one of the implementation techniques described in Section 3. With the T-tables and the bitslicing approach, we get almost equally fast implementations running at around 46 cycles/byte. The bytesliced implementation is slower since we need to use the vperm approach to compute the AES S-box. However, once ARMv8 instructions with AES extensions are available [17], the bytesliced implementation will most likely be the fastest again. Detailed benchmarking results are given in Table 1.

The ARM NEON unit is a general-purpose SIMD (Single Instruction, Multiple Data) engine, which has its own registers and instruction set. It has 16 128-bit quadword registers (`q0-q15`) which can also be viewed (aliased) as 32 64-bit doubleword registers (`d0-d31`).

NEON on the Cortex A8 has limited dual issue capabilities. Instructions are divided between load/store/permute instructions and data processing (ALU) instructions. A data processing instruction can be dual issued with a load, store, or permute instruction. For multi-cycle instructions dual issue is only performed at the first and last cycle (see [2, Sect. 16.5.3]).

The ARMv7 core has 16 user-accessible general-purpose registers `r0-r15`, and one register which holds the current program status (CPSR). Register `r15` contains the program counter, `r14` the link register, and `r13` the stack pointer. In ARM mode, the link register and stack pointer can also be used as a general purpose register. One important property of the ARM processor is the built-in barrel shifter, which can shift and rotate the last operand of an ALU instruction at no cost. The ARM processor consists of two ALU units and one load/store unit.

Since ARM and NEON have separate instruction queues, ARM instructions also can be dual issued with NEON instructions. However, several restrictions apply. First, at most 2 instructions can be executed per cycle. Second, at most one load/store/permute can be performed per cycle. Third, moving data from NEON to ARM causes a penalty of at least 20 cycles, since the NEON unit lags behind the ARM unit.

5.1 T-Table Implementation of `Grøstl` Using NEON

Using NEON, one column of the `Grøstl` state can be stored in a 64-bit doubleword NEON register. This reduces the number of xors compared to a 32-bit

```
/* ROW 1 (SH+SB+MB) */          /* increase T-table address */
/* load state bytes */          /* compute lookup address */
/* T-table lookups  */          /* xor results */
ldrb r0, [%[P], #9 ];           add %[T], %[T], #2048;
ldrb r1, [%[P], #17];
ldrb r2, [%[P], #25];
ldrb r3, [%[P], #33];           add r0, %[T], r0, asl #3;
ldrb r4, [%[P], #41];           add r1, %[T], r1, asl #3;
ldrb r5, [%[P], #49];           add r2, %[T], r2, asl #3;
ldrb r6, [%[P], #57];           add r3, %[T], r3, asl #3;
ldrb r7, [%[P], #1 ];           add r4, %[T], r4, asl #3;
vld1.64 d8,  [r0, :64];         add r5, %[T], r5, asl #3;
vld1.64 d9,  [r1, :64];         add r6, %[T], r6, asl #3;
vld1.64 d10, [r2, :64];         add r7, %[T], r7, asl #3;
vld1.64 d11, [r3, :64];
vld1.64 d12, [r4, :64];         veor q0, q0, q4;
vld1.64 d13, [r5, :64];         veor q1, q1, q5;
vld1.64 d14, [r6, :64];         veor q2, q2, q6;
vld1.64 d15, [r7, :64];         veor q3, q3, q7;
```

Listing 1. The computation of row 1 of `Grøstl`-256 with ARM NEON using the T-table approach. Instructions are issued row-by-row. The address of the permutation P is stored in `P`, and the address of the current table is stored in `T`. The initial `add` instruction computes the (new) table address for row 1.

ARM implementation. Unfortunately, the indices used for the table lookups need to be stored in ARM registers. Hence, we compute one `Grøstl` round as follows: We load bytes of the state from memory into ARM registers and compute the table lookup address using ARM instructions. The table lookup itself and the xors are performed using NEON instructions. Finally, we store the result in memory using NEON stores.

Note that the 20-cycle penalty also occurs when transferring data from NEON to ARM through memory. We avoid this penalty by interleaving the computation of one round of P with a round of Q, since no data dependency between the two permutations exist. Hence, the ARM unit can continue to work on Q until the NEON unit is finished with computing and storing the result of one P round. Furthermore, we interleave the computation of 8 different columns of one permutation, to hide instruction latencies.

To avoid expensive byte extractions, we load single bytes of the state into the ARM registers using `ldrb`. We load bytes and compute the lookups row-by-row. This has the additional advantage, that we can use the same table address for 8 consecutive lookups. The address for the lookup is computed using `add` including a barrel shift to account for 8-to-64 bit table lookups. The actual T-table lookup is performed using `vld1.64`. We reduce the number of xors by using 128-bit `veor` instructions. The computation of one example row is given in Listing 1.

Equivalent code blocks are repeated 8 times for each row and round of P and Q. For AddRoundConstant we need four 128-bit loads and four `veor` instructions.

Additionally, we need four 128-bit stores at the end of each round. To summarize, the load/store instructions will be the bottleneck and we get a lower bound of $(16 \cdot 8 + 4 + 4) \cdot 10 \cdot 2/64 = 42.5$ cycles/byte. Using our new implementation, we get 45.8 cycles/byte on a Cortex-A8 processor.

5.2 Bitsliced Implementation of Grøstl-256 Using NEON

With the representation of the bitsliced state described in Section 3.3 we need 8 loads and 8 xors for AddRoundConstant and 128 ALU instructions for SubBytes [6]. The ShiftBytes operation rotates octets of bits (of a row) by different distances. To avoid expensive masking operations, it is most efficient to store these 8 bits within one byte. To rotate bits within each byte, we make use of the variable shift instruction `vshl.u8`. Note that the shift constants for shifting bits in bitsliced representation are the same as for bytes in standard representation.

The multiplication by 2 of MixBytes is rather cheap in bitsliced implementations and consists of only 3 xors [23]. What remains is to xor different rows of the non-bitsliced state to each other. Since we store bits of rows within bytes, we need to shuffle bytes of q-registers such that the corresponding bytes overlap and can get xored. Since crossing 64-bit lanes causes additional penalties, we store P in the lower and Q in the upper half of the 128-bit registers. Furthermore, we store the rows such that we can overlap corresponding bytes by rotating 8-byte blocks using the `vext.8` instruction. For example, we compute $b_i = a_i + a_{i+1}$ of bit 0 as follows:

```
vext.8 d24, d4, d4, #1;
vext.8 d25, d5, d5, #1;
veor q10, q2, q12;
```

Note that we can dual issue `vext.8` instructions with ALU instructions. In our implementation, we are able to interleave all `vext.8` instructions with the `veor` instructions of MixBytes, as well as the `vshl.u8` and `vorr` instructions of ShiftBytes. A sample excerpt of the implementation is given in Listing 2.

In the bitsliced representation of Grøstl-256, we have 128 ALU instructions for SubBytes, followed by 96 `vext.8` instructions which are interleaved with the ALU instructions of ShiftBytes and MixBytes. Hence, in the first part of one round, ALU instructions are the bottleneck, while in the second part, it is load, store, and permute instructions. Together with 8 loads and 8 `veor` required for the AddRoundConstant operation (interleaved), we get a lower bound of $(8 + 128 + 96) \cdot 10/64 = 36.25$ cycles/byte. In reality, our benchmark resulted in 48.5 cycles/byte, which is still about the same speed as the T-table implementation. We are continuing to investigate the reasons for the difference between the lower bound and our actual performance.

5.3 Bytesliced Vperm Implementation of Grøstl-256

The third option to implement Grøstl using NEON is a bytesliced implementation using vperm to compute the SubBytes transformation. On x86, the vperm

```
vext.8 d24, d4, d4,#1;
vext.8 d25, d5, d5,#1;
vext.8 d26, d6, d6,#1;  vshl.u8  q6, q14,  q4;  # bit6: shift left
vext.8 d27, d7, d7,#1;  veor    q10,  q2, q12;  # b2_i = a2_i + a2_{i+1}
vext.8 d24, d4, d4,#6;
vext.8 d25, d5, d5,#6;  veor    q11,  q3, q13;  # b3_i = a3_i + a3_{i+1}
vext.8 d26, d6, d6,#6;  vshl.u8  q1,  q9,  q4;  # bit1: shift left
vext.8 d27, d7, d7,#6;  veor     q2, q10, q12;  # a2_i = b2_i + a2_{i+6}
vext.8 d24,d20,d20,#2;
vext.8 d25,d21,d21,#2;  veor     q3, q11, q13;  # a3_i = b3_i + a3_{i+6}
vext.8 d26,d22,d22,#2;  vshl.u8 q14, q14,  q5;  # bit6: shift right
vext.8 d27,d23,d23,#2;  veor     q2,  q2, q12;  # a2_i = a2_i + b2_{i+2}
vext.8 d24,d20,d20,#3;
vext.8 d25,d21,d21,#3;  veor     q3,  q3, q13;  # a3_i = a3_i + b3_{i+2}
vext.8 d26,d22,d22,#3;  vshl.u8  q9,  q9,  q5;  # bit1: shift right
vext.8 d27,d23,d23,#3;  veor    q10, q10, q12;  # b2_i = b2_i + b2_{i+3}
vext.8  d4, d4, d4,#4;
vext.8  d5, d5, d5,#4;  veor    q11, q11, q13;  # b3_i = b3_i + b3_{i+3}
vext.8  d6, d6, d6,#4;  vorr     q6,  q6, q14;  # bit6: combine SHL+SHR
vext.8  d7, d7, d7,#4;  vorr     q1,  q1,  q9;  # bit1: combine SHL+SHR
```

Listing 2. Bitsliced implementation of Grøstl-256 using ARM NEON

implementation has a similar speed as the T-table implementation but using NEON, vector-permute or byte-shuffle instructions are more expensive.

In vperm implementations, each byte is split into nibbles which are then used as 4-bit indices to several 16-byte lookup tables. Four lookup tables are needed to compute the SubBytes transformation. Using the vperm approach, the S-box result can be multiplied by any factor without additional cost. This has been used by all previous vperm implementations of Grøstl [1,9]. However, if all multipliers are computed in advance, many temporary results are needed and also the optimized MixBytes formulas cannot be used. The computation of one row of SubBytes is shown in the listing below:

```
vand  q2, q0, q8
vshr.u8 q1, q0, #4
veor  q0, q2, q1
vtbl.8  d6, {d24-d25}, d2
vtbl.8  d7, {d24-d25}, d3
vtbl.8  d8, {d26-d27}, d4
vtbl.8  d9, {d26-d27}, d5
veor  q3, q3, q4
vtbl.8  d4, {d24-d25}, d0
vtbl.8  d5, {d24-d25}, d1
veor  q2, q2, q4
```

```
vtbl.8  d6, {d24-d25}, d4
vtbl.8  d7, {d24-d25}, d5
veor  q3, q3, q1
vtbl.8  d8, {d24-d25}, d6
vtbl.8  d9, {d24-d25}, d7
veor  q4, q4, q0
vtbl.8  d0, {d28-d29}, d6
vtbl.8  d1, {d28-d29}, d7
vtbl.8  d2, {d30-d31}, d8
vtbl.8  d3, {d30-d31}, d9
veor  q0, q1, q0
```

Note that we need two `vtbl.8` to shuffle 16 bytes and each instruction costs 2 cycles since we shuffle across 64-bit lanes. Hence, 16 AES S-box lookups need 22 instructions and we get a lower bound of 28 cycles (14 `vtbl.8` instructions with 2 cycles each). For the multplication by 2 (MUL2) we obtain 7 instructions and a lower bound of 8 cycles as follows:

```
// MUL2
vand  q1, q0, q8
vshr.u8 q0, q0, #4
vtbl.8  d2, {d20-d21}, d2
vtbl.8  d3, {d20-d21}, d3
vtbl.8  d0, {d22-d23}, d0
vtbl.8  d1, {d22-d23}, d1
veor  q0, q0, q1
```

AddRoundConstant needs 8 `veor` instructions and for ShiftBytes we can use 14 `vext` instructions to rotate bytes within 64-bit lanes. Additionally we have 19 load and stores of constants and temporary values. Using the optimized MixBytes formulas with 48 `veor` and 16 MUL2, we get a vperm NEON implementation for `Grøstl`-256 running at 92 cycles/byte.

6 Low-Memory Vector Implementation of `Grøstl`

On 32-bit platforms, the straight-forward way to implement `Grøstl` or other AES-based designs is the T-table approach. However, this method is not very suitable in low-memory environments since tables of a few kilobytes are needed. In this case, a bytesliced implementation can be the better choice. If the cache is small, it may even be faster than a T-table implementation. In this section, we give two short examples of bytesliced implementations using very small vectors.

6.1 32-Bit Bytesliced Implementations of `Grøstl`-256 for Cortex-M0

Since the ARM Cortex-M0 processor has only a small cache, memory access is rather expensive. Therefore, it turned out to be more efficient to compute MixBytes using a bytesliced implementation instead of using precomputed T-tables. In a 32-bit bytesliced implementation, we can compute 4 columns in parallel. Only for the SubBytes layer we need to extract bytes and perform single S-box lookups using a small table. Since the Cortex-M0 has only 8 registers we need to store the state in memory and process only a small fraction of the state at once.

However, load and store instructions on the Cortex-M0 are more expensive than ALU instructions. Therefore, we try to keep values in registers and perform as many computations on them as possible. The constants for AddRoundConstant are computed instead of storing them in memory. To compute the SubBytes layer, we load 32-bit values of the state into registers and extract single bytes using ALU instructions to perform the AES S-box lookup. For ShiftBytes we load two 32-bit values containing one row of the state and rotate and swap the values inside registers.

For MixBytes we use the optimized formulas with a minimal amount of 48 xor operations. Due to the small number of registers, we need a rather high number of temporary variables, in-register `mov` instructions and memory loads. Note that on the ARM Cortex-M0 platform, `push` and `pop` need only $N + 1$ cycles to push

Table 2. Benchmark results of the low-memory 32-bit vector implementation of `Grøstl`-256 on an ARM Cortex-M0 processor. We have measured the speed in cycles/byte for long messages and the memory requirements in bytes. The evaluation using $4 \cdot \mathrm{RAM} + \mathrm{ROM}$ has been proposed by XBX [28]. All implementations have been submitted to XBX.

	speed [cycles/byte]	RAM [Bytes]	ROM [Bytes]	$4 \cdot \mathrm{RAM} + \mathrm{ROM}$ [Bytes]
bytesliced (fast)	469	344	1948	3324
bytesliced (small)	801	304	1464	2680
T-table (2kB)	406	704	6952	9768
T-table (8kB)	383	508	12630	14662
sphlib	856	792	15184	18352
8bit-c	1443	632	2796	5324
armcryptolib	17496	400	1260	2860

or pop N registers to or from the stack, compared to $2 \cdot N$ instructions for loads and stores. By computing blocks of 8 32-bit values and using `push` and `pop`, we can significantly reduce the number of cycles needed to store temporary values.

Furthermore, we have implemented the multiplication by 2 completely within memory. We use an MSB mask `0x80808080` to generate the value which is conditionally xored to the bits determined by the irreducible polynomial `0x11b`. This method is similar to the multiplication by 2 used in the bitsliced implementation. The following listing shows the corresponding Thumb assembly code:

```
// MUL2
// r5: input, output
// r6: msbmask
// r1,r2: temporary
movs r1, r0
ands r1, r6
mvns r2, r6
ands r0, r2
lsls r0, #1
lsrs r1, #7
lsls r2, r1, #1
orrs r1, r2
lsls r2, r1, #3
orrs r1, r2
eors r0, r1
```

6.2 Results

We have implemented a fast and a small Thumb 32-bit bytesliced implementation for the Cortex-M0. The main difference is the use of macros and loop unrolling to speed up the computation at the cost of more memory. The results are given in Table 2. Additionally, we have implemented improved T-table implementations using 2kB or 8kB tables. We compare our results with previously published T-table implementations of `Grøstl`-256. The results show, that in low-memory environments, the bytesliced implementation consumes much less memory at only slightly decreased speed.

7 Conclusions

In this work we have analyzed three different implementation techniques for AES-based designs and presented various new and improved vector implementations of the SHA-3 finalist `Grøstl`. Depending on the target platform and the available instructions, a different implementation technique may be the fastest. For example, in the case of ARM NEON implementations we currently get the best result using the T-table approach, while the lower bound for the bitsliced implementation is better. Furthermore, once AES instructions of ARMv8 will be available, the bytesliced implementation technique will most likely outperform the others. The case is similar for many other platforms. We hope that our work will help implementers, but also designers of new AES-based cryptographic primitives to find the right balance of implementation characteristics.

References

1. Aoki, K., Roland, G., Sasaki, Y., Schläffer, M.: Byte Slicing Grøstl – Optimized Intel AES-NI and 8-bit Implementations of the SHA-3 Finalist Grøstl. In: Lopez, J., Samarati, P. (eds.) Proceedings of SECRYPT 2011, pp. 124–133. SciTePress (2011)
2. ARM Limited: Cortex-a8 technical reference manual, revision r3p2 (2010), `http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0344k/index.html`
3. ARM Limited: NEON (March 2011), `http://www.arm.com/products/processors/technologies/neon.php`
4. Bernstein, D.J., Lange, T.: eBASH: ECRYPT Benchmarking of All Submitted Hashes (January 2011), `http://bench.cr.yp.to/ebash.html`
5. Bernstein, D.J., Lange, T.: SUPERCOP (2012), `http://bench.cr.yp.to/supercop.html`, (accessed September 9, 2012)
6. Bernstein, D.J., Schwabe, P.: NEON crypto (2012), `http://cryptojedi.org/papers/#neoncrypto`
7. Biham, E.: A Fast New DES Implementation in Software. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 260–272. Springer, Heidelberg (1997), `http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/1997/CS/CS0891.pdf`
8. Boyar, J., Peralta, R.: A New Combinational Logic Minimization Technique with Applications to Cryptology. In: Festa, P. (ed.) SEA 2010. LNCS, vol. 6049, pp. 178–189. Springer, Heidelberg (2010)
9. Çalik, Ç.: Multi-stream and Constant-time SHA-3 Implementations. NIST hash function mailing list (December 2010), `http://www.metu.edu.tr/~ccalik/software.html#sha3`
10. 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)
11. Corp, I.: Intel advanced vector extensions programming reference (2011), `http://software.intel.com/file/36945`
12. Daemen, J., Rijmen, V.: AES Proposal: Rijndael. NIST AES Algorithm Submission (September 1999), `http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf`
13. Derbez, P., Fouque, P.A., Jean, J.: Improved Key Recovery Attacks on Reduced-Round AES. In: CRYPTO Rump Session (2012)
14. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.L.: Improved Cryptanalysis of Rijndael. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001)

15. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl – a SHA-3 candidate. Submission to NIST (2008), http://www.groestl.info (retrieved July 4, 2010)
16. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl – a SHA-3 candidate. Submission to NIST (Round 3) (2011), http://www.groestl.info (November 25, 2011)
17. Grisenthwaite, R.: Armv8 technology preview (2011), http://www.arm.com/files/downloads/ARMv8_Architecture.pdf
18. Gueron, S., Krasnov, V.: Simultaneous hashing of multiple messages. Cryptology ePrint Archive, Report 2012/371 (2012), http://eprint.iacr.org/2012/371
19. Hamburg, M.: Accelerating AES with Vector Permute Instructions. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 18–32. Springer, Heidelberg (2009), http://mikehamburg.com/papers/vector_aes/vector_aes.pdf
20. Intel: Intel software development emulator (2012), http://software.intel.com/en-us/articles/intel-software-development-emulator/
21. Intel Corporation: Intel Advanced Encryption Standard Instructions (AES-NI) (March 2011), http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni/
22. Intel (Mark Buxton): Haswell New Instruction Descriptions (June 2011), http://software.intel.com/en-us/blogs/2011/06/13/haswell-new-instruction-descriptions-now-available/
23. Käsper, E., Schwabe, P.: Faster and Timing-Attack Resistant AES-GCM. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 1–17. Springer, Heidelberg (2009)
24. National Institute of Standards and Technology: FIPS PUB 197, Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, U.S. Department of Commerce (November 2001)
25. National Institute of Standards and Technology: Cryptographic Hash Project (2007), http://www.nist.gov/hash-competition.
26. NIST: Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register 72(212), 62212–62220 (2007), http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf
27. Schwabe, P., Yang, B.-Y., Yang, S.-Y.: SHA-3 on ARM11 Processors. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 324–341. Springer, Heidelberg (2012), http://cryptojedi.org/papers/#sha3arm
28. Wenzel-Benner, C., Gräf, J.: XBX: eXternal Benchmarking eXtension for the SUPERCOP Crypto Benchmarking Framework (2012), https://xbx.das-labor.org/
29. Wieser, W.: Optimization of Grøstl for 32-bit ARM Processors. Bachelor's thesis, Graz University of Technology, Austria (2011)

Collisions for the WIDEA-8 Compression Function

Florian Mendel[1], Vincent Rijmen[2], Deniz Toz[2], and Kerem Varıcı[2]

[1] Graz University of Technology, IAIK, Austria
[2] KU Leuven, ESAT/COSIC and iMinds, Belgium

Abstract. WIDEA is a family of block ciphers inspired by the IDEA block cipher. The design uses n-parallel instances of IDEA with an improved key schedule to obtain block ciphers with larger block sizes. Moreover, the given design is suggested as the compression function for Davies-Meyer mode. In this paper, we discuss the security of the block cipher when used as a compression function. Inspired by the weak key attacks on IDEA, we take the advantage of slow diffusion mechanism of the key schedule and present free-start collisions for WIDEA-8 which is the specified version by designers. Our results are practical and we are able to obtain free-start collisions with a complexity of $2^{13.53}$.

Keywords: hash functions, cryptanalysis, WIDEA-8.

1 Introduction

Block ciphers are key components of cryptography. In the last two decades, parallel to the improvement in the technology, algorithms have evolved and more efficient and secure designs have been proposed. However, some algorithms managed to survive despite the extensive cryptanalysis. The block cipher IDEA [20], designed in 1990 by Lai and Massey, is a nice example of such an algorithm. There were various attacks on reduced round version of IDEA [4–7,10,13,17], but there was no known attack for full IDEA except the discovered weak key classes [9,11,16]. Recently, in EUROCRYPT'12, an attack which is better than exhaustive search with a factor of four was presented [19] for the full number of rounds.

WIDEA-n [18] is a family of block ciphers which aims to extend the block size of IDEA from 64-bit to $n \times 64$-bit by improving the performance results. In addition, the key schedule of IDEA is patched to make the design more secure against existing attacks and a non-linear shift register is used rather than rotations in the subkey generation.

Related Work. To the best of our knowledge, no external analysis of WIDEA-n has been published so far. But as a related work, security of the single-length and double-length hashing modes by using the IDEA as compression function is analyzed in [22]. The main idea of the analysis is to use the weak keys defined previously in [11] as an iterative characteristic for the compression function. Free-start collisions and semi-free-start collisions are obtained for the various schemes with practical complexity.

Our Contribution. In this paper, we study the security of WIDEA-n block cipher when it is used as a compression function in Davies-Meyer mode. We first use an approach similar to the one described above. However, due to the changes in key schedule, we only obtain free-start collisions up to seven rounds. Then, we modify the attack according to the new key schedule. We find new iterative characteristics with high probabilities such that the basic attack strategy is still applicable. At the end, we get free-start collisions for the full (8.5 round) WIDEA-8. Furthermore, we show that two free-start collisions can be combined to get a second order differential collision. These attacks are based on the utilization of weak keys and our results are given in Table 1.

Table 1. Results for WIDEA-8

target	rounds	time	attack type	sect.
comp. function	7	1	free-start collision	§4.2
comp. function	8.5 (full)	1	free-start near-collision	§4.3
comp. function	8.5 (full)	$2^{13.53}$	free-start collision	§4.3

Outline. This paper is organized as follows. In Section 2, we give a brief description of the WIDEA-n block cipher. In Section 3, we give an overview of the weak keys and describe the previous attacks on IDEA. Then, we present our observations and describe our attack procedure in Section 4. Finally, we conclude and summarize our results in Section 5.

2 Description of WIDEA

WIDEA-n [18] is a family of block ciphers, designed by Junod and Macchetti, presented at FSE 2009. The design uses n parallel applications of the IDEA [20] round function, strengthened with a mixing layer based on an MDS matrix.

In this paper, we focus on the version with $n = 8$ since it is introduced in the original paper with full specification. WIDEA-8 accepts a 512-bit plaintext $\mathbb{X} = X_0||X_1||X_2||X_3$ and a 1024-bit user key K which can be seen as an array of eight 128-bit words as inputs, and is composed of 8.5 rounds. Throughout this paper we will use the following notation.

$\odot$	Modular multiplication in $\mathbb{Z}^*_{2^{16}+1}$
$\boxplus$	Modular addition in $\mathbb{Z}_{2^{16}}$
$\oplus$	XOR
$\lll n$	left rotation of n positions
$X^{(i)}$	The input of the i-th round

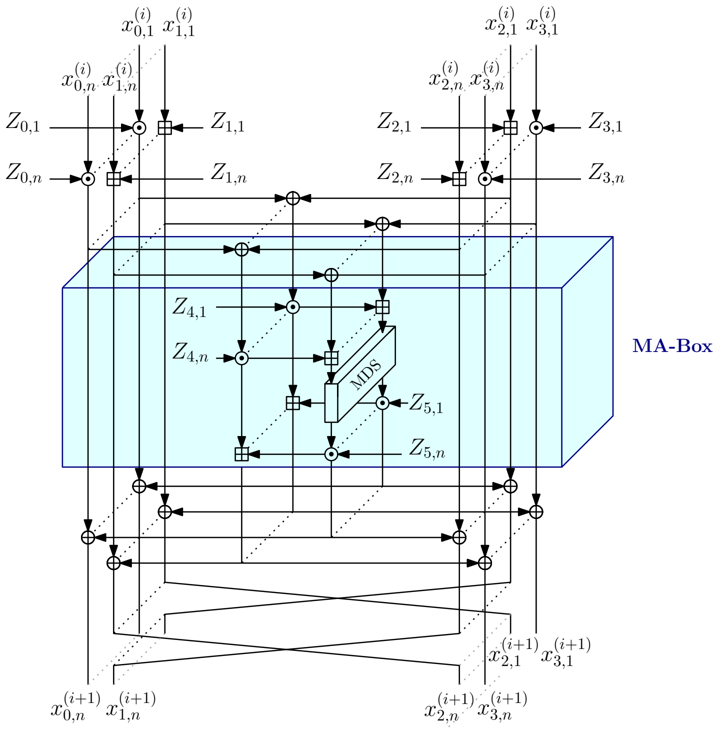

$x_{0,1}^{(i)}$ $x_{1,1}^{(i)}$ $x_{2,1}^{(i)}$ $x_{3,1}^{(i)}$
$x_{0,n}^{(i)}$ $x_{1,n}^{(i)}$ $x_{2,n}^{(i)}$ $x_{3,n}^{(i)}$
$Z_{0,1}$ $Z_{1,1}$ $Z_{2,1}$ $Z_{3,1}$
$Z_{0,n}$ $Z_{1,n}$ $Z_{2,n}$ $Z_{3,n}$
$Z_{4,1}$ $Z_{4,n}$ $Z_{5,1}$ $Z_{5,n}$
MDS
MA-Box
$x_{2,1}^{(i+1)}$ $x_{3,1}^{(i+1)}$
$x_{0,n}^{(i+1)}$ $x_{1,n}^{(i+1)}$ $x_{2,n}^{(i+1)}$ $x_{3,n}^{(i+1)}$

Fig. 1. The round function for WIDEA

Let $X^{(i)} = X_0^{(i)}||X_1^{(i)}||X_2^{(i)}||X_3^{(i)}$ with $X_j^{(i)} = x_{j,1}^{(i)}||x_{j,2}^{(i)}||\dots||x_{j,n}^{(i)}$ and $x_{j,k}^{(i)} \in \mathbb{Z}_{2^{16}}$. Then, the round function of WIDEA-8 is given in Figure 1.

In the design paper, WIDEA-8 is proposed as the compression function of Davies-Meyer mode [12] and the software performance is as good as the SHA-2 family [1] and the SHA-3 finalists [2,3,14,15,23].

2.1 Key Schedule

In the IDEA block cipher, the subkeys are generated by rotating the master key which causes some weaknesses in the design. Therefore, in the key schedule of WIDEA a non-linear feedback shift register is used to generate the subkeys. Let K_i, $0 \le i \le 7$ be the master key; C_i, $0 \le i \le 6$ be the chosen constants and Z_i, $0 \le i \le 51$ denote the subkeys that are used in the 8.5 rounds of WIDEA-n. Then, the key schedule is given as follows:

$$\begin{aligned}
Z_i &= K_i && 0 \le i \le 7 \\
Z_i &= ((((Z_{i-1} \oplus Z_{i-8}) \overset{16}{\boxplus} Z_{i-5}) \overset{16}{\lll} 5) \lll 24) \oplus C_{\frac{i}{8}-1} && 8 \le i \le 51, \quad 8|i \\
Z_i &= ((((Z_{i-1} \oplus Z_{i-8}) \overset{16}{\boxplus} Z_{i-5}) \overset{16}{\lll} 5) \lll 24) && 8 \le i \le 51, \quad 8 \nmid i
\end{aligned}$$

Here, each subkey Z_i has a size of $n \times 16$ bits and it can be split into the n 16-bit slices (i.e., $Z_i = z_{i,1}, \ldots, z_{i,n}$). Note that rotation by 5 bit positions is independently carried out on each slice $z_{i,j}$ for $1 \leq j \leq n$ and rotation by 24 bit positions is carried out globally for Z_i. For more detail, we refer to the specification of WIDEA [18].

3 Weak Keys for IDEA

If a key results in nonrandom behavior of the cipher it is called a weak key. For most of the ciphers, the weak keys are only a small fraction of the possible key space and hence the attacker tries to find the large set of weak classes to mount an attack. However, when a hash function is constructed from a block cipher, as in Davies-Meyer construction, the message takes the role of the key and the attacker has full control over it. Note that there exists various analysis on the weak keys for IDEA [9,11,16]. The ones related with our analysis are described below.

3.1 Weak Key Classes

Daemen et al. studied the classes of weak keys yielding characteristics with probability one in [11]. The nonlinear operations in the round function of IDEA are the modulo addition in $\mathbb{Z}_{2^{16}}$ and the modular multiplication in $\mathbb{Z}^*_{2^{16}+1}$ which provide good diffusion. Therefore, the basic idea is using a pair of inputs that differ only in the most significant bit (for each 16-bit word) and finding keys that will preserve this difference after modular addition and modular multiplication. To be more precise, let $\Delta = \texttt{0x8000}$, if the key value entering the modular operation, say Z_i, equals to ± 1 in $\mathbb{Z}_{2^{16}+1}$ then the difference after the modular operation again equals to Δ. This observation puts conditions on the subkey values used in the multiplication operation. But the rest of the subkeys can be chosen freely. Using this idea, the authors presented all possible characteristics for the round function of the IDEA block cipher with the conditions on the corresponding subkeys resulting in weak key classes of size up to 2^{35} out of 2^{128}.

3.2 Application to Hashing Modes

Recently, Wei et al. studied the security of the IDEA block cipher when it is used in various single-length or double-length hashing modes [22]. They were able to generate free-start collisions, semi-free-start collisions, pseudo-preimages or even hash collisions in practical complexity for most of these modes. Their attacks are based on the weak key classes mentioned above.

The simplest collision attack in the paper uses the null key (all zeros) for the encryption and each 16-bit plaintext word has the difference $(\Delta, \Delta, \Delta, \Delta)$. As stated in [11], these differences will behave linearly and lead to the same difference in the ciphertext with probability one. This difference is then canceled with the feed-forward operation resulting in a collision for the compression or hash function depending on the hashing mode.

4 Collision Attack on WIDEA-8

In this section, we study the security of the WIDEA-8 block cipher when it is used in Davies-Meyer construction. We first describe our basic attack strategy and show how the attacks on IDEA in hashing mode can be modified to attack WIDEA-8. We then present our results and give sample collisions.

4.1 Basic Attack Strategy

Although the round function of the WIDEA-8 block cipher is more complex than that of the IDEA block cipher due to the MDS operation, the previous attack strategy is still applicable if one is able to find an iterative characteristic that holds with high probability.

Observation 1. *The parallel instances of IDEA are only connected by the MDS matrix in the MA-box and hence if we can find a characteristic for one slice where the MA-box (see Figure 1) is never active, the attack is reduced to attacking only one slice instead of all eight.*

Based on this observation, if we have the input difference $(\Delta, \Delta, \Delta, \Delta)$ in one of the slices, then these differences in the 16-bit words cancel each other before the MDS operation and the input difference does not affect the other slices. As a result, we are able to obtain an iterative characteristic. A sample characteristic when there is a difference in the n-th slice is given in Figure 2.

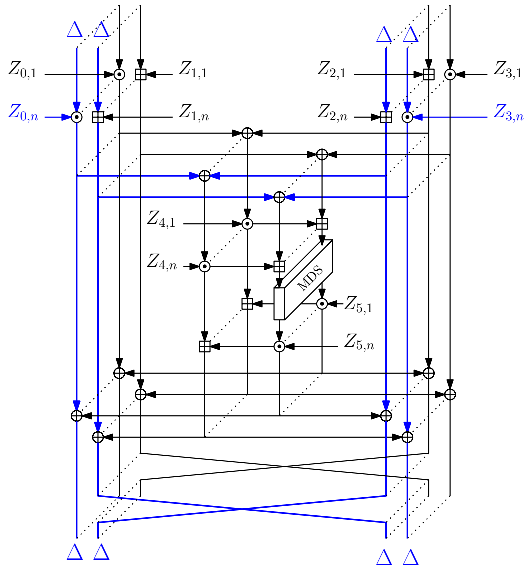


Fig. 2. A sample iterative characteristic for WIDEA-n

When an iterative characteristic is found, then due to the feed-forward operation in the Davies-Meyer mode, the output difference cancels out with the initial difference resulting in a collision for the compression function.

Unfortunately, it is not possible to use directly the null key as in the previous attacks. Whereas the key schedule of the IDEA block cipher uses only rotations and hence is linear, as described in Section 2.1 the key schedule of the WIDEA-8 block cipher consists of addition with a constant, modular addition, rotation and xor. Therefore, even though a null key is chosen after some rounds the subkeys will have nonzero values.

4.2 Collision Attack on 7 Rounds

In order to perform an attack, we want to find the longest iterative characteristic. Now, being familiar with the basic attack strategy, the existing challenge can be summarized as follows. To minimize the diffusion of the input differences, we need not only that the message words (which are used as keys) entering the multiplication have no difference but also the message words have to be zero. However, if we start with a null master key, the after three rounds all slices have nonzero values. Therefore, our aim is to find the maximum number of rounds such that all subkeys are zero at least in one of the slices. For this purpose we make use of the following observation.

Observation 2. *Given any eight consecutive subkeys* $\{Z_{i+1}, Z_{i+2}, \ldots, Z_{i+8}\}$*, it is possible to construct the whole set of subkeys.*

This allows us to start from the middle by setting the intermediate subkey values to zero and calculate forwards and backwards using the inverse key-schedule.

$$\begin{aligned} Z_i &= [(([(Z_{i+8} \oplus C_{\frac{i+1}{8}-1}) \ggg 24] \overset{16}{\ggg} 5) \overset{16}{\boxminus} Z_{i+3}] \oplus Z_{i+7}, \quad && 0 \le i \le 51, \quad 8 | i \\ Z_i &= [(([Z_{i+8} \ggg 24] \overset{16}{\ggg} 5) \overset{16}{\boxminus} Z_{i+3}] \oplus Z_{i+7}, && 0 \le i \le 51, \quad 8 \nmid i \end{aligned}$$

The best results we found are obtained by setting the subkeys $Z_{25} = Z_{26} = \ldots = Z_{32} = 0$. As it can be seen from Table 2, the subkey values entering the multiplication for WIDEA-8 equals to zero for the first slice up to Z_{42}. We want to note that we focus only on the subkey values $z_{i,j} = z_{(i+3),j}$ with $i = 0, 6, 12, ..., 48$, since if this values equal to zero then the multiplication operation behaves linear and the characteristic will hold.

As a result, we are able to pass seven rounds with probability one when there is the chosen difference $(\Delta, \Delta, \Delta, \Delta)$ where $\Delta =$ `0x8000` in the first slice. A collision example is given in Table 3.

4.3 Extending the Attack to Full WIDEA-8

In this section, we discuss how the attack on 7 rounds can be extended to full WIDEA-8. By ignoring the conditions on the message words in the first round

Table 2. Subkeys for WIDEA-8 when $Z_{25} = Z_{26} = \ldots = Z_{32} = 0$.

i	$z_{i,1}$	$z_{i,2}$	$z_{i,3}$	$z_{i,4}$	$z_{i,5}$	$z_{i,6}$	$z_{i,7}$	$z_{i,8}$
0	0000	E7FD	1444	6810	8B79	2822	47C8	0200
3	0000	E7FE	06F8	0000	0000	0000	0000	0000
6	0000	0001	F2E9	AFF7	0600	0000	0000	0000
9	0000	E7FF	FC58	0000	0000	0000	0000	0000
12	0000	F001	0520	0000	0000	0000	0000	0000
15	0000	0FFF	FAE0	0000	0000	0000	0000	0000
18	0000	F001	0520	0000	0000	0000	0000	0000
21	0000	0FFF	FAE0	0000	0000	0000	0000	0000
24	0000	F001	0520	0000	0000	0000	0000	0000
27	0000	0000	0000	0000	0000	0000	0000	0000
30	0000	0000	0000	0000	0000	0000	0000	0000
33	0000	0000	0000	0000	0000	0000	0000	0000
36	0000	0000	0000	0000	0000	0000	0000	0000
39	0000	0000	0000	0000	0000	0000	0000	0000
42	0000	0000	0000	0000	0015	E080	0B00	0000
45	4891	8264	0000	0000	0000	0000	00AF	5C00

we could find an input where most other subkeys are zero as required for the attack. In more detail, we set the subkeys $Z_{33} = Z_{34} = \ldots = Z_{40} = 0$ and find all other subkeys by computing forwards and backwards. The results are given in Table 4.

Using this as message input and the same iterative characteristic as for the attack on 7 rounds with $\Delta =$ 0x8000 we could find a free-start near-collision for full WIDEA-8. Note that due to the fact that $z_{0,8} =$ 0x42B4 another difference Δ' is needed in the chaining value $x_{0,8}$ to get the difference $\Delta =$ 0x8000 after the multiplication operation in the first round. The result is a free-start near-collision for full WIDEA-8 with complexity of 1 and only a difference in one 16-bit word at the output of the compression function after the application of the feed-forward. Moreover, we want to note that two free-start near-collisions can be combined to get a zero-sum (second-order differential collision [8,21]) for full WIDEA-8 with a complexity of only 2 compression function evaluations.

However, by using differences other than $\Delta =$ 0x8000 in the chaining input we can turn the free-start near-collision into a free-start collision for the compression function. Note that this will effect the probability of the attack due to the modular additions in WIDEA-8. Remember, we have $z_{0,8} =$ 0x42B4, $z_{2,8} =$ 0x7e49, $z_{8,8} =$ 0x2600 and $z_{49,8} =$ 0x5E00.

$$(x^{(1)}_{2,8} \overset{16}{\boxplus} z_{2,8}) \oplus ((x^{(1)}_{2,8} \oplus \Delta) \overset{16}{\boxplus} z_{2,8}) = \Delta \tag{1}$$

$$(x^{(2)}_{8,8} \overset{16}{\boxplus} z_{5,8}) \oplus ((x^{(2)}_{8,8} \oplus \Delta) \overset{16}{\boxplus} z_{8,8}) = \Delta \tag{2}$$

$$(x^{(8)}_{49,8} \overset{16}{\boxplus} z_{49,8}) \oplus ((x^{(8)}_{49,8} \oplus \Delta) \overset{16}{\boxplus} z_{49,8}) = \Delta \tag{3}$$

Table 3. A collision example for seven round of WIDEA-8 in hexadecimal

State							
5750	C1C3	1603	ADC5	2A12	DE9C	3547	1F24
5D9F	6856	D5A3	0188	808B	6D14	B0F4	58A9
6143	365B	BFDA	89DA	551B	F732	225A	FE0C
9BA8	C55E	AA2E	B4E1	3417	720D	22CF	8A28
State'							
D750	C1C3	1603	ADC5	2A12	DE9C	3547	1F24
DD9F	6856	D5A3	0188	808B	6D14	B0F4	58A9
E143	365B	BFDA	89DA	551B	F732	225A	FE0C
1BA8	C55E	AA2E	B4E1	3417	720D	22CF	8A28
$M_1 = M_2$							
0000	E7FD	1444	6810	8B79	2822	47C8	0200
0000	C7FF	67D5	2FE1	4839	0840	0000	0000
0000	D7FF	3F97	0009	931F	A917	0000	0000
0000	E7FE	06F8	0000	0000	0000	0000	0000
0000	E800	96CD	C81A	F500	0000	0000	0000
0000	D000	FC31	5803	3414	2F78	0000	0000
0000	0001	F2E9	AFF7	0600	0000	0000	0000
0000	E7FF	EC1A	5FEE	0C00	0000	0000	0000
WIDEA-8(State $, M_1) \oplus M_1 =$ WIDEA-8(State'$, M_2) \oplus M_2$							
D029	603E	368F	998F	7585	021C	492B	7DF0
BCB0	B142	15B0	B273	B503	1A6A	F410	9E4D
8F7F	BA4D	460E	8C9D	D2AD	0036	104B	43E6
E306	6246	6D73	3CDF	FD52	B205	267E	0720

We aim for differences with low Hamming weight. Moreover, we need a difference for which we can find a chaining input such that

$$(x_{0,8}^{(1)} \odot \texttt{0x42B4}) \oplus ((x_{0,8}^{(1)} \oplus \Delta) \odot \texttt{0x42B4}) = \Delta \qquad (4)$$

has a solution for some $x_{0,8}^{(1)} \in \mathbb{Z}_2^{16}$, and

$$(x_{i,8}^{(j)} \odot \texttt{0x0000}) \oplus ((x_{i,8}^{(j)} \oplus \Delta) \odot \texttt{0x0000}) = \Delta \qquad (5)$$

occurs with a high probability for all j where $i \geq 3$ and $i|3$.

Attack Procedure. We performed a search over all possible Δ values and found the best one (satisfying the conditions above) as $\Delta =$ 0x5820. We generate two input values $\mathbb{X}$ and $\mathbb{X}'$ as follows.

Table 4. Subkeys for WIDEA-8 when $Z_{33} = Z_{34} = \ldots = Z_{40} = 0$

i	$z_{i,1}$	$z_{i,2}$	$z_{i,3}$	$z_{i,4}$	$z_{i,5}$	$z_{i,6}$	$z_{i,7}$	$z_{i,8}$
0	3209	680D	AB9C	470D	6357	300A	C7C8	42B4
3	0000	BFFB	22E2	E13A	8FBC	B209	0800	0000
6	0000	2806	E120	46FD	F980	0000	0000	0000
9	0000	67FF	9C35	7EB3	3108	31C0	0400	0000
12	0000	F001	5517	790A	1080	0000	0000	0000
15	0000	27FA	E93A	9F2E	F600	0000	0000	0000
18	0000	D806	CECC	48A1	0B80	0000	0000	0000
21	0000	0FFF	72E5	CF97	FB00	0000	0000	0000
24	0000	F001	4521	1838	0680	0000	0000	0000
27	0000	0000	0000	0000	0000	0000	0000	0000
30	0000	0000	0000	0000	0000	0000	0000	0000
33	0000	0000	0000	0000	0000	0000	0000	0000
36	0000	0000	0000	0000	0000	0000	0000	0000
39	0000	0000	0000	0000	0000	0000	0000	0000
42	0000	0000	0000	0000	0000	0000	0000	0000
45	0000	0000	0000	0000	0000	0000	0000	0000
48	F7BA	0000	0000	0000	0000	0000	0000	0000
51	0000	0000	0003	C018	1C60	0100	0000	0000

– For $\mathbb{X}$, restrict $x_{0,8}^{(1)}$ to the values that satisfy Equation (4) and choose random values for the remaining 496 bits.
– Assign $\mathbb{X}' = \mathbb{X} \oplus (0^{112}||\Delta||0^{112}||\Delta||0^{112}||\Delta||0^{112}||\Delta)$

We then compute the output of full WIDEA-8 used in Davies-Meyer mode and check whether a collision occurs or not.

Complexity of the Attack. Equations (1)−(3) each have a success probability of 2^{-4} and Equation (5) is satisfied with probability $2^{-0.09}$. Therefore the complexity of the attack can be approximated as $(2^{-0.09})^{17} \cdot (2^{-4})^3 = 2^{-13.53}$ when xor is used in the feed-forward.

As a result, after generating 2^{14} initial values, one can find a free-start collisions for WIDEA-8 with full number of rounds. In practice, we found a collision after 2^8 trials which is better than our estimated complexity. The example is given in Table 5.

Table 5. A collision example for full WIDEA-8 in hexadecimal

`State`							
2C7A	0866	9F38	C148	3FB1	7BDA	0232	9054
E56C	8780	3E0D	96F3	6D1D	F028	907A	CA77
DDB6	AC09	77E4	D4C5	6715	E3CA	165A	3396
A835	DACB	CA5D	CC01	5270	F382	D7D7	7873
`State'`							
2C7A	0866	9F38	C148	3FB1	7BDA	0232	C874
E56C	8780	3E0D	96F3	6D1D	F028	907A	9257
DDB6	AC09	77E4	D4C5	6715	E3CA	165A	6BB6
A835	DACB	CA5D	CC01	5270	F382	D7D7	2053
$M_1 = M_2$							
3209	680D	AB9C	470D	6357	300A	C7C8	42B4
0000	5801	97F4	D0DA	0371	04E1	F400	0000
0000	7FF8	5C75	B946	131E	6335	CCF1	7E49
0000	BFFB	22E2	E13A	8FBC	B209	0800	0000
0000	4FF7	753E	2805	3E23	80E2	0C00	0000
0000	E7F9	7FC3	1818	DE12	EF37	C8F4	C1FF
0000	2806	E120	46FD	F980	0000	0000	0000
0000	4008	8FE9	8005	8A98	FF6E	F800	0000
$\texttt{WIDEA-8(State}\ , M_1) \oplus M_1 = \texttt{WIDEA-8(State'}, M_2) \oplus M_2$							
2C06	6743	87F8	775D	8AB8	5957	226C	4F0F
626F	934B	949F	7195	333A	997A	0D1E	9A32
3D2C	3435	3861	E7CB	2198	8074	94DA	2C26
2544	AD24	4881	E8DC	2344	015F	B015	6D81

5 Conclusion and Discussion

We have implemented the attacks and found free-start collisions for Davies-Meyer mode when it is initiated with WIDEA-8 as compression function. Since this single-length hashing mode is assumed to be secure in the ideal cipher model, it is not a good choice to use WIDEA-8 in this mode with the initially defined parameters. The easiest solution to fix this weakness seems like choosing the constant values more randomly. But still, the best way might be to use a new key schedule whose diffusion is better in the both forward and backward direction.

Acknowledgments. The work presented in this paper was done while Florian Mendel was with KU Leuven. The work has been supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II, and by the Research Fund KU Leuven, OT/08/027.

References

1. Secure Hash Standard. Federal Information Processing Standard 180-4. National Institute of Standards and Technology (2012), http://csrc.nist.gov/publications/fips/
2. Aumasson, J.P., Henzen, L., Meier, W., Phan, R.C.W.: SHA-3 proposal BLAKE. Submission to NIST (Round 3) (2010), http://131002.net/blake/blake.pdf
3. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak SHA-3 submission. Submission to NIST (Round 3) (2011), http://keccak.noekeon.org/Keccak-submission-3.pdf
4. Biham, E., Biryukov, A., Shamir, A.: Miss in the Middle Attacks on IDEA and Khufu. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 124–138. Springer, Heidelberg (1999)
5. Biham, E., Dunkelman, O., Keller, N.: New Cryptanalytic Results on IDEA. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 412–427. Springer, Heidelberg (2006)
6. Biham, E., Dunkelman, O., Keller, N.: A New Attack on 6-Round IDEA. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 211–224. Springer, Heidelberg (2007)
7. Biham, E., Dunkelman, O., Keller, N., Shamir, A.: New Data-Efficient Attacks on Reduced-Round IDEA. IACR Cryptology ePrint Archive 2011, 417 (2011)
8. 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)
9. Biryukov, A., Nakahara Jr, J., Preneel, B., Vandewalle, J.: New Weak-Key Classes of IDEA. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (eds.) ICICS 2002. LNCS, vol. 2513, pp. 315–326. Springer, Heidelberg (2002)
10. Borst, J., Knudsen, L.R., Rijmen, V.: Two Attacks on Reduced IDEA. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 1–13. Springer, Heidelberg (1997)
11. Daemen, J., Govaerts, R., Vandewalle, J.: Weak Keys for IDEA. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 224–231. Springer, Heidelberg (1994)
12. Davies, D., Price, W.: Digital signatures, an update. In: 5th International Conference on Computer Communication, pp. 845–849 (1994)
13. Demirci, H.: Square-like Attacks on Reduced Rounds of IDEA. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 147–159. Springer, Heidelberg (2003)
14. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein Hash Function Family. Submission to NIST (Round 3) (2010), http://www.skein-hash.info/sites/default/files/skein1.3.pdf
15. Gauravaram, P., Knudsen, L.R., Matusiewicz, K., Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: Grøstl – a SHA-3 candidate. Submission to NIST (Round 3) (2011), http://www.groestl.info/Groestl.pdf
16. Hawkes, P.: Differential-Linear Weak Key Classes of IDEA. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 112–126. Springer, Heidelberg (1998)
17. Junod, P.: New Attacks Against Reduced-Round Versions of IDEA. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 384–397. Springer, Heidelberg (2005)
18. Junod, P., Macchetti, M.: Revisiting the IDEA Philosophy. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 277–295. Springer, Heidelberg (2009)

19. Khovratovich, D., Leurent, G., Rechberger, C.: Narrow-Bicliques: Cryptanalysis of Full IDEA. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 392–410. Springer, Heidelberg (2012)
20. Lai, X., Massey, J.L.: A Proposal for a New Block Encryption Standard. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 389–404. Springer, Heidelberg (1991)
21. Lamberger, M., Mendel, F.: Higher-Order Differential Attack on Reduced SHA-256. Cryptology ePrint Archive, Report 2011/037 (2011), http://eprint.iacr.org/
22. Wei, L., Peyrin, T., Sokołowski, P., Ling, S., Pieprzyk, J., Wang, H.: On the (In)Security of IDEA in Various Hashing Modes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 163–179. Springer, Heidelberg (2012)
23. Wu, H.: The Hash Function JH. Submission to NIST (round 3) (2011), http://www3.ntu.edu.sg/home/wuhj/research/jh/jh_round3.pdf

Finding Collisions for Round-Reduced SM3

Florian Mendel, Tomislav Nad, and Martin Schläffer

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
`tomislav.nad@iaik.tugraz.at`

Abstract. In this work, we provide the first security analysis of reduced SM3 regarding its collision resistance. SM3 is a Chinese hash function standard published by the Chinese Commercial Cryptography Administration Office for the use of electronic authentication service systems and hence, might be used in several cryptographic applications in China. So far only few results have been published for the SM3 hash function. Since the design of SM3 is very similar to the MD4 family of hash functions and in particular to SHA-2, a revaluation of the security of SM3 regarding collision resistance is important taking into account recent advances in the cryptanalysis of SHA-2. In this paper, we extend the methods used in the recent collision attacks on SHA-2 and show how the techniques can be effectively applied to SM3. Our results are a collision attack on the hash function for 20 out of 64 steps and a free-start collision attack for 24 steps of SM3, both with practical complexity.

Keywords: hash functions, cryptanalysis, collisions, free-start collisions.

1 Introduction

A cryptographic hash function H maps a message M of arbitrary length to a fixed-length hash value h. Informally, a cryptographic hash function has to fulfill the following security requirements:

- *Collision resistance:* it is practically infeasible to find two messages M and M^*, with $M^* \neq M$, such that $H(M) = H(M^*)$.
- *Second preimage resistance:* for a given message M, it is practically infeasible to find a second message $M^* \neq M$ such that $H(M) = H(M^*)$.
- *Preimage resistance:* for a given hash value h, it is practically infeasible to find a message M such that $H(M) = h$.

The resistance of a hash function to collision and (second) preimage attacks depends in the first place on the length n of the hash value. Regardless of how a hash function is designed, an adversary will always be able to find preimages or second preimages after trying out about 2^n different messages. Finding collisions requires a much smaller number of trials: about $2^{n/2}$ due to the birthday paradox. If the internal structure of a particular hash function allows collisions or (second) preimages to be found more efficiently than what could be expected based on its hash length, then the function is considered to be broken. For a formal treatment of the security properties of cryptographic hash functions we refer to [10,11].

Most cryptanalytic results on hash functions focus on collision attacks. In the last years collisions have been shown for many commonly used hash functions. In particular, the collision attacks of Wang et al. [13,14] on MD5 and SHA-1 have convinced many cryptographers that these widely deployed hash functions can no longer be considered secure. As a consequence, NIST proposed the transition from SHA-1 to the SHA-2 family and many companies and organization are migrating to SHA-2. Furthermore, researchers are evaluating alternative hash functions in the SHA-3 initiative organized by NIST [9] to find a new hash function standard.

In this work, we analyze the Chinese hash function standard SM3. SM3 was designed by Wang et al. [1] and is published by the Chinese Commercial Cryptography Administration Office for the use of electronic authentication service systems. The amount of cryptanalytic results on SM3 is low compared to other hash function standards. Kircanski et al. [4] presented a distinguisher for the compression function of SM3 up to 35 steps with complexity $2^{117.1}$. Moreover, Zou et al. [15] presented a preimage attack on 30 steps of SM3 with complexity of 2^{249}.

The design of SM3 is very similar to the MD4 family in particular SHA-2. New collision attacks on SHA-2 and similar hash functions have been shown [2,5–8] recently. The attacks have in common that they are all of practical complexity and are based on automatic search algorithms to find complex differential characteristics.

In this paper, we develop the methods by Mendel et al. for SHA-256 [7] further and apply them on SM3. We show how the technique can be effectively applied to SM3. Furthermore, we present a collision for 20 steps and a free-start collision for 24 steps of SM3. These are the first collision attacks on the step-reduced SM3 hash and compression function.

The remainder of this paper is structured as follows. A description of the hash function is given in Section 2. In Section 3 we describe the basic attack strategy. In Section 4 we show how we can find differential characteristics and conforming message pairs for SM3. Finally, we present a collision and free-start-collision for step-reduced SM3 in Section 5 and conclude in Section 6.

2 Description of SM3

SM3 is an iterated hash function that processes 512-bit input message blocks and produces a 256-bit hash value. In the following, we briefly describe the hash function. It basically consists of two parts: the message expansion and the state update transformation. A detailed description of the hash function is given in [1].

2.1 Message Expansion

The message expansion of SM3 is linear in $GF(2)$. It splits the 512-bit message block into 16 words M_i, $i = 0, \dots, 15$, and expands them into 68 expanded message words W_i and 64 expanded message words W'_i as follows:

$$W_i = \begin{cases} M_i & 0 \le i < 16 \\ \sigma_0(W_{i-16} \oplus W_{i-9} \oplus W_{i-3} \lll 15) \oplus W_{i-13} \lll 7 \oplus W_{i-6} & 16 \le i < 68 \end{cases}$$

and

$$W'_i = W_i \oplus W_{i+4} \quad 0 \le i < 64 \,.$$

The functions $\sigma_0(X)$ is given by

$$\sigma_0(X) = X \oplus (X \lll 15) \oplus (X \lll 23)$$

2.2 State Update Transformation

The state update transformation starts from a (fixed) initial value IV of eight 32-bit words and updates them in 64 steps. In each step the 32-bit words W_i and W'_i are used to update the eight state variables $A_{i-1}, B_{i-1}, \ldots, H_{i-1}$.

$$\begin{aligned} T_1 &= (A_{i-1} \lll 12 + E_{i-1} + K_i) \lll 7 \\ T_2 &= H_{i-1} + f_0(E_{i-1}, F_{i-1}, G_{i-1}) + T_1 + W_i \\ A_i &= D_{i-1} + f_1(A_{i-1}, B_{i-1}, C_{i-1}) + (T_1 \oplus A_{i-1} \lll 12) + W'_i \\ E_i &= \Sigma_0(T_2) \\ B_i &= A_{i-1} \\ C_i &= B_{i-1} \lll 9 \\ D_i &= C_{i-1} \\ F_i &= E_{i-1} \\ G_i &= F_{i-1} \lll 19 \\ H_i &= G_{i-1} \end{aligned} \tag{1}$$

For the definition of the step constants K_i we refer to [1]. The bitwise Boolean functions f_0 and f_1 are different for each step. In the first 16 steps f_{XOR} is used for both f_0 and f_1. After step 16 f_0 is f_{IF} and f_1 is f_{MAJ}.

$$\begin{aligned} f_{XOR}(X, Y, Z) &= X \oplus Y \oplus Z \\ f_{IF}(X, Y, Z) &= XY \oplus XZ \oplus Z \\ f_{MAJ}(X, Y, Z) &= XY \oplus YZ \oplus XZ \end{aligned} \tag{2}$$

The linear function Σ_0 is defined as follows:

$$\Sigma_0(X) = X \oplus (X \lll 9) \oplus (X \lll 17) \tag{3}$$

After the last step of the state update transformation, the initial values are XORed to the output values of the last four steps (Davies-Meyer construction). The result is the final hash value or the initial value for the next message block.

3 Basic Attack Strategy

In the following, we first give a brief overview of the attack strategy used in the recent collision attacks on the MD4-family of hash functions [12, 14]. The high-level strategy can be summarized as follows:

1. Find a characteristic for the hash function that holds with high probability.
2. Use message modification techniques to fulfill conditions imposed by the characteristic. This increases the probability of the characteristic.
3. Use random trials to find values for the remaining free message bits such that the message follows the characteristic.

The most difficult and important part of the attack is to find a good differential characteristic. The second important part of the attack is to find conforming inputs for the differential characteristic. For both parts we used the technique of the recent attack on SHA-2 [7].

4 Automatic Search Tool

The collision attack on SHA-2 [7] can be summarized as follows:

1. Determine a starting point for the search which results in an attack on a large number of steps. The resulting start characteristic should span over few steps and only some message words should contain differences.
2. Use an automated search tool to find a differential characteristic for the unrestricted intermediate steps including the message expansion.
3. Continue the search to find a conforming message pair. If no message pair can be found, adjust the differential characteristic accordingly.

Due to the linearity of the message expansion, finding a good starting point is rather simple. The most difficult and important part of the attack is to find a good differential characteristic. Due to the increased complexity of SM3 compared to hash functions like SHA-1 and MD5, finding good differential characteristics by hand is almost impossible. Therefore, we use an automatic tool to find complex nonlinear differential characteristics. The tool is also used for solving nonlinear equations involving conditions on state words and free message bits, i.e. to find confirming message pairs. The tool is based on the approach of Mendel et al. [7] to find nonlinear differential characteristics and conforming message pairs for SHA-2.

4.1 Generalized Conditions

The tool and search algorithm is based on the concept of generalized conditions introduced in [2]. Generalized conditions are inspired by signed-bit differences and take all 16 possible conditions on a pair of bits into account. Table 1 lists all these possible conditions and introduces the notation for the various cases.

Using these generalized conditions and propagating them in a bitsliced manner, we can construct complex differential characteristics in an efficient way. The basic idea of the search algorithm is to randomly pick a bit from a set of bit positions with predefined conditions, impose a more restricted condition and compute how this new condition propagates. This is repeated until an inconsistency is found or all unrestricted bits from the set are eliminated. Note that this

Table 1. Notation for possible generalized conditions on a pair of bits [2]

(X_i, X_i^*)	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$	(X_i, X_i^*)	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
?	✓	✓	✓	✓	3	✓	✓	-	-
-	✓	-	-	✓	5	✓	-	✓	-
x	-	✓	✓	-	7	✓	✓	✓	-
0	✓	-	-	-	A	-	✓	-	✓
u	-	✓	-	-	B	✓	✓	-	✓
n	-	-	✓	-	C	-	-	✓	✓
1	-	-	-	✓	D	✓	-	✓	✓
#	-	-	-	-	E	-	✓	✓	✓

general approach can be used for both, finding differential characteristics and conforming message pairs. There are three important aspects of the automated tool: using a good starting point, using an efficient condition propagation and using a sophisticated search strategy. We discuss each aspect in the following sections.

4.2 Defining a Starting Point

Similar to the attack on SHA-256 [7] we construct a local collision with differences in a few steps which results in a attack on a large number of steps. Since the message expansion of SM3 is linear finding a starting point is easier than for SHA-256.

To find a good starting point for SM3, a system of linear equations representing the message expansion is constructed. Afterwards, linear constraints are added and the system is solved. In that way several good starting points for up to 24 steps have been found. The starting points for 20 and 24 steps are given in the Appendix in Table 5 and Table 7.

Note that unlike in most hash function attacks so far, the non-linear part for 24 steps is placed at the end instead of the beginning. This has several reasons. First of all the differential characteristic for the message expansion is more sparse. Furthermore, after step 16 the Boolean function IF and MAJ instead of XOR are used. This again results in more sparse characteristic. In general the more sparse a characteristic the easier it is to find conforming message pairs.

4.3 Efficient Condition Propagation

The efficient propagation of new conditions is crucial for the performance of the algorithm, since it is the most often needed operation in the search algorithm. Due to the nature of the search algorithm where changes to the characteristic (using generalized conditions) are done on bit-level, we perform the propagation of conditions also on bit-level. At the beginning of the search every bit has at least one of the 16 generalized conditions (see Table 1). During the search we impose conditions on specific bits. These bits are inputs or outputs of functions. If a bit in the output is changed then all bits which are used to determine this output bit are updated. We call such a set of bits a bit-slice. If the changed bit is an input of a function then all other bits of the corresponding bit-slice are updated. The following example illustrates this process.

Example 1 (Condition Propagation). Let $f : \mathbb{F}_{32}^3 \to \mathbb{F}_{32}$ be the Boolean IF function operating on 32-bit words and defined as follows:

$$f(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) = o.$$

Then the output bit o_i depends on the bits $\{x_i, y_i, z_i\}$ and $\{x_i, y_i, z_i, o_i\}$ forms a bit-slice. If the generalized condition ∇x_i changes then the conditions of the set $\{\nabla x_i, \nabla y_i, \nabla z_i, \nabla o_i\}$ are updated.

In our approach the update process is done exhaustively by computing all possible conditions of a bit-slice. This seems at first to be inefficient but we are using two techniques to significantly speed up the process. The first one splits the state update in smaller functions and the second one utilizes a cache. However, the update process for a modular addition is done in a slightly different way. The bit-slices of a modular addition contain also input carry and output carry. Hence, the bit-slices are connected through the carry bits. If the condition for a carry bit changes, then the connected bit-slice is updated as well. Furthermore, the whole update process is iterative and updates bits until conditions do not change any more.

4.4 Increasing the Propagation Performance

In the state update transformation of SM3, only two state variables are updated in each step, namely A_i and E_i. Therefore, we can redefine the state update such that only these two variables are involved. In this case, we get the following mapping between the original and new state variables:

A_i	B_i	C_i	D_i	E_i	F_i	G_i	H_i
A_i	A_{i-1}	A_{i-2}	A_{i-3}	E_i	E_{i-1}	E_{i-2}	E_{i-3}

Hence, only two state variables need to be stored. Furthermore, the complexity of propagating generalized conditions increases exponentially with the number of input bits and additions. Similar as in SHA-2 the number of input bits in the update of A_i,E_i and W_i is high. To reduce the computational complexity of the propagation, we further split the update of W_i, E_i and A_i into sub-steps. The decision where to split need to be done carefully. If too many sub-steps are introduced we are losing too much information resulting in a worse propagation and late detection of contradictions. If too few sub-steps are introduced the performance of the propagation is too slow. Therefore, we found the following separation of the SM3 state update which leads to a good performance/propagation ratio:

$$\begin{aligned}
S_i &= W_{i-16} \oplus W_{i-9} \oplus W_{i-3} \lll 15, \\
P_i &= S_i \oplus S_i \lll 15 \oplus S_i \lll 23, \\
W_i &= P_i \oplus W_{i-13} \lll 7 \oplus W_{i-6}, \\
W'_i &= W_i \oplus W_{i+4}, \\
L_i &= A_{i-1} \lll 12 + E_{i-1} + K_i \lll 7, \\
F_i &= A_{i-1} \oplus A_{i-2} \oplus A_{i-3} \lll 9, \\
A_i &= F_i + W'_i + A_{i-4} \lll 9 + (L_i \lll 7 \oplus A_{i-1} \lll 12), \\
G_i &= E_{i-1} \oplus E_{i-2} \oplus E_{i-3} \lll 19, \\
R_i &= E_{i-4} \lll 19 + L_i \lll 7 + W_i + G_i, \\
E_i &= R_i \oplus R_i \lll 9 \oplus R_i \lll 17.
\end{aligned}$$

By carefully analyzing the state update and message expansion we have split up the computations such that one step does not have more than 5 inputs. Using this representation of SM3 we can use a cache during the propagation efficiently. Furthermore, for those steps with only three inputs we are able to compute all possibilities beforehand, changing the propagation of this steps to a simple table lookup.

4.5 Search Strategy

To reduce the complexity of the system and eventually find a solution, random additional conditions are introduced. In other words, some variables are guessed. Even the most efficient method to propagate information may not result in a solution if we make poor guesses. We need a guessing strategy, which can efficiently use the new information generated by the propagation of information introduced by previous guesses. The goal of a good guessing strategy is to discard invalid solutions and to find a valid solution as soon as possible. The guessing strategy depends in first place on the shape of the equations and the storeable information propagated. Furthermore, external knowledge of the structure of the attacked cryptographic system can help to improve the guessing strategy.

Our guessing strategy is similar to the one used by Mendel et al. in the attack on SHA-256 reduced to 32 steps [7]. However, there are some small but important modifications. In our approach for SM3 we further refine the search strategy for SM3 by considering specific output words for guessing. As in the attack of [7], our search strategy consists of several stages and each stage can basically be divided into three parts: decision, deduction and backtracking. Note that the same separation is done in many other fields, like SAT solvers [3]. In the decision part, we decide which bit is chosen and which constraints are imposed at its position. In the deduction part we compute the propagation of the new information and check for consistence. In the case of an inconsistency we need to backtrack and undo previous decisions, which is the third and last part.

Let U be a set of generalized conditions. Repeat the following until U is empty:

Decision

1. Pick according to some heuristic (or randomly) a bit in U.
2. Impose new constraints on this bit according to Table 2.

Deduction

3. Propagate the Information to the other variables and equations as described in Section 4.3.
4. If an inconsistency is detected start backtracking, else continue with step 1.

Backtracking

5. Try the second choice for the decision bit.
6. If this still results in an inconsistency mark this bit as critical.
7. Jump back until the critical bit can be resolved.
8. Continue with step 1.

Note that in each stage different bits are chosen (guessed). In total we have two stages which can be summarized as follows.

Stage 1: In the first stage we search for a consistent differential characteristic in the state words. Therefore, we add all unconstrained bits of A_i andE_i that are `?` or `x` to the set U. Furthermore, we add bits of L_i as well to U. Experience has shown that guessing the output of modular additions first provides a significant speed up. Due to the additional freedom added by the carry bits, the propagation of conditions is slow towards the output of modular additions if these bits are not included in U.

Stage 2: In the second stage we search for conforming inputs. Therefore, we pick decision bits with many two-bit conditions, since this ensures that bits which influence a lot of other bits are guessed first. Furthermore, many other bits propagate by defining the value of a single bit. Hence, this way inconsistent characteristics are discarded earlier and valid solutions are found faster. The concept of two-bit conditions was introduced in [7] .

Note that we dynamically switch between the two stages. Additionally, we restart the search from scratch after a certain amount of inconsistencies to terminate branches which appear to be stuck because of exploring a search space far from a solution.

Table 2. Decision rules of our guessing strategy with $r \in \{0, 1\}$ a random value

Decision bit	r	Choice 1	Choice 2
`?`	1,0	`-`	`x`
`x`	0	`u`	`n`
	1	`n`	`u`
`-`	0	`0`	`1`
	1	`1`	`0`

5 Results for Reduced SM3

To find collisions for reduced SM3 we apply the techniques described in Section 4. We first construct a differential characteristic with low Hamming weight in the message expansion which functions as starting point for the automatic search algorithm. Next the search algorithm is applied. Running on a cluster with 72 nodes, the algorithm finds a differential characteristic in less than 1 hour. Afterwards, we continue the search for a conforming message pair which can be found in several seconds.

5.1 Collision Attack

Using the starting point given in Table 5 and our automatic search algorithm described in Section 4, we are able to construct collisions for up to 20 steps of SM3. The differential characteristic is given Table 6. In Table 3 we present colliding message pairs. Note that we have used an additional first message block to generate several different initial values for the second message block. These degrees of freedom were needed for the attack to work, otherwise we could not find a confirming message pair.

Table 3. Collision for 20 steps of SM3

h_0	7380166f 4914b2b9 172442d7 da8a0600 a96f30bc 163138aa e38dee4d b0fb0e4e
m_0	03c98f41 a8bda164 709a299c d76610eb 26b351ac 53547024 8efdff59 7e818400 4188b7f1 954faf0e a32f9984 6e5d8975 dc3b528a 973480e4 f6be9d9b cf07f13e
h_1	5e801aac 4b8c7a46 c8f34646 3b2420c1 97e775ae e3a6c399 83a05d40 6a257995
m_1	559654cd 8d4f9e94 8ca64e4a b85d989c 8c185880 b51caad1 03eca739 be66a265 ca21ab71 9c341028 2c043967 d4617038 bf6744ca d8772f12 a58e12c0 35f4f9f2
m_1^*	559654cd 8d1f9e84 8ca64e5a b85d989c 8c185880 950cbad1 03eca739 be66a265 ca71ab61 9c341028 2c043967 d4617038 bf6744ca d8772f12 a58e12c0 35f4f9f2
Δm_1	00000000 00500010 00000010 00000000 00000000 20101000 00000000 00000000 00500010 00000000 00000000 00000000 00000000 00000000 00000000 00000000
h_2	b2033829 677c16d2 a6de9db9 fd898668 a9119d20 476364d6 a0838adc 08d3833d

5.2 Free-Start Collision

Using the starting point given in Table 7 and our automatic search algorithm described in Section 4, we are able to construct free-start collisions for up to 24 steps of SM3. The differential characteristic is given in Table 8. In Table 4 we present a colliding message pair and IV pair resulting in a collision after 24 steps. Again this attack has practical complexity. The main difference in this attack to previous attack is, that we had to place the non-linear part at the end to get sparse characteristics.

Table 4. Free-Start-Collision for 24 steps of SM3

h_0	898991b0	8de47668	6e54847c	9167ff5e	3c7d51fe	e2101301	6c53d522	7b3809df
h_0^*	898991b0	8da47668	ee54847c	1127ff5e	3c7d51fe	e2100301	ec53d522	fb3819df
Δh_0	00000000	00400000	80000000	80400000	00000000	00001000	80000000	80001000
m	07595c54	e01e0245	facd449a	07ca096d	510445e8	4e1d0dff	97f2a3c0	79f02f14
	2ebfac50	48cdde2d	e88f68e1	2b5032d1	3aa9a79f	656d1380	693417c1	ce82a62a
m^*	07595c54	e01e0245	7acd449a	07ca096d	510445e8	4e1d0dff	97f2a3c0	79f02f14
	2ebfac50	48cdde2d	e88f68e1	2b5032d1	3aa9a79f	656d1380	693417c1	ce82a62a
Δm	00000000	00000000	80000000	00000000	00000000	00000000	00000000	00000000
	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
h_1	a83d45dc	9eda869d	48baa718	78ccd026	25696682	d0be9f4a	e70babf3	f4852e70

6 Conclusions

Since the collision attacks of Wang et al. [13, 14] on MD5 and SHA-1 many cryptographers are convinced that these widely deployed hash functions can no longer be considered secure. As a consequence, NIST proposed the transition from SHA-1 to the SHA-2 family and many companies and organization are migrating to SHA-2. Furthermore, researchers are evaluating alternative hash functions in the SHA-3 initiative. In this work, we analyze the Chinese hash function standard SM3. SM3 was designed by Wang et al [1] and is published by the Chinese Commercial Cryptography Administration Office for the use of electronic authentication service systems. The amount of cryptanalytic results on SM3 is low compared to other hash function standards.

The design of SM3 is very similar to the MD4 family, in particular to SHA-256. Since new collision attacks on SHA-256 and similar hash functions have been shown recently, a revaluation of the security of similar hash functions such as SM3 seems to be necessary. The attacks are based on a the concept of generalized conditions and an automatic search algorithm. Recently, Mendel et al. improved and extended the technique such that it can be applied on more complex ARX based hash function. In this paper we develop the methods by Mendel et al. for SHA-256 [7] further and apply them on SM3. We show how the technique can be effectively applied to SM3. Furthermore, we present a collision for 20 steps and a free-start collision for 24 steps of SM3. These are the first collision attacks on step-reduced SM3 and both attacks have practical complexity.

Acknowledgments. Part of this work was done while Florian Mendel was with KU Leuven. The work has been supported in part by the Austrian Science Fund (FWF), project P21936-N23 and by the European Commission under contract ICT-2007-216646 (ECRYPT II).

Table 5. Starting point for a collision for 20 steps of SM3

i	∇A_i	∇E_i	∇W_i	$\nabla W'_i$
-4	`--------------------------------`	`--------------------------------`		
-3	`--------------------------------`	`--------------------------------`		
-2	`--------------------------------`	`--------------------------------`		
-1	`--------------------------------`	`--------------------------------`		
0	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
1	`????????????????????????????????`	`????????????????????????????????`	`---------x-x---------------x----`	`--x------x---------x-------x----`
2	`????????????????????????????????`	`????????????????????????????????`	`---------------------------x----`	`---------------------------x----`
3	`????????????????????????????????`	`????????????????????????????????`	`--------------------------------`	`--------------------------------`
4	`????????????????????????????????`	`????????????????????????????????`	`--------------------------------`	`---------x-x---------------x----`
5	`--------------------------------`	`--------------------------------`	`--x--------x-------x------------`	`--x--------x-------x------------`
6	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
7	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
8	`--------------------------------`	`--------------------------------`	`---------x-x---------------x----`	`---------x-x---------------x----`
9	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
10	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
11	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
12	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
13	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
14	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
15	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
16	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
17	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
18	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
19	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`

Table 6. Differential characteristic for a collision for 20 steps of SM3

i	∇A_i	∇E_i	∇W_i	$\nabla W'_i$
-4	`--------------------------------`	`--------------------------------`		
-3	`--------------------------------`	`--------------------------------`		
-2	`--------------------------------`	`--------------------------------`		
-1	`--------------------------------`	`--------------------------------`		
0	`-------1------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
1	`xxx-----xx----------xxxxxxxx--1-`	`x-x----xxxxx------x-10xxx-xx--xx`	`--0------x-x-------1-------x----`	`--x------x-1-------x-------x----`
2	`-xx-x---xx-xx--xxx--x--------xxx`	`---x0xxxx----xx-x-xxx-x--x----xx`	`---------------------------x----`	`---------------------------x----`
3	`x-x-x--------xxxxxxxx-------x-x-`	`xx-x--x1x-xxx-x-----------xxx---`	`--------------------------------`	`--------------------------------`
4	`xxxxx--------------xx--------xxx`	`xxxxxxxxxxxxxxx-------------x-xx`	`---------0-1---------------0----`	`---------x-x---------------x----`
5	`--------------------------------`	`--------------------------------`	`--x------0-x-------x-------1----`	`--x--------x-------x------------`
6	`--------------------------------`	`--------------------------------`	`---------------------------0----`	`--------------------------------`
7	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
8	`--------------------------------`	`--------------------------------`	`---------x-x---------------x----`	`---------x-x---------------x----`
9	`--------------------------------`	`--------------------------------`	`--0--------1-------1------------`	`--------------------------------`
10	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
11	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
12	`--------------------------------`	`--------------------------------`	`---------1-0---------------0----`	`--------------------------------`
13	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
14	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
15	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
16	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
17	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
18	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
19	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`

Table 7. Starting point for a free-start collision for 24 steps of SM3

i	∇A_i	∇E_i	∇W_i	$\nabla W'_i$
-4	`---------x--------x-------------`	`------x------------x------------`		
-3	`---------x----------------------`	`-------------------x------------`		
-2	`---------x----------------------`	`-------------------x------------`		
-1	`--------------------------------`	`--------------------------------`		
0	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
1	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
2	`--------------------------------`	`--------------------------------`	`x-------------------------------`	`x-------------------------------`
3	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
4	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
5	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
6	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
7	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
8	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
9	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
10	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
11	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
12	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
13	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
14	`x--------x-------x--------------`	`--------------------------------`	`--------------------------------`	`x--------x-------x--------------`
15	`????????????????????????????????`	`????????????????????????????????`	`--------------------------------`	`--------------------------------`
16	`????????????????????????????????`	`????????????????????????????????`	`--------------------------------`	`--------------------------------`
17	`????????????????????????????????`	`????????????????????????????????`	`--------------------------------`	`---x-------------x-x------------`
18	`????????????????????????????????`	`????????????????????????????????`	`x--------x-------x--------------`	`x--------x-------x--------------`
19	`????????????????????????????????`	`????????????????????????????????`	`--------------------------------`	`--------------------------------`
20	`---------x--------x-------------`	`------x------------x------------`	`--------------------------------`	`x-x-xx---x-x-x---x-xxx-------x--`
21	`---------x----------------------`	`-------------------x------------`	`---x-------------x-x------------`	`---x-------------x-x------------`
22	`---------x----------------------`	`-------------------x------------`	`--------------------------------`	`--------------------------------`
23	`--------------------------------`	`--------------------------------`	`--------------------------------`	`x-x---------------xx------------`

Table 8. Differential characteristic for a free-start collision for 24 steps of SM3

i	∇A_i	∇E_i	∇W_i	$\nabla W'_i$
-4	`---------x--------x-------------`	`------x------------x------------`		
-3	`---------x----------------------`	`-------------------x------------`		
-2	`---------x----------------------`	`-------------------x------------`		
-1	`--------------------------------`	`--------------------------------`		
0	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
1	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
2	`--------------------------------`	`--------------------------------`	`x-------------------------------`	`x-------------------------------`
3	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
4	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
5	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
6	`--------------------------------`	`--------------------------------`	`1-------------------------------`	`--------------------------------`
7	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
8	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
9	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
10	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
11	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
12	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
13	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
14	`x--------x-------x--------------`	`--------------------------------`	`0--------0-------0--------------`	`x--------x-------x--------------`
15	`x----x--xx-xx-xxx-xx--x-------x-`	`----xx-------x-------xx-----x-x-`	`--------------------------------`	`--------------------------------`
16	`-xx-x--x-x---x---x---x-x--xx----`	`----x--x--x-x------x--1x--------`	`--------------------------------`	`--------------------------------`
17	`x--x----------x----xx-----x----x`	`---------x-x------x-x-x0------x-`	`---0-------------0-1------------`	`---x-------------x-x------------`
18	`xx----------------xx-x----------`	`----x------x-------xxx----xx--xx`	`x--------x-------x--------------`	`x--------x-------x--------------`
19	`xxxxxxx--xxx------x--x----x-x-x-`	`-------xxx---------x-00-------xx`	`--------------------------------`	`--------------------------------`
20	`---------x--------x-------------`	`------x------------x----------11`	`--------------------------------`	`x-x-xx---x-x-x---x-xxx-------x--`
21	`---------x----------------------`	`-------------------x--0---------`	`---x-------------x-x------------`	`---x-------------x-x------------`
22	`---------x----------------------`	`-------------------x------------`	`1--------1-------0--------------`	`--------------------------------`
23	`--------------------------------`	`--------------------------------`	`--------------------------------`	`x-x---------------xx------------`

References

1. Specification of SM3 cryptographic hash function (in Chinese), http://www.oscca.gov.cn/UpFile/20101222141857786.pdf
2. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
3. Gu, J., Purdom, P.W., Franco, J., Wah, B.W.: Algorithms for the Satisfiability (SAT) Problem: A Survey. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp. 19–152. American Mathematical Society (1996)
4. Kircanski, A., Shen, Y., Wang, G., Youssef, A.: Boomerang and Slide-Rotational Analysis of the SM3 Hash Function. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography. LNCS. Springer (to appear, 2012)
5. Mendel, F., Nad, T., Scherz, S., Schläffer, M.: Differential Attacks on Reduced RIPEMD-160. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 23–38. Springer, Heidelberg (2012)
6. Mendel, F., Nad, T., Schläffer, M.: Cryptanalysis of Round-Reduced HAS-160. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 33–47. Springer, Heidelberg (2012)
7. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer, Heidelberg (2011)
8. Mendel, F., Nad, T., Schläffer, M.: Collision Attacks on the Reduced Dual-Stream Hash Function RIPEMD-128. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 226–243. Springer, Heidelberg (2012)
9. National Institute of Standards and Technology. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Federal Register 27(212), 62212–62220 (November 2007), http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf
10. Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
11. Stinson, D.R.: Some Observations on the Theory of Cryptographic Hash Functions. Des. Codes Cryptography 38(2), 259–277 (2006)
12. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
13. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
14. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
15. Zou, J., Wu, W., Wu, S., Su, B., Dong, L.: Preimage Attacks on Step-Reduced SM3 Hash Function. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 375–390. Springer, Heidelberg (2012)

Many Weak Keys for PRINTcipher: Fast Key Recovery and Countermeasures

Stanislav Bulygin[1,2], Michael Walter[3], and Johannes Buchmann[1,2]

[1] Center for Advanced Security Research Darmstadt - CASED
Mornewegstraße 32, 64293 Darmstadt, Germany
{Stanislav.Bulygin}@cased.de

[2] Technische Universität Darmstadt, Department of Computer Science
Hochschulstraße 10, 64289 Darmstadt, Germany
michael.walter@swel.com, buchmann@cdc.informatik.tu-darmstadt.de

[3] University of California, San Diego, Department of Computer Science and Engineering
9500 Gilman Drive, Mail code 0404, La Jolla, CA 92093-5004, USA
miwalter@eng.ucsd.edu

Abstract. In this paper we investigate the invariant property of PRINTcipher first discovered by Leander *et al.* in their CRYPTO 2011 paper. We provide a complete study and show that there exist 64 families of weak keys for PRINTcipher–48 and as many as 115,669 for PRINTcipher–96. Moreover, we show that searching the weak key space may be substantially sped up by splitting the search into two consecutive steps. We show that for many classes of weak keys, key recovery can be done with very small time complexity in the chosen/known plaintext scenario. This shows that the cipher is actually much more vulnerable to this type of attacks than was even thought previously. Still, effective countermeasures exist against the attack. The method of finding all weak key families has value on its own. It is based on Mixed Linear Integer Programming and can be adapted to solving other interesting problems on similar ciphers.

Keywords: PRINTcipher, invariant coset attack, mixed integer linear programming, weak keys, chosen plaintext attack, key recovery.

1 Introduction

Lightweight cryptography gained its importance due to emergence of many applications that involve using small and resource constraint devices such as RFID tags, smart cards, and sensor networks. Conventional cryptographic algorithms turned out to be too massive to be implemented on such devices. Therefore, the need for new cryptographic primitives arose in the community. In particular, the whole arsenal of lightweight block ciphers has been developed in recent years to satisfy the needs of secure usage of small devices. The block cipher PRESENT is one outstanding example that gained popularity [1]. Other block ciphers, such as

KATAN and KTANTAN family [2], LED block cipher [3] and many others were presented recently. Following the design principle of PRESENT, several block ciphers with even more lightweight structure have been proposed: PRINTcipher [4] and EPCBC [5] are immediate examples, as well as SPONGENT hash family [6].

PRINTcipher is a block cipher proposed at CHES 2010 [4] and is really pushing the design solutions for lightweight ciphers to their limits. PRINTcipher has been an object of numerous attacks since then. Methods of linear and differential cryptanalysis [7,8,9] as well as algebraic cryptanalysis [10] have been proposed to analyze PRINTcipher. Also certain results on side channel analysis of PRINTcipher appeared [10,11]. Notably, cryptanalytic methods proposed so far were not able to break more than $\frac{2}{3}$ of PRINTcipher's rounds for a large portion of keys[1].

At CRYPTO 2011 Leander *et al.* proposed a very powerful attack, which they called the *invariant coset attack* [12]. Using this attack it is possible to break the *full* PRINTcipher (both the 48- and the 96-bit versions) for a significant portion of keys using only a few chosen plaintexts. Note, however, that despite the fact that distinguishing a weak key can be done in unit time, the complete key recovery for weak key families presented in [12] would take quite a considerable time in practice. The authors of [12] presented two families of weak keys for PRINTcipher–48, each having 2^{51} keys, as well as two families for PRINTcipher–96. However, it remained unknown whether other families existed and what was a systematic way of finding them.

In this paper we thoroughly investigate the invariant property of PRINTcipher. We suggest a systematic method of obtaining all invariant cosets of the cipher and corresponding families of weak keys. In particular, we are able to recover *all* 64 families of weak keys for PRINTcipher–48 and *all* 115,669 families for PRINTcipher–96. It is possible to speed the key recovery process compared to a simple brute force of the remaining key bits. In the case of PRINTcipher–48 we identified many weak keys that can be recovered in a matter of minutes on a single PC. This shows that the cipher is actually much more vulnerable to this type of attacks than was even thought previously. The key recovery procedure is improved compared with the simple brute force by a factor of 8 to 2048 depending on the weak key class. For PRINTcipher–96 the gain can be as large as 2^{27}. Similarly to [12] we analyze countermeasures against the attack. Since our analysis of the invariant property is complete we may argue about security of the cipher against the invariant coset attack. In particular, *we claim that with countermeasures suggested in this paper both versions of PRINTcipher remain secure ciphers.*

The method of finding weak key families is based on Mixed Integer Linear Programming (MILP). It appears to be a novel technique to use MILP in such a context. Noteworthy is also that this method can be adapted to solving other interesting problems on similar ciphers and therefore has value on its own.

The outline of the paper is as follows. In Section 2 we briefly recall the definition of PRINTcipher and outline its properties that are relevant for the further

[1] The best attack of [9] can break 31 out of 48 rounds of PRINTcipher–48 for the fraction of 0.036% of keys using almost the entire code book.

exposition; we also recall the attack of [12]. Section 3 presents methods of obtaining and using invariant cosets (or *invariant projected subsets* as we call them). Section 3.1 presents the general background on defining sets of invariant projected subsets and their characterization. These defining sets then give rise to families of weak keys. Section 3.2 presents a method of finding all defining sets based on MILP that turns out to be highly efficient in practice. Section 3.3 summarizes the results for PRINTcipher–48 and PRINTcipher–96 obtained by our methods, computing the number of keys in each weak key family and complexity of the key recovery, as well as protecting measures. Finally, Section 4 provides a retrospective of the proposed methods and puts them in a more general context providing possible further directions for research. In Appendices A and B the reader may find the table with results for PRINTcipher–48 as well as a worked out example illustrating computations from Section 3.3 resp.

2 The Block Cipher PRINTcipher

2.1 Description of the Cipher

PRINTcipher [4] is a substitution-permutation network, which design is largely inspired by the block cipher PRESENT [1]. The main differences with PRESENT is the absence of a key schedule (all round keys are the same and are equal to a master key) and key-dependent S-Boxes. PRINTcipher comes in two variations: PRINTcipher-48 encrypts 48-bits blocks with an 80-bit key and has 48 rounds, PRINTcipher-96 encrypts 96-bit blocks with a 160-bit key and has 96 rounds. Here we present a short overview of the cipher, referring the reader to [4] for a more detailed description and analysis.

The encryption process of PRINTcipher–48 follows a classical SP-network structure, see Figure 1 for the round function. Both versions have similar structure. The key $k = (sk_1, sk_2)$ is divided into two parts. The subkey sk_1 is used for XORing with the state and sk_2 defines the S-Box layer, see below. The linear diffusion layer is implemented by a bit permutation similar to PRESENT and is given by the map P:

$$P(i) = \begin{cases} 3i \bmod n-1 & \text{for } 0 \le i \le n-2, \\ n-1 & \text{for } i = n-1, \end{cases} \tag{1}$$

so that the $i-$th bit of the state is moved to the position $P(i)$. The RC_i for $i = 1, \ldots, n$ are 6-bit long round constants obtained via a certain LFSR and are placed in the last two position triplets. The S-Box layer is a layer of $n/3$ 3-bit S-Boxes, where each S-Box is chosen according to the value of the two corresponding bits of the subkey sk_2. Therewith, there are 4 possible S-Boxes at each position called V_0, V_1, V_2, V_3 in [4]. One may also consider such an S-Box as a composition of a key dependent bit permutation that acts on groups of three bits (triplets) and then followed by the layer of fixed S-Boxes, each one being a 3-bit S-Box with the truth table as in Table 1. This S-Box, called V_0 in [4], is preceded by a key-dependent permutation defined by Table 2. In Table 2 the

Table 1. Truth table for the S-Box V_0

x	0	1	2	3	4	5	6	7
$S[x]$	0	1	3	6	7	4	5	2

Table 2. Key depended permutation

a_1	a_0	Permutation
0	0	(0,1,2)
0	1	(0,2,1)
1	0	(1,0,2)
1	1	(2,1,0)

three input bits are permuted according to the two consecutive bits from the subkey sk_2 called a_0 and a_1.

We need properties of masks that exist for keyed S-Boxes of PRINTcipher. By an α–β *mask* we understand a situation when fixing some α out of 3 S-Box input values yields fixed β output values regardless of the values at other input/output positions. There is obviously a 3–3 mask, since knowledge of three input bits of an S-Box yields all three output bits. There is also obviously a 0–0 mask. It turns out that for certain values of sk_2-bits, PRINTcipher's S-Box has 2–2 masks for all $3 \cdot 3 = 9$ possible combinations of input and output positions. Table 3 shows that each possible 2-mask on the input bits has a corresponding 2-mask for output bits. In this table $+/-$ notation shows which bits of the mask are fixed ($+$) and ones which are not ($-$). For example, the first row says that if we have a mask where the first two bits of both input and output to a keyed S-Box are fixed, then both S-Boxes with $a_0 = 0$ satisfy the mask. Moreover, the two fixed output bits of the mask, as well as the two input bits, must be 0. Table 3 plays an important role in studying the invariant coset attack of [12]. Notably, there are also 2–1, 2–0, 1–0 masks and no others. Clearly, the 2–2 and 3–3 masks are the most desirable for cryptanalytic properties.

2.2 Invariant Coset Attack of Leander *et al.*

In their work [12] Leander *et al.* showed that for PRINTcipher there exist subsets of plaintexts which, when encrypted with keys from certain other subsets, end

Table 3. Behavior of the 2–2 masks in S-Boxes of PRINTcipher

Input mask	Output mask	Values of (a_0, a_1)	input values	output values
$++-$	$++-$	0*	00*	00*
$++-$	$+-+$	10	10*	1*1
$++-$	$-++$	11	11*	*10
$+-+$	$++-$	10	0*0	00*
$+-+$	$+-+$	00 or 11	0*1 or 1*0	1*1
$+-+$	$-++$	01	1*1	*10
$-++$	$++-$	11	*00	00*
$-++$	$+-+$	01	*10	1*1
$-++$	$-++$	*0	*11	*10

up in the same subset of plaintexts. Thus they showed an invariant property for PRINTcipher under certain weak keys. The subsets they presented are of quite special form: they are linear cosets. An invariant coset for the plaintexts is of the form $U + d$ for some linear subspace $U \subseteq \mathbb{F}_2^n$ and some vector $d \in \mathbb{F}_2^n$; the weak keys are from the coset $U + c + d$ for some other vector $c \in \mathbb{F}_2^n$. Notably, the U's are linear subspaces of a special kind. Namely, they are linear subspaces where all vectors have the value 0 at certain positions. The vectors c and d then "adjust" the zeros, so that the invariant property holds. The authors of [12] show that distinguishing the weak keys so obtained can be done using only a few chosen plaintexts, thus presenting a powerful attack. In the *invariant coset attack* they heavily use the property of 2–2 masks presented in Table 3.

3 Obtaining and Exploiting Invariant Projected Subsets

In this section we describe methods of obtaining all invariant cosets for both versions of PRINTcipher. These invariant cosets are of the same type as in [12], i.e. we only consider cosets that are described by the "projection" equations specifying values of vectors at certain positions. In Section 3.1 we investigate defining sets of invariant cosets (or invariant projected subsets, as we call them), i.e. sets of positions the subsets are projected onto. We provide their characterization that is used in Section 3.2 to provide a method for finding all possible defining sets using techniques of Mixed Integer Linear Programming. In Section 3.3 we describe the structure of weak key classes that stem from the invariant projected subsets, compute a number of elements in each class and compute time complexity of weak key recovery. Notably, we show that it is possible to speed up the search process by separating the search space into two steps.

3.1 Defining Sets of Invariant Projected Subsets

Definition 1. *A* projected subset $U \subset \mathbb{F}_2^n$ *is defined as*

$$U := \{u = (u_0, \dots, u_{n-1}) \in \mathbb{F}_2^n | u_{i_1} = a_1, \dots, u_{i_t} = a_t \text{ for some } t \geq 0,\\ 0 \leq i_1 < \cdots < i_t \leq n-1 \text{ and some } a = (a_1, \dots, a_t) \in \mathbb{F}_2^t\}.$$

We define $def_U \subset \mathbb{Z}_n$ to be the subset of indexes $i_1, \dots, i_t$ with the above property and for such a subset we define a vector $val_U \in (\mathbb{F}_2 \cup \{'*'\})^n$ as follows: $val_U[i_j] = a_j, 1 \leq j \leq t$ and $val_U[i] =' *', i \in \mathbb{Z}_n \setminus def_U$. We call def_U the defining set *of U.*

Definition 2. *Let $V \subset \mathbb{Z}_n$ with n divisible by 3, then V is a $\bar{1}-$*subset *iff $\forall\ 0 \leq j < n/3 : |\{3j, 3j+1, 3j+2\} \cap V| \neq 1$.*

The above definition calls a subset of positions a $\bar{1}-$subset iff in each consecutive triplet of positions there are 0, 2, or 3 elements from that subset.

Definition 3. *Let $T \subseteq \mathbb{Z}_n$ with n divisible by 3. For $i \in \{1,2,3\}$ define*

$$T_i := \bigcup_{\substack{0 \le j < n/3 \\ |\{3j,3j+1,3j+2\} \cap T| = i}} [\{3j, 3j+1, 3j+2\} \cap T].$$

In other words, we divide the set $\{0,\dots,n-1\}$ into triplets and then look which triplets have exactly i elements of T in them and collect these elements into T_i. Let U be a projected subset with the defining set def_U. For short notation we denote $U_i := (def_U)_i$ for $i = 1,2,3$.

Now let $E_{sk_1,sk_2,r} = S_{sk_2} \circ RC_r \circ P \circ XOR_{sk_1}$ be the round function of PRINTcipher-n for the round $1 \le r \le n$, where sk_1 and sk_2 are parts of the secret key as defined in Section 2.1.

Theorem 1. *Let $T \subset \mathbb{Z}_n$ with n divisible by 3 and*

1. $T \cap \{n-6,\dots,n-1\} = \oslash$;
2. *T is a $\bar{1}-$subset of $\mathbb{Z}_n$;*
3. $\forall\ 0 \le j < n/3 : |\{3j, 3j+1, 3j+2\} \cap T| = |\{3j,3j+1,3j+2\} \cap P(T)|$.

Then there exists a projected subset $U \subset \mathbb{F}_2^n$ with $def_U = T$ such that $E_{sk_1,sk_2,r}(U) = U$ for some $sk_1 \in \mathbb{F}_2^n$ and $sk_2 \in \mathbb{F}_2^{2n/3}$ and any $r \ge 1$.The set $\{val_U[i] : i \in U_2\}$ is uniquely determined by T.
Vice versa: if for a projected subset $U \subset \mathbb{F}_2^n$ holds $E_{sk_1,sk_2,r}(U) = U$ for some $sk_1 \in \mathbb{F}_2^n$ and $sk_2 \in \mathbb{F}_2^{2n/3}$ and any $r \ge 1$, then def_U satisfies conditions (1.)–(3.) above.

Proof. $(\Rightarrow)$: We want to show that there exist $sk_1 \in \mathbb{F}_2^n$ and $sk_2 \in \mathbb{F}_2^{2n/3}$ such that $E_{sk_1,sk_2,r}(U) = U$ for all $r \ge 1$ for some projected subset U with $def_U = T$. For the construction of U we need to specify the values in val_U as well as the corresponding values of sk_1 and sk_2. Note that due to properties (2.) and (3.), the set of positions $P(T)$ is also a $\bar{1}$-subset. Now observe that positions of $P(T)$ define input masks to the S-Box layer S_{sk_2} and those from T define output masks of that layer. Indeed, we start with the positions from T at the beginning of the round. So in order to have an invariant property, we have to end with T as well. Due to properties (2.) and (3.) we have that only 0–0, 2–2 and 3–3 cases are possible for the S-Box masks (see Section 2.1). Note that in the case of a 2–2 mask most of the time the corresponding values of the key sk_2 are fixed and determined by T, see Table 3. In the case of ambiguity, we may take arbitrary value for the corresponding bit of sk_2. Also from Table 3 we see that for the 2–2 masks S-Box output values are uniquely defined in all cases, the same for input values except for the mask $+-+ \to +-+$. In this case arbitrary but fixed choice of the corresponding bits of sk_2 resolves the ambiguity. After the 2–2 cases are resolved (either through mandatory assignment or arbitrary but now fixed choice) we move to the 0–0 and 3–3 masks. Here both outputs of S-Boxes and the corresponding values of sk_2 can be assigned arbitrary values (sk_2 bits are equal in this case). Note that assignment of output values in the case of a 3–3

mask provides an assignment of values in $\{val_U[i] : i \in U_3\}$. We have that inputs and outputs to the S-Boxes at positions $P(T)$ and T resp. are now defined. This yields, via the operation XOR_{sk_1}, unique values for the key sk_1 at positions T. Moreover, the values in val_U are now fixed too. Therewith the set U is now fully defined by $def_U = T$ and such constructed val_U. Note that since outputs of S-Boxes at positions $T_2 = U_2$ (see Definition 3) are uniquely determined by the 2–2 masks, we have that the values $\{val_U[i], i \in U_2\}$ are uniquely determined. Condition (1.) makes sure that the round counter RC_i is avoided (i.e. there we have only 0–0 mask) and therewith there is no chance to change input/output values of S-Boxes, and so invariant property is preserved.
$(\Leftarrow)$: For the reverse direction, note that def_U should satisfy (2.) and (3.), since for the invariant property we need an α–α mask and there are only 0–0, 2–2, and 3–3 of this sort, see Section 2.1. Again, the invariant property must not be spoiled by the round counter, thus the condition (1.) should be satisfied.

We refer to the example in Appendix B for assisting in understanding the notation and how the invariant property works. Note that in the case Theorem 1 holds, we have $E_{sk_1,sk_2,r}(V + d) = V + d$ for a certain linear subspace $V \subseteq \mathbb{F}_2^n$ and $d \in \mathbb{F}_2^n$. This $V + d$ is an *invariant coset* as per [12]. In order to follow our terminology, we prefer the term *invariant projected subset.* Also note that if $U = V + d$ is an invariant projected subset, then def_U is a disjoint union of U_2 and U_3: $def_U = U_2 \sqcup U_3$, see Definition 3. As a result of Theorem 1 we have a one-to-one correspondence between defining sets of invariant projected subsets and subsets of $\mathbb{Z}_n$ that satisfy (1.)–(3.) of Theorem 1.

3.2 Defining Sets via Polytopes in $\mathbb{Z}^n$

Now having the characterization of defining sets, we provide an efficient method of finding all subsets of $\mathbb{Z}_n$ satisfying conditions (1.)–(3.) of Theorem 1. The method is based on providing a one-to-one correspondence between defining sets of the invariant projective subsets of PRINTcipher–n and points of a certain polytope in $\mathbb{Z}^n$. One can then efficiently find these points by applying techniques of Mixed Integer Linear Programming (MILP).

Theorem 2. *Let IP_n be a subset in $\{0,1\}^n \subset \mathbb{Z}^n$, where n is divisible by 3, defined as a subset of those $x = (x_0, \ldots, x_{n-1}) \in \{0,1\}^n$ that satisfy*[2]

$$x_{n-6} = \cdots = x_{n-1} = 0,$$
$$\textit{for all } 0 \leq j < n/3 :$$

$$x_{3j} + x_{3j+1} + x_{3j+2} = x_{P^{-1}(3j)} + x_{P^{-1}(3j+1)} + x_{P^{-1}(3j+2)}, \tag{2}$$
$$x_{3j} + x_{3j+1} \geq x_{3j+2}, \quad x_{3j} + x_{3j+2} \geq x_{3j+1}, \quad x_{3j+1} + x_{3j+2} \geq x_{3j},$$
$$\sum_{i=0}^{n-1} x_i > 0.$$

[2] Note that the arithmetic is integer.

For each $v \in IP_n$ define $T_v := \{i | v_i = 1\} \subset \mathbb{Z}_n$. Then for every $v \in IP_n$ the set T_v satisfies (1.)–(3.) of Theorem 1. Conversely, for each set T that satisfies (1.)–(3.) of Theorem 1 there exists $v \in IP_n$ such that $T = T_v$.

Proof. For the proof we need the following observation, which is proved by a direct inspection. Namely, for a pair of vectors $a = (a_0, a_1, a_2)$ and $b = (b_0, b_1, b_2)$ from $\mathbb{Z}^3 \cap \{0,1\}^3$ that satisfy

$$b_0 + b_1 + b_2 = a_0 + a_1 + a_2, \quad b_0 + b_1 \geq b_2, \quad b_0 + b_2 \geq b_1, \quad b_1 + b_2 \geq b_0,$$

only the following cases for Hamming weights of a and b are possible: $wt(a) = wt(b) = i$ for $i = 0, 2, 3$. Now property (1.) follows directly from the condition in the first line of (2). It is not hard to see that the above observation easily yields claimed properties (2.) and (3.). The last line of the conditions (2) makes sure that we do not include the trivial all-zero solution.

It is not difficult to see that the converse statement is also true.

Combining Theorem 1 and Theorem 2 we obtain a one-to-one correspondence between defining sets of invariant projected subsets and vectors from $IP_n \subset \{0,1\}^n$.

So the question now is how to compute all elements of IP_n for $n = 48, 96$. One possible solution is to use the MILP. Recall that the problem of the MILP is to optimize (i.e. either maximize or minimize) a linear objective function under linear constraints with solutions lying in $\mathbb{Z}^p \times \mathbb{R}^{n-p}$. In our case the feasible solutions are all in $\{0,1\}^n$ as per (2) and we thus have a binary integer linear program – a certain kind of MILP. Therefore, we can use an MILP solver with the additional requirement that the solutions should lie in $\{0,1\}^n$. Also, in our case we do not actually have an optimization problem, but rather a constraint satisfaction problem that likewise can be solved by an MILP solver.

3.3 Families and Classes of Weak Keys

In this subsection we provide detailed results for PRINTcipher that can be obtained by the method from Section 3.2. We used SAGE computer algebra system [13] together with the MILP solver CPLEX[3] through the SAGE interface[4].

The results as presented in Tables 4 and 5 for both versions of PRINTcipher. In the tables we also indicate how our work improves the initial result of [12].

Each *family* is composed of potentially several *classes* of weak keys. By a class we understand a set of weak keys that ensure the invariant property for certain plaintext subsets, see more on that below. The *family*, in turn, unites all classes of weak keys stemming from invariant projected sets with the same defining set. We can provide only an upper bound on the number of all weak keys, since weak key classes can intersect, see "How many weak keys are there in a family defined

[3] IBM ILOG CPLEX 12.1 under the academic license.

[4] See `http://sagemath.org/doc/reference/sage/numerical/mip.html?highlight=linear%20programming`

Table 4. Results and comparison for PRINTcipher–48

	Our analysis	Leander et al. [12]
# weak key families / all found?	64/**Yes**	2/No
upper bound on # weak keys	$2^{52.5}$	2^{52}
fastest time for key recovery in CP/KP scenario	2^{24}	2^{38}

Table 5. Results and comparison for PRINTcipher–96

	Our analysis	Leander et al. [12]
# weak key families / all found?	115,669/**Yes**	2/No
upper bound on # weak keys	$2^{117.7}$	2^{102}
fastest time for key recovery in CP/KP scenario	2^{30}	2^{76}

by def_U?" below. We also provide the minimum of the time complexities of all possible *classes* of weak keys.

Table 6 in Appendix A presents detailed results for PRINTcipher–48. Defining sets of weak key families found in [12] are marked in bold there.

Description of Weak Key Classes. From Theorem 1 we know that once we have a subset of positions $T \subset \mathbb{Z}_n$ that satisfies conditions (1.)–(3.) there exists a key $k = (sk_1, sk_2)$ and a projected subset U with $def_U = T$ such that $E_{sk_1,sk_2,r}(U) = U$ for all $r \geq 1$. Moreover, the set of projected values $\{val_U[i] : i \in U_2\}$ is uniquely determined by T. Now we would like to have more: we want to have a description of all *classes* of weak keys (i.e. those that preserve the invariant property) that correspond to $T = def_U$. These classes form a *family*.

For this we first need to fix values for all elements in $\{val_U[i] : i \in def_U\}$. Values $\{val_U[i] : i \in U_2\}$ are uniquely determined by T as per Theorem 1. Then choose a vector $v_3 \in \mathbb{F}_2^{|U_3|}$ and assign $val_U[i_j] = v_3[j], 1 \leq j \leq |U_3|$, where $U_3 = \{i_1, \ldots, i_{|U_3|}\}$ with $i_1 < \cdots < i_{|U_3|}$. So now the invariant projected subset U is fully defined. Let us show that there exists a *class* of weak keys $WK(def_U, v_3)$ such that for any $(sk_1, sk_2) = k \in WK(def_U, v_3) : E_{sk_1,sk_2,r}(U) = U$ for all $r \geq 1$. Let us describe separately the two parts sk_1 and sk_2 of a key $k = (sk_1, sk_2) \in WK(def_U, v_3)$.

We start with sk_2. Define for $0 \leq j \leq 3$:

$$S_j := \{0 \leq i < n/3 : |\{3i, 3i+1, 3i+2\} \cap def_U| = j\}.$$

So S_j collects those S-Boxes that have j bits both in input and output masks (i.e. a j–j mask). We construct the sk_2-part of k as follows.

- Bits $sk_2[2i]$ and $sk_2[2i+1]$ for $i \in S_3$ can attain arbitrary values. There are $\frac{2}{3}|U_3|$ such bits (a 3–3 mask).
- For $i \in S_2$, the bits $sk_2[2i]$ and $sk_2[2i+1]$ get their values according to column 3 of Table 3 by examining the corresponding parts of U_2 and $(P(def_U))_2$ to get $+/-$ masks. The star sign '*' means that the corresponding bit of sk_2 can attain arbitrary value. Denote by $U^* \subset \mathbb{Z}_{2n/3}$ the set of positions of sk_2 that correspond to '*'s. In the case of the $+-+ \rightarrow +-+$ mask assign arbitrary value to $sk_2[2i] = sk_2[2i+1]$. Let us denote the set of S-Boxes yielding the $+-+ \rightarrow +-+$ mask by $U^= \subset S_2 \subset \mathbb{Z}_{n/3}$.

- Bits $sk_2[2i]$ and $sk_2[2i+1]$ for $i \in S_0$ can attain arbitrary values (a 0–0 mask). There are $\frac{2}{3}n - \frac{2}{3}|U_3| - |U_2|$ such bits. Indeed, from the length of sk_2, which is $\frac{2}{3}n$, we subtract positions from S_3 $(= \frac{2}{3}|U_3|)$ and S_2 $(= |U_2|)$.

Now determine the sk_1 part. Let the choice of sk_2 be done as above and fixed.

- Denote $Y = XOR_{sk_1}(X)$. Note that $\{val_U[i] : i \in def_U\}$ are fixed, therefore $X_i, i \in def_U$ have fixed values. Now, since the choice of sk_2 bits for active S-Boxes (i.e. those with $i \in S_2 \cup S_3$) has been fixed, we have that inputs to S-Boxes at positions $P(def_U)$ have fixed values. Therefore $Y_i, \in P^{-1}(P(def_U)) = def_U$ are fixed as well. As a result $sk_1[i] = X_i \oplus Y_i, i \in def_U$ are now determined.
- Bits $sk_1[i], i \in \mathbb{Z}_n \setminus def_U$ can attain arbitrary values. There are $n - |def_U|$ such bits.

An explicit example explaining the notation is given below:

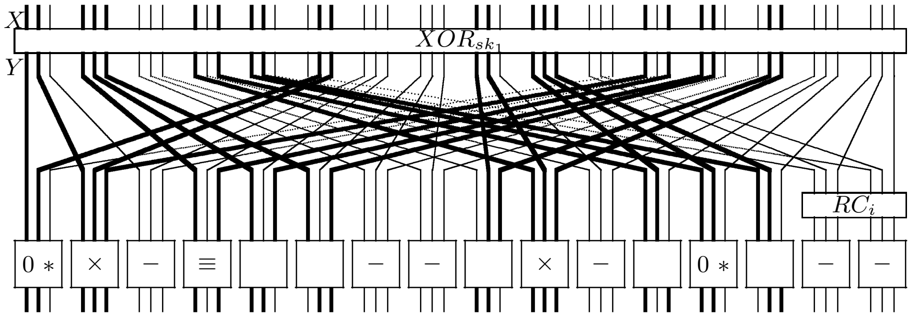


Fig. 1. Defining set no. 14, Table [6]
Bold: positions of def_U and their transition through the round
$\times$: S-Boxes 1 and 9 belonging to S_3
$-$: S-Boxes 2,6,7,10,14, and 15 belonging to S_0
$\equiv$: S-Box 3 belongs to $U^= \subset S_2$
$*$: sk_2-positions of $U^* = \{1, 25\} \subseteq \mathbb{Z}_{32}$

Having the above construction, we obtain the following proposition.[5]

Proposition 1. *Let $T \subset \mathbb{Z}_n$ satisfy (1.)–(3.) of Theorem [1]. Let $v_3 \in \mathbb{F}_2^{|T_3|}$ and $U \subset \mathbb{F}_2^n$ be a projected subset with $def_U = T$ and val_U composed of unique values for $\{val_U : i \in U_2\}$ and $\{val_U(U_3[i]) = v_3[i] : 1 \leq i \leq |U_3|\}$. Then for the set $WK(def_U, v_3)$ constructed as above holds*

$$E_{sk_1, sk_2, r}(U) = U, r \geq 1,$$

for any $(sk_1, sk_2) \in WK(def_U, v_3)$.

[5] For space reasons, we must refer to the full version of the paper for a formal proof.

How to Distinguish the Weak Keys? Now let us detail how to distinguish in the *chosen plaintext scenario* the weak keys belonging to $WK(def_U, v_3)$ for some defining set def_U of an invariant projected set U with a fixed choice of bits $\{val_U[i] : i \in U_2\}$ and $\{val_U[U_3[i]] = v_3[i] : 1 \leq i \leq |U_3|\}$. The attacker chooses an arbitrary $p \in U$ and encrypts p with a key $(sk_1, sk_2) = k \in \mathbb{F}_2^{5n/3}$ obtaining $c = E_{sk_1,sk_2,n}(p)$. If $c[i] = p[i]$ for all $i \in def_U$, i.e. $c \in U$, then k is a candidate element of $WK(def_U, v_3)$. To be sure we need several chosen plaintexts, since with probability $2^{-|def_U|}$ one has $p[i] = c[i], i \in def_U$, where the key k can be any key. So in order to distinguish we need $\#CP = \lceil \frac{5}{3}n/|def_U| \rceil$ plaintexts from U and their corresponding encryptions. For $\frac{5}{3}n = 80$ it is at most 5 chosen plaintexts, see also [12].

How Many Weak Keys Are There in a Family Defined by def_U? For a given def_U there are $2^{|U_3|}$ possibilities to assign values for the vector v_3. Therefore, each family of weak keys has $2^{|U_3|}$ classes. We want to determine

$$\Big| \bigcup_{v_3 \in \mathbb{F}_2^{|U_3|}} WK(def_U, v_3) \Big|.$$

The following bounds for the number above hold:

$$2^{|U_3|}|WK(def_U, 0)| \geq \Big| \bigcup_{v_3 \in \mathbb{F}_2^{|U_3|}} WK(def_U, v_3) \Big| \geq |WK(def_U, 0)|. \tag{3}$$

Indeed, if we fix vector $v_3 = 0$, then the number of elements in all classes is at least the number of elements in one class and is at most $2^{|U_3|}$ times the number of elements in that class, since

$$|WK(def_U, a)| = |WK(def_U, b)| \quad \forall a, b \in \mathbb{F}_2^{|U_3|}.$$

We cannot simply say that $|\cup_{v_3 \in \mathbb{F}_2^{|U_3|}} WK(def_U, v_3)|$ is equal to the upper bound, since it may happen that $WK(def_U, a) \cap WK(def_U, b) \neq \oslash$ for some $a \neq b$. Note, however, that for many def_U's the value $| \cup_{v_3 \in \mathbb{F}_2^{|U_3|}} WK(def_U, v_3)|$ *does* attain the upper bound. We skip the details due to space constraints.

Now as to the computation of $|WK(def_U, 0)|$ (or, equivalently $|WK(def_U, a)|$ for any other $a \in \mathbb{F}_2^{|U_3|}$), we just follow the lines of the argument preceding Proposition 1. Namely, if we compute the number of arbitrarily assigned key bits, we need to add the numbers of bits of $sk_1[i], i \in \mathbb{Z}_n \setminus def_U$ $(= n - |def_U|)$, from $sk_2[2i], sk_2[2i+1]$ for $i \in S_0$ and $i \in S_3$ $(= \frac{2}{3}n - \frac{2}{3}|U_3| - |U_2|$ and $= \frac{2}{3}|U_3|$ resp.), as well as the number of bits of $U^* \subset \mathbb{Z}_{2n/3}$ $(= |U^*|)$ and one bit per $U^= \subset \mathbb{Z}_{n/3}$ $(= |U^=|)$. Summing up, we obtain

$$\log_2 |WK(def_U, 0)| = \frac{5}{3}n - |def_U| - |U_2| + |U^*| + |U^=|. \tag{4}$$

Therewith the upper bound (and often the actual value) is

$$\log_2 \Big| \bigcup_{v_3 \in \mathbb{F}_2^{|U_3|}} WK(def_U, v_3) \Big| \geq \frac{5}{3}n - 2|U_2| + |U^*| + |U^=|. \tag{5}$$

Summing up the numbers obtained via (5) over all def_U we have an upper bound on the overall number of weak keys: it is $2^{52.51}$ for PRINTcipher–48 and $2^{117.7}$ for PRINTcipher–96.

What Is the Complexity of the Weak Key Recovery? Once a weak key is *distinguished*, the attacker wants to *recover* the remaining key bits that are not deduced immediately from the distinguishing phase. Of course, we may simply use several known plaintexts and brute force the keys in $WK(def_U, v_3)$, but we can actually do better: we can separate the key recovery process in two consecutive steps, each having smaller time complexity.

Step 1 (chosen plaintext). Using chosen plaintexts from the distinguishing phase brute force inactive key bits, i.e. $sk_1[i], i \in \mathbb{Z}_n \setminus def_U$ and $sk_2[2i], sk_2[2i+1], i \in S_0$. For the actual implementation of this step we assign arbitrary values to $sk_2[2i], sk_2[2i+1], i \in S_3$, $sk_2[2i] = sk_2[2i+1], i \in U^=$, $sk_2[i], i \in U^*$ and compute the corresponding bits of sk_1 to get candidate keys for testing.

Note that after assigning arbitrary values to certain key bits, we rely on the assumption that the remaining "cipher" behaves as a random permutation. Since the overall number of plaintext bits in several chosen plaintexts exceeds the number of key bits we brute force, we expect, as usual, that we have a unique solution for these key bits. Due to certain degeneration properties of PRINTcipher it is not always true, however. Without going into technical details, we just state that there exist certain positions at which bits of sk_1 may attain arbitrary values in this chosen plaintext scenario. Therefore we assign in this step some arbitrary values to these bits, call them *weird bits*, and brute force them in the next step. Denote the number of such weird bits w.

Now the computation of the work factor is similar to the one for the number of weak keys as above. We have the work factor of Step 1:

$$\log_2 WF_1 = n - |def_U| + \frac{2}{3}n - \frac{2}{3}|U_3| - |U_2| - w = \frac{5}{3}(n - |def_U|) - \frac{1}{3}|U_2| - w. \quad (6)$$

Note that the number of chosen plaintexts from the distinguishing phase is indeed enough, since the remaining "block length" $n - |def_U|$ times $\#CP$ exceeds $\log_2 WF_1$ for all def_U.

Step 2 (known plaintext). This is as far as we can go with chosen plaintexts. We cannot distinguish bits $sk_2[2i], sk_2[2i+1], i \in S_3 \cup U^=$ and $sk_2[i]$ for $i \in U^*$ having only these plaintexts. Therefore, for the second phase we take *one* known plaintext that is not in U and brute force these key bits. Note, however, that now the sk_1-bits $sk_1[P^{-1}(3i)]$, $sk_1[P^{-1}(3i+1)], sk_1[P^{-1}(3i+2)]$ corresponding to $i \in S_3$ cannot be determined from one round as in the case of chosen plaintexts where the invariant property holds. Similar situation is with the bits $sk_1[P^{-1}(3i)], sk_1[P^{-1}(3i+2)]$ with $i \in U^=$. So we have to brute force these bits as well. We also brute force the weird bits in this step. We have

$$\log_2 WF_2 = \frac{5}{3}|U_3| + |U^*| + 3 \cdot |U^=| + w. \quad (7)$$

Combining (6) and (7) we have that the overall work factor for the key recovery is

$$\log_2 WF \approx \max\{\log_2 WF_1, \log_2 WF_2\} =$$
$$\max\left\{\frac{5}{3}(n - |def_U|) - \frac{1}{3}|U_2| - w, \frac{5}{3}|U_3| + |U^*| + 3 \cdot |U^=| + w\right\}. \quad (8)$$

Now it is interesting to point out that a simple brute force approach would just search through a weak key class in the known plaintext scenario and its work factor would be the number of elements given by (4). We define a *gain* of our key recovery procedure over the simple brute force as a difference between (4) and (8), which gives a logarithm of the speed-up factor. Gains for all classes in the case of PRINTcipher–48 are given in Appendix A. For PRINTcipher–96 the largest gain is 27 yielding a speed-up factor of 2^{27} which is quite substantial.

See the example in Appendix B for a detailed work-out of the above computations.

Remark 1. The key recovery procedure in chosen/known plaintext scenario as described above was implemented for PRINTcipher–48 and tested for weak key classes that allow practical time key recovery. We could always correctly and uniquely recover the secret key.

Countermeasures against the Attack. Note that due to existence of the round constant RC_i in the last 6 bits (corresponding to the last two S-Boxes no. $n/3-2$ and $n/3-1$), our invariant projected subsets should not be active in these two S-Boxes. As can be seen from Table 6, for PRINTcipher–48 there exist no def_U that avoids the last *three* S-Boxes. So, as has already been pointed out in [12], spreading out the round constant over the last three S-Boxes (two bits of the constant per S-Box) protects against the attack. Note, however, that this choice is not as obvious as it may seem. For example, a SPONGENT-like solution [6], where S-Boxes 0,1 and 14,15 are used for placing the round constant (or any three of them) does not provide a secure solution, since classes no. 23 and 58 from Table 6 avoid them, providing at least 2^{50} weak keys. However, the "SPONGENT" solution obviously defeats the classes no. 44 and 47 found in [12].

Note that opposed to the 48-bit case, in the case of PRINTcipher–96 there *exist* defining sets that avoid the last three S-Boxes. In fact, there are 28 such sets. So the countermeasure suggested in [12] for PRINTcipher–48 does not work for all families here. Still, there is a collection of combinations of three S-Boxes that cannot be avoided by any defining set. The S-Boxes 0,1,23 is one possible solution among many others. So, in order to defeat the attack, one has to spread out the round counter over these S-Boxes.

Important note: One may argue that the invariant attack in its complete form as described in this paper is of not much value, since there still exist simple countermeasures that defeat all families of weak keys. Note, however, that since our analysis is *complete*, i.e. we have found *all* weak key families, we may argue about security of the cipher against the invariant attack. Whereas having only partial results of [12] it is not possible. In fact, as we have seen above, countermeasures for PRINTcipher–96 that may be hinted from [12] are not effective for many families of weak keys.

4 Related and Future Work

In this paper we have undertaken a complete study of the invariant coset attack initially presented at CRYPTO 2011 by Leander *et al.* By explicitly providing characterization of defining sets of invariant projected subsets and weak key families and classes we were able to recover all families of keys that are weak in the sense of the invariant coset attack. We also showed that both versions of the cipher can be made immune to this attack at no additional cost. The latter conclusion was only possible to make due to completeness of our analysis.

Note also that methods of this paper, such as finding all defining set as per Section 3.2, are interesting on their own. It can be shown that similar approach can be employed in other contexts, e.g. for finding optimal guessing strategies for algebraic cryptanalysis. These directions put the methods in a more general context that deserves further investigation.

Acknowledgements. The first author is supported by the German Science Foundation (DFG) grant BU 630/22-1. The second author is supported in part by the NSF grant CNS-1117936. We would like to thank Gregor Leander for his critical comments that helped to improve the paper considerably. We are also thankful to Mohamed Ahmed Abdelraheem for providing the C implementation of PRINTcipher that was used in the implementation of the attacks. Finally, we thank anonymous referees for their numerous valuable suggestions and remarks.

References

1. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
2. 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)
3. 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)
4. Knudsen, L., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: A Block Cipher for IC-Printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)
5. Yap, H., Khoo, K., Poschmann, A., Henricksen, M.: EPCBC - A Block Cipher Suitable for Electronic Product Code Encryption. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 76–97. Springer, Heidelberg (2011)
6. Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varıcı, K., Verbauwhede, I.: spongent: A Lightweight Hash Function. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 312–325. Springer, Heidelberg (2011)
7. Abdelraheem, M.A., Leander, G., Zenner, E.: Differential Cryptanalysis of Round-Reduced PRINTcipher: Computing Roots of Permutations. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 1–17. Springer, Heidelberg (2011)

8. Ågren, M., Johansson, T.: Linear Cryptanalysis of PRINTCIPHER – Trails and Samples Everywhere. In: Bernstein, D.J., Chatterjee, S. (eds.) INDOCRYPT 2011. LNCS, vol. 7107, pp. 114–133. Springer, Heidelberg (2011)
9. Karakoç, F., Demirci, H., Harmancı, A.E.: Combined Differential and Linear Cryptanalysis of Reduced-Round PRINTCIPHER. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 169–184. Springer, Heidelberg (2012)
10. Bulygin, S., Buchmann, J.: Algebraic Cryptanalysis of the Round-Reduced and Side Channel Analysis of the Full PRINTCipher-48. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 54–75. Springer, Heidelberg (2011)
11. Zhao, X., Wang, T., Guo, S.: Fault Propagate Pattern Based DFA on SPN Structure Block Ciphers using Bitwise Permutation, with Application to PRESENT and PRINTcipher, ePrint, http://eprint.iacr.org/2011/086.pdf
12. Leander, G., Abdelraheem, M.A., AlKhzaimi, H., Zenner, E.: A Cryptanalysis of PRINTCIPHER: The Invariant Subspace Attack. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 206–221. Springer, Heidelberg (2011)
13. William Stein, S., et al.: SAGE Mathematics Software, pp. 593–599. The Sage Development Team (2008), http://www.sagemath.org

A The List of Defining Sets of All Possible Invariant Projected Subsets of PRINTCIPHER–48

Table 6 presents defining sets of all possible invariant projected sets for PRINTcipher–48.

B Example of Computations for Weak Keys Number and Work Factors

Example 1. In this example we would like to provide a detailed workout of the computations discussed in Section 3.3. We work with PRINTcipher–48 and a specific defining set (no. 5 in Table 6):

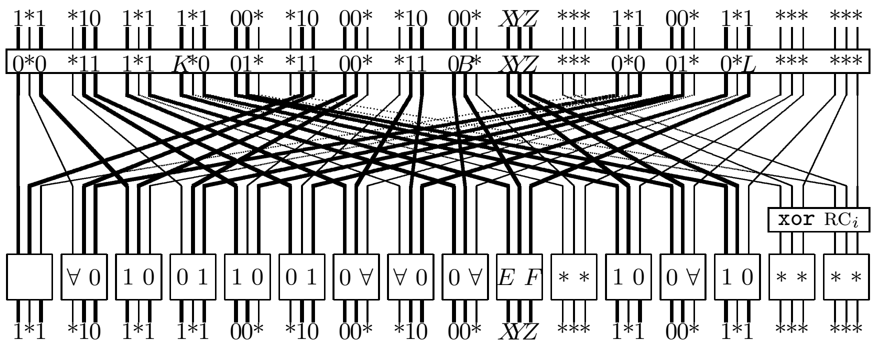
$$def_U = \{0, 2, 4, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 25, 27, 28, 29, 33, 35, 36, 37, 39, 41\}.$$

In the figure below the positions of def_U and their transition through the round is denoted in bold.

Note that inputs and outputs of the S-Boxes in this example are uniquely determined by the set def_U and Table 3. In the figure the symbol $\forall$ means that any value of the corresponding sk_2-bit works for the invariant property. For ABC and XYZ it holds that $SBOX_{EF}(ABC) = XYZ$. Note that XYZ itself can attain arbitrary values and ABC is determined by these and EF bits of the key sk_2. The "passive" bits are denoted with "*" and can attain arbitrary value. We have $|U_2| = 24, |U_3| = 3, |U^*| = 5, |U^=| = 0, w = 0$. The following table makes the situation on the figure in a bit more formal way:

Table 6. Defining sets of invariant projected subsets of PRINTcipher–48
Bold font: families found in [12]
$\#k$: number of elements in the corresponding family
$\log WF$: logarithm of the time complexity (work factor) of key recovery for each class in the family
Gain: logarithm of the speed-up factor of the mixed CP/KP scenario over the simple brute force in KP
$\perp$: an upper bound, not an exact value

No.	def_U	$\#k$	$\log WF$	Gain
1	[0, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 21, 22, 23, 24, 25, 27, 28, 29, 33, 34, 36, 37, 39, 41]	$36^{\perp}$	25	11
2	[0, 1, 3, 4, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 25, 27, 28, 29, 33, 34, 36, 37, 39, 41]	42	26	10
3	[0, 2, 3, 5, 6, 8, 9, 11, 12, 13, 16, 17, 19, 20, 22, 23, 24, 25, 27, 28, 29, 33, 34, 36, 37, 39, 41]	37	27	7
4	[0, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 21, 22, 24, 25, 27, 28, 29, 33, 34, 36, 37, 39, 41]	41	25	10
5	[0, 2, 4, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 25, 27, 28, 29, 33, 35, 36, 37, 39, 41]	37	27	7
6	[0, 2, 3, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 20, 22, 23, 24, 25, 27, 28, 29, 33, 35, 36, 37, 39, 41]	36	27	6
7	[0, 1, 4, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 25, 27, 28, 29, 34, 35, 36, 37, 39, 41]	37	27	7
8	[0, 1, 3, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 20, 22, 23, 24, 25, 27, 28, 29, 34, 35, 36, 37, 39, 41]	37	27	7
9	[0, 2, 3, 5, 6, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 33, 34, 36, 38, 39, 41]	$34^{\perp}$	24	10
10	[0, 2, 3, 5, 7, 8, 9, 11, 12, 13, 16, 17, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 33, 34, 36, 38, 39, 41]	39	24	9
11	[0, 1, 3, 4, 5, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 21, 22, 23, 24, 25, 27, 28, 29, 33, 34, 36, 38, 39, 41]	44	25	10
12	[0, 2, 3, 5, 6, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 20, 21, 22, 24, 25, 27, 28, 29, 33, 34, 36, 38, 39, 41]	$33^{\perp}$	24	9
13	[0, 2, 3, 5, 7, 8, 9, 11, 12, 13, 16, 17, 19, 20, 21, 22, 24, 25, 27, 28, 29, 33, 34, 36, 38, 39, 41]	35	24	8
14	[0, 1, 3, 4, 5, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 21, 22, 24, 25, 27, 28, 29, 33, 34, 36, 38, 39, 41]	40	25	9
15	[0, 1, 3, 4, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 21, 23, 24, 25, 27, 28, 29, 33, 34, 36, 38, 39, 41]	39	26	7
16	[0, 2, 3, 5, 6, 8, 9, 11, 12, 13, 16, 17, 19, 20, 21, 23, 24, 25, 27, 28, 29, 33, 34, 36, 38, 39, 41]	35	24	8
17	[0, 2, 3, 5, 7, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 33, 35, 36, 38, 39, 41]	38	26	6
18	[0, 2, 4, 5, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 21, 22, 23, 24, 25, 27, 28, 29, 33, 35, 36, 38, 39, 41]	38	26	6
19	[0, 2, 3, 5, 7, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 22, 24, 25, 27, 28, 29, 33, 35, 36, 38, 39, 41]	34	26	5
20	[0, 2, 4, 5, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 21, 22, 24, 25, 27, 28, 29, 33, 35, 36, 38, 39, 41]	34	26	5
21	[0, 2, 3, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 23, 24, 25, 27, 28, 29, 33, 35, 36, 38, 39, 41]	35	27	5
22	[0, 2, 4, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 21, 23, 24, 25, 27, 28, 29, 33, 35, 36, 38, 39, 41]	34	27	4
23	[6, 8, 9, 11, 12, 13, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 34, 35, 36, 38, 39, 41]	50	38	6
24	[0, 1, 3, 5, 7, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 34, 35, 36, 38, 39, 41]	40	26	8
25	[0, 1, 4, 5, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 21, 22, 23, 24, 25, 27, 28, 29, 34, 35, 36, 38, 39, 41]	39	26	7
26	[0, 1, 3, 5, 7, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 22, 24, 25, 27, 28, 29, 34, 35, 36, 38, 39, 41]	36	26	7
27	[0, 1, 4, 5, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 21, 22, 24, 25, 27, 28, 29, 34, 35, 36, 38, 39, 41]	35	26	6
28	[0, 1, 3, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 23, 24, 25, 27, 28, 29, 34, 35, 36, 38, 39, 41]	36	27	6
29	[0, 1, 4, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 21, 23, 24, 25, 27, 28, 29, 34, 35, 36, 38, 39, 41]	34	27	4
30	[0, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 33, 34, 37, 38, 39, 41]	$30^{\perp}$	25	11
31	[0, 2, 3, 4, 7, 8, 9, 11, 12, 13, 16, 17, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 33, 34, 37, 38, 39, 41]	41	24	11
32	[0, 1, 3, 4, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 33, 34, 37, 38, 39, 41]	45	25	11
33	[0, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 16, 17, 18, 19, 20, 21, 22, 24, 25, 27, 28, 29, 33, 34, 37, 38, 39, 41]	$35^{\perp}$	24	11
34	[0, 2, 3, 4, 7, 8, 9, 11, 12, 13, 16, 17, 19, 20, 21, 22, 24, 25, 27, 28, 29, 33, 34, 37, 38, 39, 41]	37	24	10
35	[0, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 19, 20, 21, 23, 24, 25, 27, 28, 29, 33, 34, 37, 38, 39, 41]	37	24	10
36	[0, 2, 4, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 33, 35, 37, 38, 39, 41]	40	26	8
37	[0, 2, 3, 4, 7, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 33, 35, 37, 38, 39, 41]	40	26	8
38	[0, 2, 3, 4, 7, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 22, 24, 25, 27, 28, 29, 33, 35, 37, 38, 39, 41]	36	26	7
39	[0, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 23, 24, 25, 27, 28, 29, 33, 35, 37, 38, 39, 41]	37	27	7
40	[0, 1, 4, 5, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 34, 35, 37, 38, 39, 41]	40	26	8
41	[0, 1, 3, 4, 7, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 34, 35, 37, 38, 39, 41]	43	26	11
42	[0, 1, 3, 4, 7, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 22, 24, 25, 27, 28, 29, 34, 35, 37, 38, 39, 41]	39	26	10
43	[0, 1, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, 20, 21, 23, 24, 25, 27, 28, 29, 34, 35, 37, 38, 39, 41]	39	27	9
44	**[0, 1, 4, 5, 12, 13, 16, 17, 24, 25, 28, 29, 36, 37, 40, 41]**	**51**	**48**	**3**
45	[0, 1, 3, 4, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 25, 27, 28, 29, 33, 34, 36, 37, 40, 41]	40	26	8
46	[0, 2, 3, 5, 6, 7, 9, 11, 12, 13, 16, 17, 19, 20, 22, 23, 24, 25, 27, 28, 29, 33, 34, 36, 37, 40, 41]	34	27	4
47	**[0, 1, 3, 4, 5, 9, 11, 12, 13, 16, 17, 24, 25, 27, 28, 29, 33, 35, 36, 37, 40, 41]**	**51**	**38**	**7**
48	[0, 2, 4, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 25, 27, 28, 29, 33, 35, 36, 37, 40, 41]	36	27	6
49	[0, 2, 3, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 20, 22, 23, 24, 25, 27, 28, 29, 33, 35, 36, 37, 40, 41]	35	27	5
50	[0, 1, 4, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 25, 27, 28, 29, 34, 35, 36, 37, 40, 41]	35	27	5
51	[0, 1, 3, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 20, 22, 23, 24, 25, 27, 28, 29, 34, 35, 36, 37, 40, 41]	35	27	5
52	[0, 1, 3, 4, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 19, 21, 23, 24, 25, 27, 28, 29, 33, 34, 36, 38, 40, 41]	38	26	6
53	[0, 2, 3, 5, 6, 7, 9, 11, 12, 13, 16, 17, 19, 20, 21, 23, 24, 25, 27, 28, 29, 33, 34, 36, 38, 40, 41]	33	27	3
54	[0, 2, 3, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 20, 21, 23, 24, 25, 27, 28, 29, 33, 35, 36, 38, 40, 41]	35	27	5
55	[0, 2, 4, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 19, 21, 23, 24, 25, 27, 28, 29, 33, 35, 36, 38, 40, 41]	34	27	4
56	[0, 1, 3, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 20, 21, 23, 24, 25, 27, 28, 29, 34, 35, 36, 38, 40, 41]	35	27	5
57	[0, 1, 4, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 19, 21, 23, 24, 25, 27, 28, 29, 34, 35, 36, 38, 40, 41]	33	27	3
58	[6, 7, 9, 11, 12, 13, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 34, 35, 36, 38, 40, 41]	48	38	4
59	[0, 2, 3, 4, 6, 7, 9, 11, 12, 13, 16, 17, 19, 20, 21, 23, 24, 25, 27, 28, 29, 33, 34, 37, 38, 40, 41]	35	27	5
60	[0, 1, 3, 4, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 33, 34, 37, 38, 40, 41]	43	25	9
61	[0, 2, 3, 4, 6, 7, 9, 11, 12, 13, 16, 17, 18, 20, 21, 23, 24, 25, 27, 28, 29, 33, 35, 37, 38, 40, 41]	37	27	7
62	[0, 2, 4, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 33, 35, 37, 38, 40, 41]	39	26	7
63	[0, 1, 3, 4, 6, 7, 9, 11, 12, 13, 16, 17, 18, 20, 21, 23, 24, 25, 27, 28, 29, 34, 35, 37, 38, 40, 41]	38	27	8
64	[0, 1, 4, 5, 6, 7, 9, 11, 12, 13, 16, 17, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 34, 35, 37, 38, 40, 41]	38	26	6


1*1 *10 1*1 1*1 00* *10 00* *10 00* XYZ *** 1*1 00* 1*1 *** ***
0*0 *11 1*1 K*0 01* *11 00* *11 0B* XYZ *** 0*0 01* 0*L *** ***
xor RC$_i$
∀ 0 | 1 0 | 0 1 | 1 0 | 0 1 | 0 ∀ | ∀ 0 | 0 ∀ | E F | * * | 1 0 | 0 ∀ | 1 0 | * * | * *
1*1 *10 1*1 1*1 00* *10 00* *10 00* XYZ *** 1*1 00* 1*1 *** ***

S-Box	in_mask	out_mask	U^*	(a_0, a_1)	in_vals	out_vals
0	$++-$	$+-+$	0	(1,0)	10*	1*1
1	$-++$	$-++$	1	(∀,0)	*11	*10
2	$++-$	$+-+$	0	(1,0)	10*	1*1
3	$-++$	$+-+$	0	(0,1)	*01	1*1
4	$+-+$	$++-$	0	(1,0)	0*0	00*
5	$+-+$	$-++$	0	(0,1)	1*1	*10
6	$++-$	$++-$	1	(0,∀)	00*	00*
7	$-++$	$-++$	1	(∀,0)	*11	*10
8	$++-$	$++-$	1	(0,*)	00*	00*
9	$+++$	$+++$	0	(E,F)	ABC	XYZ
10	$---$	$---$	0	(*,*)	***	***
11	$++-$	$+-+$	0	(1,0)	10*	1*1
12	$++-$	$++-$	1	(0,∀)	00*	00*
13	$++-$	$+-+$	0	(1,0)	10*	1*1
14	$---$	$---$	0	(*,*)	***	***
15	$---$	$---$	0	(*,*)	***	***
Sum			5			

Now, let us see how the values from the table above are distributed in one round:

$in =$	1*1 *10 1*1 1*1 00* *10 00* *10 00* XYZ *** 1*1 00* 1*1 *** ***
$sk_1 =$	0*0 *11 1*1 K*0 01* *11 00* *11 0B* XYZ *** 0*0 01* 0*L *** ***
$XOR =$	1*1 *01 0*0 A*1 01* *01 00* *01 0B* 000 *** 1*1 01* 1*C *** ***
$P =$	10* *11 10* *01 0*0 1*1 00* *11 00* ABC *** 10* 00* 10* *** ***
$sk_2 =$	10 ∀0 10 01 10 01 0∀ ∀0 0∀ EF ** 10 0∀ 10 ** **
$out =$	1*1 *10 1*1 1*1 00* *10 00* *10 00* XYZ *** 1*1 00* 1*1 *** ***

In this table the values in *out* and *in* are the same and are taken from the *out_vals* column of the first table. Then, the values in P correspond to the inputs to the S-Boxes (or, equivalently, to the outputs of the diffusion layer) and are taken from the column *in_vals* of the first table. We then permute the values in P with P^{-1} to get the output of the XOR_{sk_1} operation. Having that, we may compute many values of sk_1 right away. Note that $K = 1 + A, L = 1 + C$ and are determined by the values of XYZ and EF.

For a specific example, let us take $XYZ = 000$ so that $v_3 = (0, 0, 0)$. So we are working now with

$$U = 1{*}1\ {*}10\ 1{*}1\ 1{*}1\ 00{*}\ {*}10\ 00{*}\ {*}10\ 00{*}\ 000\ {*}{*}{*}\ 1{*}1\ 00{*}\ 1{*}1\ {*}{*}{*}\ {*}{*}{*}.$$

Note that independently of the values of EF we have $SBOX^{-1}_{EF}(000) = 000 = ABC$ and so $K = L = 1$. The weak keys from $WK(def_U, v_3)$ are of the form (sk_1, sk_2), where

$$sk_1 = 0{*}0\ {*}11\ 1{*}1\ 1{*}0\ 01{*}\ {*}11\ 00{*}\ {*}11\ 00{*}\ 000\ {*}{*}{*}\ 0{*}0\ 01{*}\ 0{*}1\ {*}{*}{*}\ {*}{*}{*},$$

$$sk_2 = 10\quad {*}0\quad 10\quad 01\quad 10\quad 01\quad 0{*}\quad {*}0\quad 0{*}\quad {*}{*}\quad {*}{*}\quad 10\quad 0{*}\quad 10\quad {*}{*}\quad {*}{*}.$$

Now let us compute the number of elements in $WK(def_U, 0)$. Using (4) we have

$$|WK(def_U, 0)| = 2^{80-27-24+5} = 2^{34}.$$

The upper bound provided by (3) is actually tight in this case and

$$\Big| \bigcup_{v_3 \in \mathbb{F}_2^3} WK(def_U, v_3) \Big| = 2^3 \cdot 2^{34} = 2^{37}.$$

Now as to the time complexity of the key recovery, from (8) we have

$$\log_2 WF = \max\left\{ \frac{5}{3} \cdot 21 - \frac{1}{3} \cdot 24, \frac{5}{3} \cdot 3 + 5 \right\} = \max\{27, 10\} = 27.$$

So for the key recovery it takes around 2^{27} encryptions having 3 chosen and 1 known plaintext. We have a *gain* of $34 - 27 = 7$ bits compared to the simple brute force attack.

It is not hard to see that the class of weak keys $WK(def_U, 0)$ is different from the ones presented in [12]. Indeed, for example the keys with

$$sk_1 = 0\underline{0}0\ {*}11\ 1{*}1\ 1{*}0\ 01{*}\ {*}11\ 00{*}\ {*}11\ 00{*}\ 000\ {*}{*}{*}\ 0{*}0\ 01{*}\ 0{*}1\ {*}{*}{*}\ {*}{*}{*}$$

do not belong to the class defined by def_U no. 44, since there $sk_1[1] = 1$ and the keys with

$$sk_1 = 0{*}0\ {*}11\ 1{*}1\ 1{*}0\ 01{*}\ {*}11\ 00{*}\ {*}11\ 00{*}\ 000\ {*}{*}{*}\ 0{*}0\ 01{*}\ 0\underline{0}1\ {*}{*}{*}\ {*}{*}{*}$$

do not belong to the class defined by def_U no. 47, since there $sk_1[40] = 1$, see [12].

Applying Remote Side-Channel Analysis Attacks on a Security-Enabled NFC Tag

Thomas Korak and Thomas Plos

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
{thomas.korak,thomas.plos}@iaik.tugraz.at

Abstract. The number of applications that rely on near-field communication (NFC) technology is significantly growing. Especially for security-related applications, short communication ranges as they are provided by NFC systems are advantageous to minimize the risk of eavesdropping. In this work we show that although the communication range of NFC systems is limited to several centimeters, side-channel information modulated on the reader signal can be measured at much larger distances. We name the side-channel information modulated on the reader signal *parasitic load modulation*. By measuring the *parasitic load modulation* of a tag, so-called remote side-channel analysis (SCA) attacks can be applied. We verify the practicability of such remote attacks by analyzing a security-enabled NFC tag with an integrated Advanced Encryption Standard (AES) module. The analyzed NFC tag operates at a carrier frequency of 13.56 MHz and uses the well known ISO 14443A communication standard. We were able to conduct successful remote SCA attacks at distances up to 1 m. No special measurement equipment is required, a self-made loop antenna, a broadband amplifier, and an oscilloscope are sufficient. We further formulate a relationship between attack performance and measurement distance that is confirmed by our practical results. These are the first remote SCA attacks on an NFC tag and on tags operating in the high-frequency range at 13.56 MHz at all. The results emphasize that the integration of suitable SCA countermeasures is inevitable.

Keywords: remote side-channel analysis (SCA) attacks, differential electromagnetic analysis (DEMA) attacks, near-field communication (NFC), radio-frequency identification (RFID), Advanced Encryption Standard (AES).

1 Introduction

Near-field communication (NFC) is a contactless communication technology that has become increasingly important in the last years. A variety of applications make already use of NFC technology, e.g., ticketing, transportation, mobile payment, smart posters or access-control systems. The integration of NFC functionality into the latest generation of smart phones underlines the relevance of this technology and will further push its spreading. With the availability of NFC-enabled devices in large scale, new applications such as the future *Internet of Things* (IoT) will arise.

NFC technology bases on standards and platforms used by radio-frequency identification (RFID) systems such as ISO 14443-3/4 [12, 13] as well as on specifications released by the NFC Forum [23]. In a basic NFC system, a reader device (e.g., a smart phone) generates a radio frequency (RF) field that is used for contactless communication with a so-called tag[1]. NFC systems have a rather short communication range of only a few centimeters and their RF field operates in the high-frequency (HF) range at 13.56 MHz. The tag is a small microchip attached to an antenna. Especially passive tags that also receive their power supply from the RF field are interesting for applications like the future IoT, where tags have to be available in large quantities and at a competitive price.

Due to the limitations of low-cost tags in terms of power supply (passive operation) and chip size (directly influences production costs), manufacturers initially integrated only proprietary cryptographic algorithms into tags for enabling security functionality. However, the incidents of the last years (e.g., CRYPTO 1 algorithm [24], DST40 cipher [2], Hitag 2 cipher [5]) have shown that proprietary algorithms are not suitable for providing sufficient security for NFC or RFID systems. This has led to a paradigm shift, where manufacturers started integrating cryptographic algorithms with state-of-the-art security to low-cost tags, like for example the Advanced Encryption Standard (AES) [22].

Even if a cryptographic algorithm is mathematically secure, its implementation on a device can be vulnerable to side-channel analysis (SCA) attacks. Such attacks measure side-channel information (e.g., computation time, power consumption, electromagnetic emanation) of the device while it is computing the cryptographic algorithm. With the measured side-channel information, secret data involved in the algorithm computation (e.g., the encryption key) can be extracted. Especially differential power analysis (DPA) attacks invented by Kocher et al. [17] in 1999 are a very powerful technique, as they need no detailed knowledge about the attacked device and allow to detect even very weak data dependencies in the power consumption. As shown by Gandolfi et al. [7], also the electromagnetic (EM) emanation of a device can be deployed for such attacks, which are called then differential EM analysis (DEMA) attacks.

Over the years numerous articles have been published that present DPA or DEMA attacks on contact-based devices [1, 4, 8, 14, 19, 20, 26]. However, only a handful of articles deals with attacks on contactless devices like NFC or RFID tags. DPA and DEMA attacks on HF tags operating at 13.56 MHz have been presented by Hutter et al. [9], Kasper et al. [15], and Oswald et al. [27]. Attacks on an NFC tag have been shown by Korak et al. [18]. All these attacks have been conducted in close proximity of the tag. So-called remote SCA attacks that can be applied from distance have been first demonstrated by Oren and

[1] Note that NFC technology supports also a so-called peer-to-peer mode where both devices are actively powered and use their own RF field for communication, which we do not consider here in our work.

Shamir [25] for tags operating in the ultra-high frequency (UHF) range around 860 to 960 MHz and later also by Plos [28].

UHF tags use different communication principles than HF and NFC tags. One difference is that UHF tags operate in the far field using electromagnetic waves, allowing them to achieve communication ranges of 10 m and more. HF and NFC tags on the other hand operate in the near field using inductive coupling, leading to much shorter communication ranges (up to 1.5 m for HF tags and up to 10 cm for NFC tags). Moreover, UHF tags use backscatter modulation for data transmission to the reader, HF and NFC tags use load modulation. Observations of Oren and Shamir have shown that the backscatter of UHF tags also contains data-dependent information—named *parasitic backscatter* by them—that enables remote SCA attacks.

In this work we practically demonstrate that NFC tags that have only a communication range of several centimeters are also vulnerable to remote SCA attacks, similar to UHF tags. We show that an attacker can easily measure data-dependent information modulated on the 13.56 MHz RF signal at distances up to 1 m. We name the data-dependent information present in the RF signal *parasitic load modulation*. This is the first article that presents results of successful remote SCA attacks on tags in the HF range. For our analyses we use a security-enabled NFC tag that is passively powered and that has an AES module integrated. By measuring the *parasitic load modulation* of the NFC tag, we successfully revealed the secret key of the AES module via remote SCA attacks at distances from 25 cm up to 1 m. For our attacks no extra analog preprocessing circuits like amplitude demodulator or hardware filter are required as they are used for example in the close-proximity measurements in [15, 27]. Instead we simply measure the peaks of the reader signal that contain the *parasitic load modulation* followed by a downsampling step in software. Our measurement setup only consists of a self-made loop antenna, an RF amplifier, and an oscilloscope. This makes our attack very easy to apply, even for people without detailed RFID/NFC knowledge. Our results let us come to the conclusion that NFC tags (and probably HF tags in general) are vulnerable to remote SCA attacks, making the integration of proper countermeasures inevitable for such tags.

The remainder of this article is structured as follows. Section 2 describes the remote SCA attacks and Section 3 provides details about the analyzed NFC tag. Measurement setup and preprocessing steps are explained in Section 4. The results of the remote SCA attacks are presented in Section 5, followed by a summary and discussion in Section 6. Conclusions are drawn in Section 7.

2 Remote SCA Attacks

The remote SCA attacks presented in this paper have their origin in the DPA attacks first demonstrated by Kocher et al. [17] in 1999. DPA attacks take advantage of the fact that different data processed on complementary metal-oxide semiconductor (CMOS) devices result in different power-consumption values. However, not only the power consumption of a device is a suitable side channel, but also the EM emanation as illustrated by Gandolfi et al. [7]. DPA attacks that use the EM emanations of a device are called DEMA attacks.

For conducting DPA or DEMA attacks, a large number of power or EM traces has to be recorded in a first step that target the same operation of the cryptographic device but with different, known input data. In the second step, statistical methods (e.g., the Pearson correlation coefficient) are applied on the recorded traces together with modeled side-channel information of the device in order to reveal secret information (e.g., the encryption key used by the device). For a detailed description of DPA and DEMA attacks we refer to the book of Mangard et al. [21].

The power consumption of a device needed for a DPA attack is typically determined by measuring the voltage drop across a resistor that is inserted in the supply or ground line of the device. For contactless devices like NFC and RFID tags, inserting a resistor for directly measuring the power consumption is often not possible. Hence, special antennas or probes are used to gather the EM emissions of the tag chip in close proximity that can be used afterwards for DEMA attacks [9, 15, 18, 27].

An important aspect that has to be considered when measuring the EM emissions of a tag chip is the presence of the RF signal of the reader. The reader signal is much stronger than the side-channel information emitted by the tag. In order to overcome this problem, the tag itself can be modified or analog preprocessing circuits can be used to minimize the influence of the reader signal. Carluccio et al. [3] suggested for example to separate the tag chip from its antenna. By connecting the tag chip to an external antenna via wires, it is possible to place the tag chip away from the reader field, lowering the impact of the field on the measurements. Kasper et al. [15] use an analog preprocessing circuit that subtracts the reader signal from the signal measured with an EM probe close to the tag chip. Oswald et al. [27] apply an analog demodulation circuit to suppress the influence of the reader signal.

For our remote SCA attacks we use a different approach that requires neither the separation of the tag chip from its antenna nor the application of special analog preprocessing circuits. We even go the opposite way by exploiting the strong reader field as a carrier of the weak data-dependent information emitted by the tag. As we assume that most of the data-dependent information is amplitude modulated on the reader signal, it is sufficient to measure only the peaks of the reader signal. This simple measurement concept was originally used for analyzing the emissions of UHF tags [29]. In this work we show that this concept is also suitable for gathering the data-dependent emissions of NFC/HF tags, even at greater distances.

NFC and HF RFID systems are inductively coupled. This means that the antennas of reader and tag are loosely coupled and act like an air-core transformer [6], which is illustrated in Figure 1. Applying an alternating voltage at the reader antenna (U_{Reader}) results in a magnetic field that itself induces an alternating voltage at the tag antenna (U_{Tag}). The voltage at the tag is not only used for data transmission from the reader to the tag (i.e., by modulating the

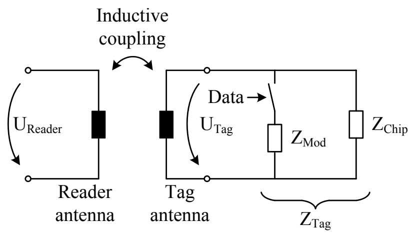


Fig. 1. Basic principle of inductive coupling between reader and tag antenna

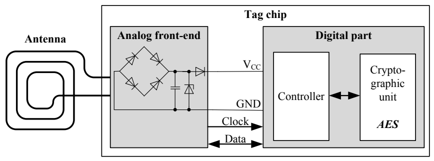


Fig. 2. Architecture of the evaluated chip

reader field in step with the data), but also to provide the power supply for passive tags. Data transmission from the tag to the reader is done by so-called load modulation, where an impedance Z_{Mod} is switched in step with the data. Switching the impedance Z_{Mod} changes also the overall impedance of the tag Z_{Tag}, which in turn results in changes in the magnetic field and in detectable voltage variations at the reader antenna.

However, not only the intended load modulation influences the magnetic field, but also changes in the power consumption of the tag chip. As the power consumption directly relates to the chip's effective impedance Z_{Chip}, the overall impedance of the tag is changed as well. In that way data-dependent information present in the power consumption of the tag chip is modulated on the reader field. We call this effect *parasitic load modulation*, according to a similar effect named *parasitic backscatter* that was observed by Oren and Shamir for tags operating in the UHF range [25]. In the following, we use this *parasitic load modulation* for conducting remote SCA attacks on a security-enabled NFC tag.

It is not obvious that remote attacks are applicable on NFC or HF tags. First, NFC and HF tags are inductively coupled and operate in the near field where RF signals are attenuated with $1/d^3$ (with d being the distance). UHF tags on the other hand operate in the far field, where RF signals are only attenuated with $1/d$. Moreover, the *parasitic backscatter* observed by Oren and Shamir does not influence the reader field, rather it relates to independent electromagnetic waves emitted by the tag antenna. Consequently, a favorable placement of the measurement antenna is possible that allows to gather mainly the signal emitted by the tag antenna. However, this is not possible when measuring the *parasitic load modulation* of a tag, as it is directly modulated on the strong reader field.

3 Device under Attack

In this section we give an overview of the attacked hardware device. For the evaluation we use a prototype of an NFC-tag chip. Mounted on an antenna, the chip behaves like a commercial, passive RFID tag. The NFC tag operates in the HF range at a frequency of 13.56 MHz for communication with the reader. The communication protocol is implemented according to the ISO 14443-3/4

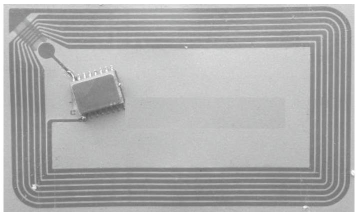


Fig. 3. The NFC-tag chip mounted on a commercial RFID-tag antenna

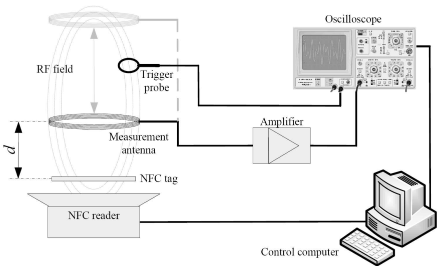


Fig. 4. Schematic measurement setup

standard [12, 13]. The chip consists of two main parts as it can be seen in Figure 2: the analog front-end (AFE) and the digital part. The antenna is connected to the AFE that extracts the power supply and the clock signal from the reader field and provides them to the digital part. The digital part is responsible for processing the commands and for sending proper responses to the reader. This part also contains a cryptographic unit with an AES implementation to provide security services like tag authentication. In order to authenticate the NFC tag to a reader device the *Internal Authenticate (IntAuth)* command is used according to [10]. The *IntAuth* command sent from the reader to the tag includes an eight-byte long challenge. The response of the tag to that command is a *waiting-time extension (WTX)* in order to signalize the reader that the requested computation will need more time. The reader has to acknowledge the received *WTX* command by sending the same command back to the tag. After the tag has received the *WTX* again it generates an eight-byte long random number, concatenates it to the previously received challenge and encrypts the sixteen bytes (plaintext) using the AES. The last step of the *IntAuth* procedure consists in sending the result of the AES computation (the ciphertext) to the reader. The AES part is implemented as special-purpose hardware to meet the most important requirements for RFID-tag chips: low power consumption and small chip area. Low power consumption is a requirement because the chip uses the power supply generated from the reader field. Chip area is an important factor concerning the production costs.

We used the antenna of a commercial HF-RFID tag. We removed the original chip from this antenna and mounted our NFC-tag chip on it instead as it can be seen in Figure 3. The chip is placed in a metal housing, so its direct emanation is attenuated.

4 Measurement Setup and Preprocessing

In this section we explain our used measurement setup as well as the preprocessing steps that we have performed on the recorded traces. One important remark is that our setup does not require any analogue preprocessing circuits

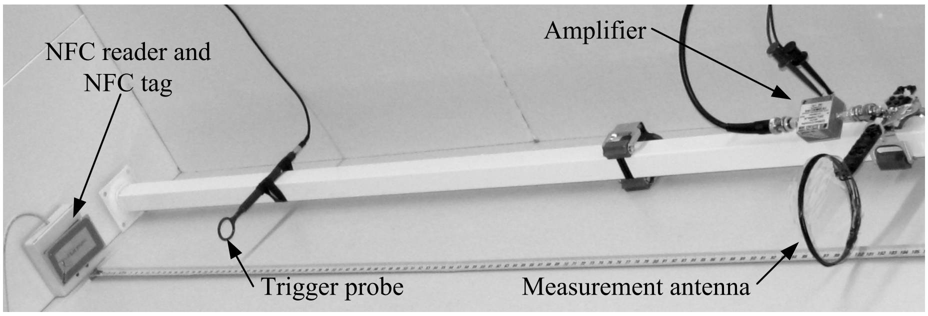


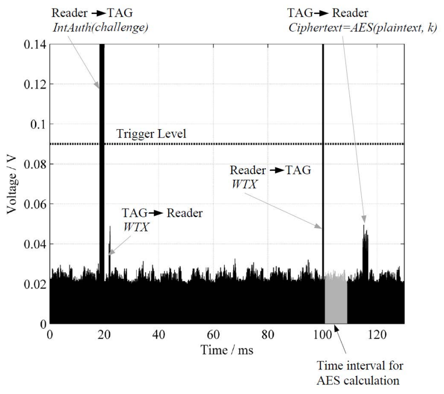
Fig. 5. Measuring arrangement for $d = 100\,cm$

like a signal-cancellation circuit (c.f. [15]) or a demodulation circuit (c.f. [27]). We only use an amplifier for larger distances between tag and measurement antenna in order to increase the amplitude of the measured signal.

In Figure 4 the schematic measurement setup is depicted and Figure 5 shows a picture taken during a measurement at a distance of $100\,cm$. An NFC reader (Tagnology TagScan) as well as an oscilloscope (LeCroy LC584) are connected to the control computer. This computer stores the traces and sends commands to the NFC tag using MATLAB scripts. Furthermore we use a self-made loop antenna with $N_{ant} = 5$ windings and a diameter of $d_{ant} = 8\,cm$ in order to measure the reader signal. We have tried different values for N_{ant} as well as for d_{ant}, but the best results could be achieved with the values mentioned above. In order to amplify the signal measured with the antenna we used a broadband amplifier with a gain of $30\,dB$.

We do not need a dedicated trigger pin on the device under attack. We extract the trigger information for the start of the recording step using a pattern in the communication between NFC tag and reader as it can be seen in Figure 6. This figure shows an EM trace of the whole *IntAuth* procedure as described in the previous section. The time interval where the AES calculation takes place is highlighted in the trace. This part was recorded in order to perform the DEMA attacks. For recording the traces the trigger probe was placed at a distance of $25\,cm$. In order to reduce the effort we have fixed the trigger probe at $d = 25\,cm$ for all measurements. However, as Figure 6 illustrates, commands sent from the reader to the tag that are used for triggering can be clearly identified in the trace (tag answers are smaller but can also be easily identified). Hence, placing the trigger probe at larger distances is no problem. Experiments yielded that the trigger information can be easily detected at distances exceeding $100\,cm$.

We used a special recording technique in order to get rid of analog preprocessing circuits. Our assumption is that the data-dependent information is modulated on the amplitude of the reader signal due to *parasitic load modulation*. So we just recorded the peaks of the reader signal to increase the resolution of the measurement. The upper plot in Figure 7 shows one trace where the whole



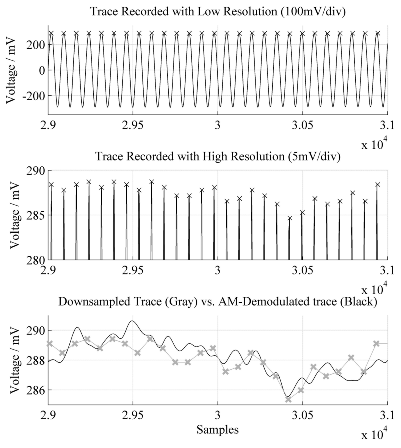


Fig. 6. Sequence of commands for the internal authentication using AES. The time interval where the AES calculation takes place is highlighted in the trace.

Fig. 7. The plot on top shows an EM trace recorded with low resolution. The plot in the middle shows the same trace recorded with a high resolution. The plot on the bottom shows a comparison between a downsampled trace and an AM-demodulated trace.

amplitude is recorded and the lower plot shows the same trace zoomed into the peaks. Before storing the traces we performed a downsampling step. The reader frequency $f_c = 13.56\, MHz$ as well as the used sampling rate $sr = 1\, GS/s$ are known, so the downsampling rate ds can be calculated using Equation 1.

$$ds = \frac{sr}{f_c} = \frac{1GS/s}{13.56MHz} = 73.75\, [Samples] \qquad (1)$$

The value of ds has to be an integer value. In order to achieve this we have used an adaptive downsampling rate. After using $ds = 74$ three times, $ds = 73$ was used one time and so on ($74 + 74 + 74 + 73 = 4 \cdot 73.75$). Downsampling using the mentioned rate means we only use one point per period $T_c = 1/f_c = 1/13.56\, MHz = 73.75\, ns$ where the point equals the maximum value in this period. Using this method we could achieve a memory saving of a factor of 74. This method seems to be similar to an analog demodulation followed by a lowpass filtering. To illustrate the similarity we have applied these two steps on the trace from Figure 7. The two steps were performed in MATLAB. Using the build-in function *amdemod* we demodulated the trace. Afterwards the resulting trace was lowpass-filtered using the function $filtfilt$ with a cut-off frequency $f_{co} = 15\, MHz$. Except of the downsampling step no other preprocessing steps (e.g., alignment, filtering) are required in order to perform successful DEMA attacks.

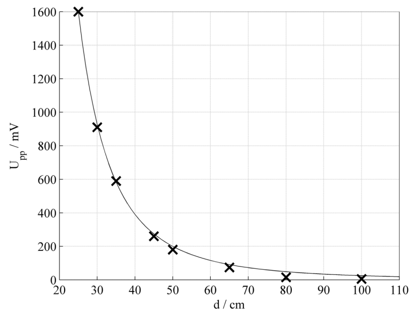


Fig. 8. Peak-to-peak voltage (U_{pp}) of the reader signal as a function of the distance between reader and measurement antenna ($U_{pp} = f(d)$)

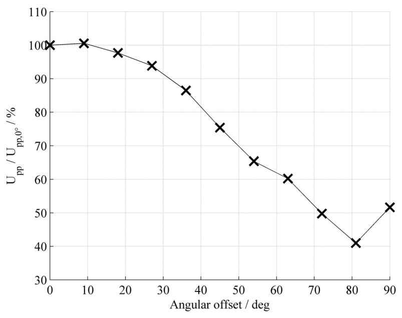


Fig. 9. Peak-to-peak voltage (U_{pp}) of the reader signal as a function of the angular offset ($U_{pp} = f(angular\,offset)$)

Next the focus is put on the relationship between the amplitude of the measured reader signal U_{pp} and the distance d between reader and measurement antenna. According to [6] this relationship can be described with the following equation: $U_{pp} \approx \frac{1}{d^3}$. The measurements of the amplitude of the reader signal at distances d between $25\,cm$ and $100\,cm$ confirmed the theory. Figure 8 illustrates the performed comparison of measured values and values calculated based on the equation given above. The black graph corresponds to values calculated using the equation and the data points marked with 'x' correspond to measured values.

Furthermore the influence of the angular offset of the measurement antenna was examined. For that purpose U_{pp} was measured for angles between 0° (i.e., reader antenna and measurement antenna are coaxial as shown in Figures 4 and 5) and 90° with a step size of 9° at a constant distance $d = 30\,cm$. The results are summarized in Table 1. Figure 9 depicts the relationship between angular offset and voltage where the values of the y-axis are normalized to U_{pp} at 0° ($U_{pp,0°}$). At 90° the U_{pp} value can be increased from 470 mV to 618 mV by rotating the measurement antenna by 90° (i.e., reader antenna and measurement antenna are coplanar). The influence of the angular offset highly depends on the antenna design of the used reader.

Table 1. Relation between U_{pp} and angle for $d = 30\,cm$

Angle	U_{pp}		*Angle*	U_{pp}		*Angle*	U_{pp}	
deg	*mV*	%	*deg*	*mV*	%	*deg*	*mV*	%
0	910	100	36	787	86	72	453	50
9	915	101	45	686	75	81	373	41
18	889	98	54	595	65	90	470	51
27	854	94	63	548	60			

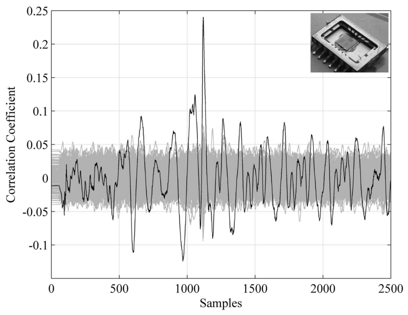


Fig. 10. DEMA-attack result with opened chip housing

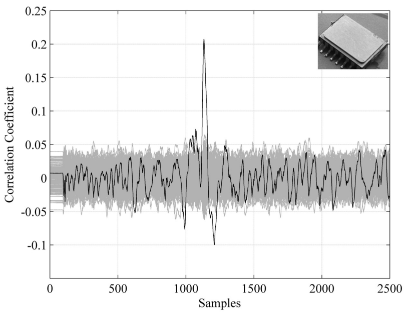


Fig. 11. DEMA-attack result with closed chip housing

In the following we explain how to take advantage of the zoom of the oscilloscope. We have recorded traces with different resolutions and have counted the different occurring voltage values. With the lowest resolution ($100\,mV/div$) many samples have a similar voltage value as only 8 different voltage values appear. With increasing resolution also the number of different voltage values increases. As a result of only measuring the peaks of the signal the voltage values are fine grained and the quality of the DEMA attack increases.

In the following we will use f_{zoom} (zoom factor) in order to describe the percentage of the amplitude recorded with a given U_{pp} (dependent on d) and a given resolution res (V/div) according to the setting of the oscilloscope. $fullscale$ equals the number of divisions on the voltage axis of the oscilloscope. f_{zoom} is calculated according to Equation 2.

$$f_{zoom} = \frac{fullscale \cdot res}{U_{pp}} \cdot 100\,[\%] \tag{2}$$

5 Side-Channel Analysis Results

In this section the results of the remote SCA-attacks using EM traces measured at distances between $25\,cm$ and $100\,cm$ are presented. At the beginning a verification of our assumption that the EM emanation of the chip is modulated on the reader signal (*parasitic load modulation*) is given.

5.1 Verification of the Parasitic Load Modulation

In order to verify the assumption that the EM emanation of the chip is modulated on the reader signal and that it is not the direct emanation of the chip, we first performed attacks at a low distance of only $7\,cm$ for two scenarios. In the first scenario the chip housing was opened and in the second scenario it was closed. 20 sets each containing 5000 EM traces were recorded for each scenario and a

DEMA attack was performed on each set. Next the mean ($\bar{\rho}_{opened}$, $\bar{\rho}_{closed}$) and the standard deviation (σ_{opened}, σ_{closed}) of the highest correlation values were calculated for both scenarios yielding to the following results: $\bar{\rho}_{opened} = 0.246$, $\sigma_{opened} = 0.032$, $\bar{\rho}_{closed} = 0.244$, $\sigma_{closed} = 0.025$. Figure 10 shows one DEMA-attack result for the scenario with opened chip housing and Figure 11 shows one DEMA-attack result for the scenario with closed chip housing. Gray traces correspond to wrong key hypotheses and black traces correspond to the correct key hypothesis. Small variations in the highest correlation values can be observed but the statistical analysis yield that the direct emanation of the chip does not influence the results significantly.

5.2 Remote SCA-Attack Results

In the following the results of the remote DEMA attacks for high distances are presented. We have performed attacks for the following distances: $25\,cm$, $35\,cm$, $45\,cm$, $50\,cm$, $65\,cm$, $80\,cm$, and $100\,cm$. In order to evaluate the attack performance for every distance we used the *nearest-rival distinguishing power (nrdp)* measure as presented in [30]. This value is calculated according to Equation 3. $\rho_{maxCorrect}$ is the correlation value of the correct key hypothesis, $\rho_{maxWrong}$ is the maximum correlation value of the wrong key hypotheses and σ relates to the standard deviation. The number of traces used for the attack is N. If $nrdp > 0$ the attack is successful, else ($nrdp \leq 0$) the correct key hypothesis cannot be distinguished from wrong key hypotheses.

$$nrdp = \frac{1}{\sigma} \cdot (\rho_{maxCorrect} - \rho_{maxWrong}); \quad \sigma = \frac{1}{\sqrt{N}} \tag{3}$$

In order to evaluate the influence of the used resolution on the attack result, attacks with different resolutions were performed for $d = 35\,cm$. The result of this experiment is depicted in the upper plot in Figure 12. The left-most data-point corresponds to the highest possible resolution which can be achieved with the used oscilloscope, namely $2\,mV/div$. With $U_{pp} = 590\,mV$ for this distance $3.4\,\%$ of the amplitude are recorded as a consequence ($f_{zoom} = 3.4\,\%$). The important observation here is that the highest resolution does not lead to the attack with the highest correlation coefficient. Using $f_{zoom} = 8.5\,\%$ leads to the best results. In the case of $d = 35\,cm$ this corresponds to a resolution of $5\,mV/div$. In the lower plot in Figure 12 the used values for f_{zoom} for the different distances are depicted. The high f_{zoom} value of over $25\,\%$ for $d = 65\,cm$ appears because with the highest resolution of the oscilloscope ($2\,mV/div$) and the U_{pp} value for that distance a smaller value was not achievable. In order to achieve f_{zoom} values in the region around $10\,\%$ for higher distances (smaller U_{pp} values), a second amplifier stage was used to increase the gain. With this modification f_{zoom} values around $10\,\%$ can be reached again at $d = 80\,cm$ and $d = 100\,cm$.

The plot in Figure 13 shows the correlation coefficient for the correct key hypothesis for the analyzed distances. There is a big descent for the value of ρ between 25 cm and 45 cm: $\Delta\rho_{25cm-45cm} = \rho_{25cm} - \rho_{45cm} = 0.12938$. The

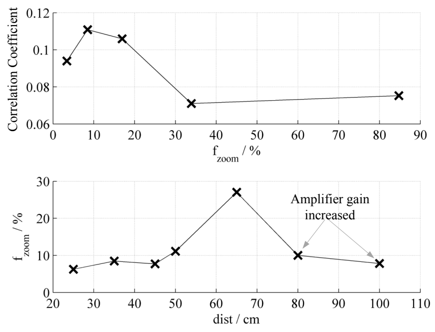


Fig. 12. Upper plot: correlation coefficient as function of f_{zoom} ($d = 35\,cm$). Lower plot: used f_{zoom} values for analyzed distances.

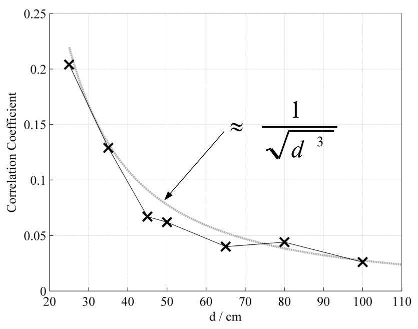


Fig. 13. The resulting correlation coefficient ρ for the analyzed distances

difference of the ρ values between 45 cm and 100 cm is comparatively small: $\Delta\rho_{45cm-100cm} = \rho_{45cm} - \rho_{100cm} = 0.04897$.

Comparing Figure 8 and Figure 13 shows that the relations $U_{pp} \leftrightarrow d$ and $\rho \leftrightarrow d$ are similar. This similarity can be described as follows: The power consumption and as a consequence also the EM emanation of a device at each point in time depends on a noise part P_{noise}, a constant part P_{const} and the exploitable part P_{exp} as explained in the book of Mangard et al. [21]. The total power consumption in every point in time is the sum of these three parts according to Equation 4.

$$P_{total} = P_{noise} + P_{const} + P_{exp} \tag{4}$$

With the information given above the signal-to-noise ratio SNR can be calculated. The SNR is defined as the ratio between the variance of the signal and the variance of the noise. As $Var(P_{const}) = 0$, the SNR can be calculated using Equation 5.

$$SNR = \frac{Var(P_{exp})}{Var(P_{noise})} \tag{5}$$

The higher the SNR at the targeted point in time for an attack is, the better are the results of the correlation attack (higher correlation value ρ). In [21] the relation between ρ and SNR is given according to Equation 6. One important remark is that the approximation given in Equation 6 is only valid for small values of SNR and for $\mid \rho \leq 0.2 \mid$. For the scenario presented in this work these limitations hold.

$$\rho \approx \sqrt{SNR} \tag{6}$$

In a next step relations between d and $Var(P_{exp})$ as well as between d and $Var(P_{noise})$ have to be found. In our model P_{exp} can be seen as the exploitable part of the EM signal of the chip which is modulated on the reader signal. As we could show in the previous section, the relation between U_{pp} of the reader signal and d is the following: $U_{pp} \approx \frac{1}{d^3}$. As a result also the variations caused by P_{exp} decrease with the factor $\frac{1}{d^3}$. So the first observation is that $Var(P_{exp}) \approx \frac{1}{d^3}$. Next the focus is put on P_{noise}. In our remote scenario P_{noise} can be split up into two parts, $P_{noiseIC}$ and $P_{noiseENV}$. $P_{noiseIC}$ is the noise part contributed from the chip and $P_{noiseENV}$ is the environmental noise recorded with the antenna. $P_{noiseENV}$ is independent of d and $P_{noiseIC} \approx \frac{1}{d^3}$ and it can furthermore be assumed that $P_{noiseENV} \gg P_{noiseIC}$. Combining the upper results the relation between SNR and d and as a consequence also the relation between ρ and d (using Equation 6) can be given according to Equation 7. This theoretical result confirms our practical measurements (cf. Figure 13).

$$SNR \approx \frac{1}{d^3} \rightarrow \rho \approx \sqrt{SNR} \approx \frac{1}{\sqrt{d^3}} \tag{7}$$

6 Summary of the Results and Discussion

Table 2 provides an overview of the results achieved with the remote SCA attacks for distances between 25 cm and 100 cm. The values for the *nearest-rival distinguishing power (nrdp)* show that the attacks for all distances lead to correct results. The correlation coefficient $\rho_{maxCorrect}$ decreases according to Equation 7 with increasing distance. As a result the number of traces used for the attack in order to achieve correct results increases. The starting point of the analyses was $d = 25\,cm$ and the distance was increased as soon as the SCA-attack result was expressive. This leads to the different number of traces for the different attack distances. Using a different number of traces decreases the comparability between the attacks on the one hand. On the other hand the achieved results are sufficient in order to confirm the theoretical assumptions like the relation between ρ and d given in Equation 7.

In order to achieve f_{zoom} values of 10 % we have used a second amplifier stage for the distances 80 cm and 100 cm. This explains the increased U_{pp} values given in Table 2 for these two distances.

Table 2. SCA-attack results achieved at the analyzed distances (for $d = 80\,cm$ and $d = 100\,cm$ a second amplifier stage was used)

d	U_{pp}	*Resolution*	f_{zoom}	*Traces used*	$\rho_{maxCorrect}$	*nrdp*
cm	mV	$\frac{mV}{div}$	%			
25	1 600	10	6.25	3 000	0.204	10.95
35	590	5	8.47	3 000	0.128	5.48
45	260	2	7.69	4 500	0.074	0.40
50	180	2	11.11	9 000	0.062	3.51
65	74	2	27.03	14 000	0.039	0.70
80	1 000	10	10.00	14 000	0.043	3.67
100	640	5	7.81	30 000	0.025	0.51

For our remote SCA attacks we have placed reader and tag close to each other. However, in a real-world attack scenario, it would be advantageous for an attacker to place the reader also at a certain distance from the tag. In that way the whole attack can be applied completely remotely without being noticed by the tag owner. As demonstrated by Kfir et al. [16], remotely powering and also communicating with an NFC tag up to ranges of 40 cm can be easily realized with low-cost equipment (below 100 $). Kfir et al. achieved this range extension by using a larger reader antenna and by increasing the strength of the reader field. Using such a setting a large amount of traces can be recorded unnoticed by the owner of the NFC tag. For HF tags operating in the so-called vicinity range (e.g., according to ISO15693 [11]) that can anyway achieve larger communication ranges of up to 1.5 m, such remote attacks are even much easier to conduct as no modification of the reader device is necessary.

7 Conclusion

In this work we presented so-called remote SCA attacks on the AES implementation of a security-enabled NFC tag. Our results confirm that it is possible to measure side-channel information of NFC tags at distances up to 1 m although the communication range between reader and tag in NFC systems is limited to a few centimeters according to the specification. The remote attacks are enabled by the fact that the power consumption of the tag chip influences the amplitude of the reader signal. We name this effect *parasitic load modulation*. By measuring the small amplitude variations of the reader signal, DEMA attacks can be applied from a distance. In order to measure these weak variations no special analog preprocessing circuits are required, a self-made loop antenna, an amplifier, and an oscilloscope are sufficient. As the desired side-channel information is modulated on the reader signal, only its peaks were recorded, followed by a downsampling step in software. This makes the attack not only very simple to conduct but also improves the attack performance as the recorded traces become significantly smaller in size. Moreover, we formulate the relation between obtained correlation coefficient and distance of tag and measurement antenna, confirming our practical observations. These are the first remote SCA attacks on an NFC/RFID tag operating in the HF range at all. We assume that other tags operating in the HF range at a carrier frequency of 13.56 MHz (e.g., ISO15693 tags) are vulnerable as well to such attacks, underlining the importance of integrating proper SCA countermeasures.

Acknowledgements. The work presented in this article has been supported by the European Commission through the ICT program TAMPRES (under contract ICT-SEC-2009-5-258754).

References

[1] Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM Side-channel(s). In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003)

[2] Bono, S., Green, M., Stubblefield, A., Juels, A., Rubin, A., Szydlo, M.: Security Analysis of a Cryptographically-Enabled RFID Device. In: Proceedings of USENIX Security Symposium, Baltimore, Maryland, USA, pp. 1–16 (July-August 2005)

[3] Carluccio, D., Lemke, K., Paar, C.: Electromagnetic Side Channel Analysis of a Contactless Smart Card: First Results. In: Oswald, E. (ed.) RFIDSec 2005, Graz, Austria, July 13-15, pp. 44–51 (2005)

[4] Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards. In: Second Advanced Encryption Standard (AES) Candidate Conference, Rome, Italy (1999)

[5] Courtois, N.T., O'Neil, S., Quisquater, J.-J.: Practical Algebraic Attacks on the Hitag2 Stream Cipher. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 167–176. Springer, Heidelberg (2009)

[6] Finkenzeller, K.: RFID-Handbook, 2nd edn. Carl Hanser Verlag (April 2003) ISBN 0-470-84402-7

[7] 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)

[8] Ha, J.C., Kim, C., Moon, S.-J., Park, I., Yoo, H.: Differential Power Analysis on Block Cipher ARIA. In: Yang, L.T., Rana, O.F., Di Martino, B., Dongarra, J. (eds.) HPCC 2005. LNCS, vol. 3726, pp. 541–548. Springer, Heidelberg (2005)

[9] Hutter, M., Mangard, S., Feldhofer, M.: Power and EM Attacks on Passive 13.56 MHz RFID Devices. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 320–333. Springer, Heidelberg (2007)

[10] International Organisation for Standardization (ISO). ISO/IEC 7816-4: Information technology - Identification cards - Integrated circuit(s) cards with contacts - Part 4: Interindustry commands for interchange (1995), `http://www.iso.org`

[11] International Organisation for Standardization (ISO). ISO/IEC 15693-3: Identification cards - Contactless integrated circuit(s) cards - Vicinity cards – Part 3: Anticollision and transmission protocol (2001)

[12] International Organization for Standardization (ISO). ISO/IEC 14443-3: Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards - Part3: Initialization and Anticollision(2001), `http://www.iso.org`

[13] International Organization for Standardization (ISO). ISO/IEC 14443-4: Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards - Part4: Transmission Protocol (2008), `http://www.iso.org`

[14] Jaffe, J.: More Differential Power Analysis: Selected DPA Attacks. Presented at ECRYPT Summerschool on Cryptographic Hardware, Side Channel and Fault Analysis (June 2006)

[15] Kasper, T., Oswald, D., Paar, C.: EM Side-Channel Attacks on Commercial Contactless Smartcards Using Low-Cost Equipment. In: Youm, H.Y., Yung, M. (eds.) WISA 2009. LNCS, vol. 5932, pp. 79–93. Springer, Heidelberg (2009)

[16] Kfir, Z., Wool, A.: Picking Virtual Pockets using Relay Attacks on Contactless Smartcard Systems. In: Proceedings SecureComm 2005, Athens, Greece, September 5-9, pp. 47–58. IEEE Computer Society (2005)

[17] Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)

[18] Korak, T., Plos, T., Hutter, M.: Attacking an AES-Enabled NFC Tag: Implications from Design to a Real-World Scenario. In: Schindler, W., Huss, S.A. (eds.) COSADE 2012. LNCS, vol. 7275, pp. 17–32. Springer, Heidelberg (2012)

[19] Lemke, K., Schramm, K., Paar, C.: DPA on *n*-Bit Sized Boolean and Arithmetic Operations and Its Application to IDEA, RC6, and the HMAC-Construction. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 205–219. Springer, Heidelberg (2004)

[20] Mangard, S.: Exploiting Radiated Emissions - EM Attacks on Cryptographic ICs. In: Ostermann, T., Lackner, C. (eds.) Proceedings of Austrochip 2003, October 3, pp. 13–16 (2003) ISBN 3-200-00021-X

[21] Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks – Revealing the Secrets of Smart Cards. Springer (2007) ISBN 978-0-387-30857-9

[22] National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard (November 2001), `http://www.itl.nist.gov/fipspubs/`

[23] NFC Forum. NFC Forum Type 4 Tag Operation - Technical Specification (March 2007)

[24] Nohl, K.: Cryptanalysis of Crypto-1. Computer Science Department University of Virginia, White Paper (2008)

[25] Oren, Y., Shamir, A.: Remote Password Extraction from RFID Tags. IEEE Transactions on Computers 56(9), 1292–1296 (2007)

[26] Örs, S.B., Gürkaynak, F.K., Oswald, E., Preneel, B.: Power-Analysis Attack on an ASIC AES Implementation. In: Proceedings of International Conference on Information Technology: Coding and Computing (ITCC 2004), Las Vegas, Nevada, USA, April 5-7, vol. 2, IEEE Computer Society (2004) ISBN 0-7695-2108-8

[27] Oswald, D., Paar, C.: Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 207–222. Springer, Heidelberg (2011)

[28] Plos, T.: Susceptibility of UHF RFID Tags to Electromagnetic Analysis. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 288–300. Springer, Heidelberg (2008)

[29] Plos, T., Maierhofer, C.: On Measuring the Parasitic Backscatter of Sensor-enabled UHF RFID Tags. In: Proceedings of ARES 2012, Prague, Czech Republic, pp. 38–46. IEEE (August 2012)

[30] Whitnall, C., Oswald, E.: A Comprehensive Evaluation of Mutual Information Analysis Using a Fair Evaluation Framework. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 316–334. Springer, Heidelberg (2011)

Practical Leakage-Resilient Pseudorandom Objects with Minimum Public Randomness

Yu Yu[1,2] and François-Xavier Standaert[3]

[1] Tsinghua University, Institute for Interdisciplinary Information Sciences, China
[2] East China Normal University, Department of Computer Science, China
[3] Université catholique de Louvain, ICTEAM/ELEN/Crypto Group, Belgium

Abstract. One of the main challenges in leakage-resilient cryptography is to obtain proofs of security against side-channel attacks, under realistic assumptions and for efficient constructions. In a recent work from CHES 2012, Faust et al. proposed new designs of stream ciphers and pseudorandom functions for this purpose. Yet, a remaining limitation of these constructions is that they require large amounts of public randomness to be proven leakage-resilient. In this paper, we show that tweaked designs with minimum randomness requirements can be proven leakage-resilient in minicrypt. That is, either these constructions are secure, or we are able to construct public-key cryptographic primitives from symmetric-key building blocks and their leakage functions (which is very unlikely). Hence, our results improve the practical relevance of two important leakage-resilient pseudorandom objects.

1 Introduction

Side-channel attacks are an important threat to the security of embedded devices like smart cards and RFID tags. Following the first publications on Differential Power Analysis [19] (DPA) and Electro-Magnetic Analysis [12,29] (EMA), a large body of work has investigated techniques to improve the security of cryptographic implementations. During the first ten years after the publication of these attacks, the solutions proposed were mainly taking advantage of hardware/software modifications. For example, it as been proposed to exploit new circuit technologies or to randomize the time and data in the implementations (see [3,4,36] for early proposals of these ideas, and many improvements and analyzes published at CHES). In general, these countermeasures are successful in the sense that they indeed reduce the amount of information leakage. Yet, security evaluations considering worst-case (profiled) side-channel attacks such as [33] usually reveal that reaching high security levels is expensive and highly dependent of physical assumptions. Taking the example of secret sharing (aka masking), multiple shares are required for this purpose (i.e. so-called higher-order security [34]). However, the implementation cost of higher-order masking schemes is significant [31], and the risk of physical effects leading to exploitable weaknesses (such as glitches [21]) leads to additional design constraints.

Motivated by the great challenges in physical security, recent works have also considered the possibility to analyze the effectiveness of countermeasures against side-channel attacks in a more formal way, and to design new primitives (aimed to be) inherently more secure against such attacks. Taking the case of symmetric cryptography building blocks (that are important primitives to design as they are usual targets of DPA attacks [20]), a variety of models have been introduced for this purpose, ranging from specialized to general. For example, a PRNG secure against side-channel key recovery attacks was proposed at ASIACCS 2008 by Petit et al. [25], and analyzed in front of a class of (realistic yet specific) leakage functions. Following, a construction of leakage-resilient stream cipher has been presented by Dziembowski and Pietrzak at FOCS 2008, together with a proof of security in the standard model [9]. Quite naturally, such "physical security proofs" raise a number of concerns regarding their relevance to practice, a topic that has been intensively discussed over the last couple of years. In particular, one of the fundamental issues raised by leakage-resilient cryptography is to determine reasonable restrictions of the leakage function, e.g. in terms of informativeness and computational power. As far as computational power is concerned (which will be our main concern in this paper), an appealing solution is to consider the leakage function to be polynomial time computable, as initially proposed by Micali and Reyzin [24], and leading to contrasted observations. On the one hand, polynomial time functions are significantly more powerful than actual leakage functions. For example, they allow so called "precomputation attacks" (aka future computation attacks) that are arguably unrealistic in practice [35]. On the other hand, meaningful alternatives seem quite challenging to specify. Furthermore, given that one obtains proofs of security under such strong leakages without paying too large implementation overheads, polynomial time functions remain a useful abstraction.

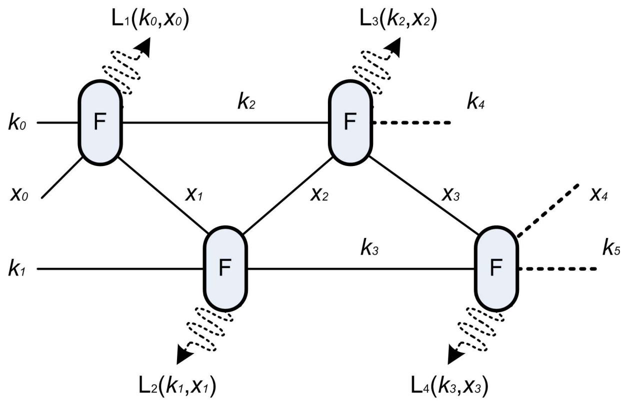


Fig. 1. The Eurocrypt 2009 stream cipher

In this context, one of the design tweaks used by Dziembowsky and Pietrzak is the so-called "alternating structure". Figure 1 depicts such an alternating structure for a simplified stream cipher proposed by Pietrzak at Eurocrypt 2009 [27], that can be instantiated only from (AES-based) weak Pseudo-Random Functions (wPRFs)[1]. If one assumes that the two branches of such an alternating structure leak independently, no leakage occuring in one of the branches can be used to compute bits that will be manipulated in future computations of the other branch, hence ruling out the possibility of precomputation attacks. The main drawback of this proposal is that a key bit-size of $2n$ can only guarantee a security of at most 2^n. Hence, as it appears unrealistic that a circuit actually leaks about something it will only compute during its future iterations, a following work by Yu et al. investigated the possibility to mitigate the need of an alternating structure [37]. In a paper from CCS 2010, they first proposed to design a "natural" (i.e. conform to engineering intuition) leakage-resilient stream cipher, which could only be proven secure under a (non-standard) random oracle based assumption. Next, they proposed a variant of the FOCS 2008 (and Eurocrypt 2009) designs, replacing the alternating structure by alternating public randomness, and under the additional (necessary) assumption that the leakage function is non adaptive. Eventually, in a recent work of CHES 2012, Faust et al. showed that large amounts of public randomness (i.e. linear in the number of stream cipher iterations) were actually required for the proof of Yu et al. to hold [10]. While it remains an open question to determine whether the exact construction proposed in [37] (using only two alternating public values) can be proven secure or attacked in a practical setting, this last result reveals a tension between the proof requirements and how the best known side-channel attacks actually proceed against leakage-resilient constructions [23].

Considering the previous observations, this paper tackles the fundamental question of how much public randomness is actually needed to obtain proofs of leakage-resilience in symmetric cryptography. For this purpose, we investigate (yet another) variant of stream cipher, where only a single public random value is picked up prior to (independent of) the selection of the leakage functions, and then expanded thanks to a PRNG. Quite naturally, a strong requirement for this approach to be interesting is that the seed of the PRNG should *not* be secret (or we would need a leakage-resilient PRNG to process it, i.e. essentially the problem we are trying to solve). Surprisingly, we show that this approach can be proven secure in minicrypt [17] (i.e. the hypothetical world introduced by Impagliazzo, where one-way functions exist, but public-key cryptography does not). More precisely, using the technique of [1] (see also similar ones in earlier literature [7,8,26,28]), we show that either the proposed solution is leakage-resilient,

[1] Besides their possible implementation costs, additional components in leakage-resilient constructions can also become a better target for a side-channel adversary, e.g. as discussed with the case of randomness extractors in the FOCS 2008 stream cipher [22,32]. In this respect, relying only on AES-based primitives (for which the security against side-channel attacks has been carefully analyzed) is an interesting feature of the Eurocrypt 2009 proposal in Figure 1.

or we are able to construct black-box constructions of public-key encryption schemes from symmetric primitives and their leakage functions. When using block ciphers such as the AES to instantiate the stream cipher, the latter is very unlikely due to known separation results between one-way functions and PKE [18]. We then conclude this work by illustrating that this observation also applies to PRFs for which various designs were already proposed [5,10,23,35].

Summarizing, proofs of leakage-resilience require to restrict the leakage function both in terms of informativeness and computing power. As finding useful and realistic restrictions is hard with state-of-the-art techniques, we consider an alternative approach, trying to limit the implementation overheads due to unrealistic models. Admittedly, our analysis is based on the same assumptions as the previously mentioned works (i.e. polynomial time, bounded and non-adaptive leakage functions). The quest for more realistic models remains a very important research direction. Meanwhile, we believe that our intermediate conclusion is important, as it highlights that leakage-resilient (symmetric) cryptography can be obtained with minimum public randomness (i.e. the public seed of a PRNG).

2 Background

2.1 The CCS 2010 Stream Cipher

The CCS 2010 construction, depicted in Figure 2, is based on the observation from the practice of side-channel attacks that leakage functions are more a property of the target device and measurement equipment than something that is adaptively chosen by the adversary. It therefore considers a weaker security model, in which the polynomial time (and bounded) leakage functions are fixed before the stream cipher execution starts. By considering those non-adaptively chosen leakage functions, the construction can be made more efficient and easier to implement in a secure way. This stream cipher is initialized with a secret key k_0 and two values p_0 and p_1 that can be public. Those two values are then used in an alternating way: at round i, one computes k_i and x_i by applying the

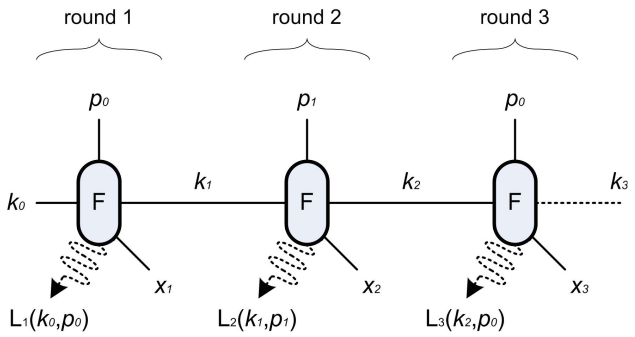


Fig. 2. The CCS 2010 stream cipher

wPRF to inputs k_{i-1} and $p_{i-1 \bmod 2}$. Thanks to the removal of the alternating structure, the complexity of a brute-force attack on this construction becomes directly related to the full length of the key material, which is now exploited in each round.

2.2 The CHES 2012 Stream Cipher

In a paper from CHES 2012, Faust et al. observed that the technical tools used to prove the CCS 2010 construction actually require to use independent public values in all the stream cipher rounds (rather than only two alternating ones). Therefore, only the slightly modified the construction suggested in Figure 3, assuming a common random string $p_0, p_1, p_2, \ldots$, can be proven secure with these tools. The practical advantages of this construction compared to the FOCS 2008 / Eurocrypt 2009 ones naturally become more contrasted. On the positive side, the fact that the values $p_0, p_1, p_2, \ldots$ are public can still make it easier to ensure that rounds leak independently of each other (which is implicitly required by the arguments of the leakage function): for example, a number of public p_i's can be stored in non-volatile memories for this purpose. On the other hand, this amount of public randomness required is linear in the number of stream cipher rounds, which is hardly realistic (hence leading the authors of [10] to pay more attention to leakage-resilient PRFs for which this penalty is less damaging - see Section 4 for a brief discussion).

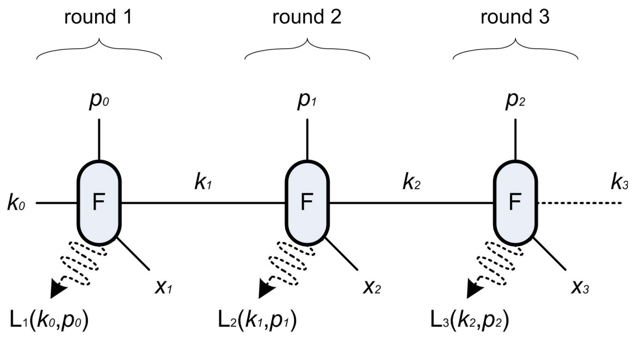


Fig. 3. The CHES 2012 stream cipher

3 Natural PRNG with Minimum Public Randomness

3.1 A New Proposal

As mentioned in introduction, it in unclear whether the need of large public randomness in leakage-resilient stream ciphers is due to proof artifacts or if the lack of such randomness can be exploited in realistic side-channel attacks. This

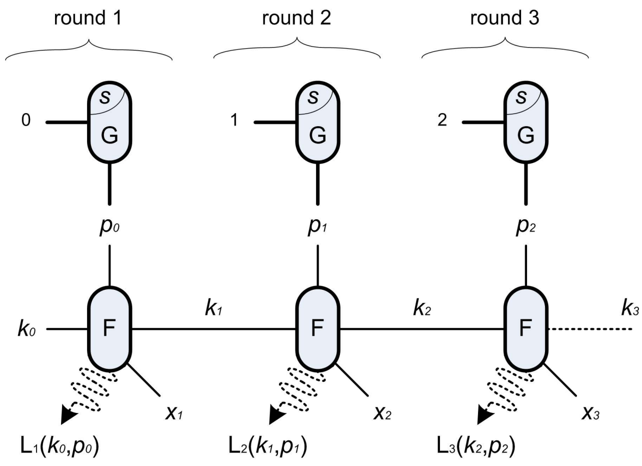


Fig. 4. Leakage-resilient stream cipher with minimum randomness

question is important as such attacks would most likely reveal an issue in the most natural construction of [37], where no public randomness is used at all and the proof is based on a random oracle assumption. In order to answer it, we propose an alternative stream cipher depicted in Figure 4.

The Proposed Stream Cipher. We denote our stream cipher with SC, let n be the security parameter, and (k_0,s) be the initial state of SC, where $k_0 \in \{0,1\}^n$ is a secret key and $s \in \{0,1\}^n$ a public seed, both randomly chosen. SC expands s into $p_0, p_1, p_2, \ldots$ on-the-fly by running a PRF $\mathsf{G} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ in counter mode[2], i.e., $p_i := \mathsf{G}(s,i)$. Then, SC uses the generated public strings $p_0, p_1, p_2, \ldots$ to randomize another PRF $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$, which updates the secret state k_i and produces the output x_i, i.e. $(k_i, x_i) := \mathsf{F}(k_{i-1}, p_{i-1})$. That is, the stream cipher SC in Figure 4 is essentially similar to the previous ones, excepted that any public string p_i is obtained by running a PRF on a counter value, using the public seed s.

Instantiation and Efficiency. Following [27], we instantiate F and G with a block cipher $\mathsf{BC} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, e.g. the AES. As will be shown in Lemma 4, it is sufficient to produce $\log(1/\varepsilon)$ bits of fresh pseudo-randomness for every p_i (and pad the rest with zero's), with ε a security parameter of the PRF F (see Definition 1). This further improves efficiency, as we only need to run G once every $\lfloor n/\log(1/\varepsilon) \rfloor$ iterations of F.

Leakage Models of the CCS 2010/CHES 2012 stream ciphers. For every i^{th} iteration, let $\mathsf{L}_i : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\lambda$ be a function (on k_{i-1}

[2] Alternatively, we can also expand s by iterating a length-doubling PRNG in a forward-secure way, but this would lead to less efficient designs and is not needed (since s is public).

and p_{i-1}) that outputs the leakage incurred during the computation of F on (k_{i-1},p_{i-1}). The CCS 2010/CHES 2012 constructions model the leakages as follows [10,37]:

1. (*Efficient computability*). L_i can be computed by polynomial-size circuits.
2. (*Bounded leakage per iteration*). The leakage function has bounded range given by $\lambda \in O(\log(1/\varepsilon))$, where ε is a security parameter of the PRF F (see Definition 1).
3. (*Non-adaptivity*). The selection of the leakage functions L_i is made prior to (or independent of) s, and thus only depends on k_{i-1} and p_{i-1}.

Note that strictly speaking, the leakage models needed to prove the security of the CCS 2012 and CHES 2012 stream ciphers are not exactly equivalent. Namely, the CHES 2012 stream cipher can further tolerate that each L_i not only depends on the current state (k_{i-1},p_{i-1}), but also on the past transcript $\mathsf{T}_{i-1} \overset{\mathsf{def}}{=} (x_1, \cdots, x_{i-1}, p_0, \cdots, p_{i-2}, \mathsf{L}_1(k_0, p_0), \cdots, \mathsf{L}_{i-1}(k_{i-2}, p_{i-2}))$. This is naturally impossible if only two p_i's are used.

Leakage models of FOCS 2008/Eurocrypt 2009 stream ciphers. The FOCS 2008/Eurocrypt 2009 constructions consider a model similar to the above one, but they do not require condition #3 and allow the adaptive selection of the leakage functions. That is, at the beginning of each round, the adversary adaptively chooses a function L_i based on his current view. As previously mentioned, this leads to unrealistic attacks as the adversary can simply recover a future secret state, say k_{100}, by letting each L_i leak some different λ bits about it. The authors of [9,27] deal with this issue by tweaking their stream cipher design with an alternating structure (as in Figure 1).

In the next sections, we will prove the leakage-resilient security of our stream cipher in the (non-adaptive) model from CCS 2010/CHES 2012. More precisely, we will also consider its less restrictive version where the leakage functions can depend on the past transcript. Yet, for brevity, we will not explicitly put T_{i-1} as an input of each L_i, as an adversary can hardwire them into L_i. Note also that we do not need to model leakages on G since the seed s (from which all $p_0, \cdots$, p_i can be efficiently computed) is public.

3.2 Security Analysis

Notations and Definitions. For security parameter n, a function $\mathsf{negl} : \mathbb{N} \to [0, 1]$ is *negligible* if for any $c > 0$ there is a n_0 such that $\mathsf{negl}(n) \leq 1/n^c$ for all $n \geq n_0$. We use uppercase letters (e.g. X) to denote a random variable and lowercase letters (e.g. x) to denote a specific value, with exceptions being n, t and q which are reserved for security parameter, circuit-size (or running time) and query complexity, respectively. We write $x \leftarrow X$ to denote the sampling of a random x according to X. We use U_n to denote the uniform distribution over $\{0,1\}^n$. For function f, we denote its circuit-size complexity by $\mathsf{size}(f)$ or t_f. We denote with $\Delta_D(X,Y)$ the advantage of a circuit D in distinguishing the random variables X,Y: $\Delta_{\mathsf{D}}(X,Y) \overset{\mathsf{def}}{=} |\ \Pr[\mathsf{D}(X)=1] - \Pr[\mathsf{D}(Y)=1]\ |$. The *computational distance* between two random variables X,Y is defined with $\mathsf{CD}_t(X,Y) \overset{\mathsf{def}}{=} \max_{\mathsf{size}(\mathsf{D})\le t}\ \Delta_{\mathsf{D}}(X,Y)$, which takes the maximum over all distinguishers D of size t. For convenience, we use $\mathsf{CD}_t(X,Y|Z)$ as shorthand for $\mathsf{CD}_t((X,Z),(Y,Z))$. The min-entropy of X is defined as $\mathbf{H}_\infty(X) \overset{\mathsf{def}}{=} -\log(\max_x \Pr[X=x])$. We finally define average (aka conditional) min-entropy of a random variable X conditioned on Z as:

$$\widetilde{\mathbf{H}}_\infty(X|Z) \overset{\mathsf{def}}{=} -\log\left(\ \mathbb{E}_{z\leftarrow Z}\left[\ \max_x \Pr[X=x|Z=z]\ \right]\ \right),$$

where $\mathbb{E}_{z\leftarrow Z}$ denotes the expected value computed over all $z \leftarrow Z$.

Standard Security Notions. Indistinguishability requires that no efficient adversary is able to distinguish a real distribution from an idealized one (e.g. uniform randomness) with non-negligible advantage. In this paper, we will work in the concrete non-uniform setting[3]. Yet, we note that the proof can be made uniform by adapting the technique from [2,38] (see [9] for a discussion). Given this precision, a standard PRF is defined as:

Definition 1 (PRF). $\mathsf{F}: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ *is a pseudorandom function (PRF) if for all polynomial-size distinguisher* D *making up to any polynomial number of queries, we have:*

$$|\Pr[\mathsf{D}^{\mathsf{F}(k,\cdot)}=1 \mid k \leftarrow U_n] - \Pr[\mathsf{D}^{\mathsf{R}(\cdot)}=1]\ | \ \le\ \mathsf{negl}(n),$$

where R *is a random function uniformly drawn from function family* $\{\{0,1\}^n \to \{0,1\}^m\}$. *Furthermore, we say that* F *is a (t,q,ε)-secure PRF if for all distinguishers* D *of size t making q queries, the above advantage is bounded by ε.*

Security without Leakages. Without considering side-channel adversaries, the security of SC is easily proven using a standard hybrid argument, by considering F (on any fixed input) as a PRG, and without any security requirement about G (which could just output any constant). This is formalized by the following theorem:

Theorem 1 (Security without Leakages). *If* F *is a* $(t,1,\varepsilon)$*-secure PRF, then* SC *is* (t',ℓ,ε')*-secure, i.e.* $\mathsf{CD}_{t'}((X_1,X_2,\cdots,X_\ell),U_{n\ell}|S) \le \varepsilon'$, *with* $t' \approx t - \ell{\cdot}t_{\mathsf{F}}$ *and* $\varepsilon' \le \ell \cdot \varepsilon$.

Leakage-Resilient Security. We first observe that as soon as some leakage is given to the adversary, he can easily exploit it to distinguish x_i from

[3] An efficient uniform adversary can be considered as a Turing-machine which on input 1^n (security parameter in unary) terminates in time polynomial in n, whereas its non-uniform counterpart will, for each n, additionally get some polynomial-length advice.

uniform randomness (e.g. $\mathsf{L}_i(k_{i-1}, p_{i-1})$ leaking the first bit of x_i is enough for this purpose). Thus, all previous approaches in leakage-resilient cryptography require that any (computationally bounded) adversary observing the leakages for as many rounds as he wishes should not be able to distinguish the next x_ℓ without seeing $\mathsf{L}_\ell(k_{\ell-1}, p_{\ell-1})$ [9,10,27,37]. Formally, let:

$$\mathsf{view}_\ell(\mathsf{A}, \mathsf{SC}, K_0, S) \overset{\mathsf{def}}{=} (S, X_1, \cdots, X_{\ell-1}, \mathsf{L}_1(K_0, P_0), \cdots, \mathsf{L}_{\ell-1}(K_{\ell-2}, P_{\ell-2})) \quad (1)$$

denote the view of adversary A after attacking SC (initialized with K_0 and S) for ℓ rounds, for which we use shorthand view_ℓ in the rest of the paper. Given a distinguisher D, we then define its indistinguishability advantage (on uniform K_0 and S) as:

$$\mathsf{AdvInd}(\mathsf{SC}, \mathsf{A}, \mathsf{D}, \ell) \overset{\mathsf{def}}{=} |\Pr_{K_0,S}[\mathsf{D}(\mathsf{view}_\ell, X_\ell) = 1] - \Pr_{K_0,S}[\mathsf{D}(\mathsf{view}_\ell, U_n) = 1]\,|.$$

We will use $\mathsf{size}(\mathsf{A}) \overset{\mathsf{def}}{=} \ell(t_\mathsf{G} + t_\mathsf{F}) + \sum_{i=1}^{\ell-1} t_{\mathsf{L}_i}$ to denote the circuit-size complexity of the physical implementation of SC and $\mathsf{size}(\mathsf{D})$ to denote the circuit-size complexity of D.

Using these notations, our main result can be stated as follows.

Theorem 2 (Leakage-Resilient Security). *If F is $(t,2,\varepsilon)$-secure PRF, and G is a (t,q,ε)-secure PRF, then for any $\ell \leq q$, adversary A, distinguisher D with $(\mathsf{size}(\mathsf{A}) + \mathsf{size}(\mathsf{D})) \in \Omega(2^{3\lambda}\varepsilon \cdot t/n)$ and for any leakage size (per round) λ, we have that either:*

$$\mathsf{AdvInd}(\mathsf{SC}, \mathsf{A}, \mathsf{D}, \ell) \in O(\ell\sqrt{2^{3\lambda} \cdot \varepsilon}),$$

or otherwise there exist efficient black-box constructions of public key encryption (PKE) from the PRFs F and G and the leakage functions $\mathsf{L}_1,\cdots,\mathsf{L}_{\ell-1}$.

How to Interpret the Result? The above theorem is a typical "win-win" situation, similar to those given in [1,7,8,26,28], where a contradiction to one task gives rise to an efficient protocol for another seemingly unrelated (and sometimes more useful) task. As mentioned in introduction, we know from [18] that black box constructions of PKE from PRFs are very unlikely to exist. Thus, if the building primitives F and G are one-way function equivalents (i.e. they are not PKE primitives), for example using practical block ciphers such as the AES, and the leakage functions are intrinsic to hardware implementation (i.e. not artificially chosen) then the stream cipher SC will be leakage-resilient as stated above. Before giving the proof, we recall the notion of HILL pseudo-entropy:

Definition 2 (HILL Pseudo-entropy [14,16]). *X has at least k bits of HILL pseudo-entropy, denoted by $\mathbf{H}^{\mathsf{HILL}}_{\varepsilon,t}(X) \geq k$, if there exists Y so that $\mathbf{H}_\infty(Y) \geq k$ and $\mathsf{CD}_t(X,Y) \leq \varepsilon$. X has at least k bits of HILL pseudo-entropy conditioned on Z , denoted by $\mathbf{H}^{\mathsf{HILL}}_{\varepsilon,t}(X|Z) \geq k$, if there exists (Y,Z') such that $\widetilde{\mathbf{H}}_\infty(Y|Z') \geq k$ and $\mathsf{CD}_t((X,Z),(Y,Z')) \leq \varepsilon$.*

Outline of the Proof. We will present the proof in two main steps. First, we will show the security of our stream cipher when the seed is kept secret. This part of the proof essentially borrows techniques from previously published papers. Next, we will show our main result, i.e. that either leakage-resilience is maintained when S is public, or we have efficient black box constructions of PKE from PRFs as stated in Theorem 2.

Lemma 1 (Security of SC with Secret S). *Let* $P_{[0\cdots\ell-1]} \overset{\mathsf{def}}{=} (P_0, \cdots, P_{\ell-1})$. *For the same* F, G, ℓ, A, D *as given in Theorem 2, we have that:*

$$| \Pr_{K_0,S}[\mathsf{D}(\mathsf{view}_\ell \setminus S, P_{[0\cdots\ell-1]}, X_\ell) = 1] - \mathsf{D}(\mathsf{view}_\ell \setminus S, P_{[0\cdots\ell-1]}, U_n) = 1] \mid \in O(\ell\sqrt{2^{3\lambda}\cdot\varepsilon}).$$

Proof sketch. Since G is a secure PRF and S is leak-free, it suffices to prove the security by replacing every P_i by true randomness P'_i. The conclusion follows from Lemma 2 below, by letting $i = \ell$ and applying computational extractor[4] F on $K_{\ell-1}$ and $P'_{\ell-1}$. It essentially holds because $P'_{\ell-1}$ is independent of all preceding random variables. □

Lemma 2 (The i^{th} round HILL Pseudo-entropy). *Assume that we use uniform randomness* $P'_0, \cdots, P'_{\ell-1}$ *and define the view accordingly as below:*

$$\mathsf{view}'_\ell \overset{\mathsf{def}}{=} (P'_0, \cdots, P'_{\ell-1}, X_1, \cdots, X_{\ell-1}, \mathsf{L}_1(K_0, P'_0), \cdots, \mathsf{L}_{\ell-1}(K_{\ell-2}, P'_{\ell-2})). \quad (2)$$

Then we have:

$$\mathbf{H}^{\mathsf{HILL}}_{\varepsilon_i,t_i}(K_{i-1}|\mathsf{view}'_i \setminus P_{i-1}) \geq n - \lambda, \quad (3)$$

where $\varepsilon_i = 2(i-1)\sqrt{2^{3\lambda}\cdot\varepsilon}$ *and* $(t_i + (i-1)t_{\mathsf{F}} + \sum_{j=1}^{i-1} t_{\mathsf{L}_i}) \in \Omega(2^{3\lambda}\varepsilon \cdot t/n)$.

A proof of this Lemma can be found in [10] (and implicitly in [9,27,37]). We will provide an alternative proof with improved bounds in Section 3.3, by utilizing recent technical lemmata from [11] (slightly improving the dense model theorem [9,30]) and Lemma 4 from [6], which explicitly states that a PRF used as computational exactor only needs $\log(1/\varepsilon)$ bits of randomness (which, as mentioned in Section 3.1, is desirable for efficiency).

The only difference between Lemma 1 and our final goal (i.e. Theorem 2) is that the security guarantee of the former one forbids adversary to see S (it only makes P_0, $\cdots$, $P_{\ell-1}$ public). We now argue why this security guarantee remains when additionally conditioned on S. Beforehand, we introduce preliminaries about key-agreement and PKE.

[4] As shown in Lemma 4, PRFs are computational extractors in the sense that when applied to min-entropy sources (or their computational analogue HILL pseudo-entropy sources), one obtains pseudo-random outputs provided that independent P'_is are used.

Key-Agreement and PKE. PKE is equivalent to a 2-pass key-agreement protocol [18], which in turn can be obtained from a 2-pass bit-agreement protocol with noticeable correlation and overwhelming security [15]. *Bit-agreement* refers to a protocol in which two efficient parties Alice and Bob (without any pre-shared secrets) communicate over an authenticated channel. At the end of the protocol, Alice and Bob output a bit b_{A} and b_{B}, respectively. The protocol has *correlation* ϵ, if it holds that $\Pr[b_{\mathsf{A}} = b_{\mathsf{B}}] \geq \frac{1+\epsilon}{2}$. Furthermore, the protocol has security δ, if for every efficient adversary Eve, which can observe the whole communication C, it holds that $\Pr[\mathsf{Eve}(1^k, C) = b_{\mathsf{B}}] \leq 1 - \frac{\delta}{2}$.

The following Lemma completes the proof of Theorem 2.

Lemma 3 (Secret vs. Public S). *For the same* F, G, ℓ, A, D *as given in Theorem 2 such that by keeping S secret, the stream cipher* SC *is secure as stated in Lemma 1, i.e.*

$$|\Pr_{K_0,S}[\mathsf{D}(\mathsf{view}_\ell \setminus S, P_{[0\cdots\ell-1]}, X_\ell) = 1] - \mathsf{D}(\mathsf{view}_\ell \setminus S, P_{[0\cdots\ell-1]}, U_n) = 1]\ | = \mathsf{negl}(n), \tag{4}$$

we have that either the above is still negligible when additionally conditioned on S, or otherwise there exists efficient black-box constructions of public key encryption from the PRFs F *and* G *and the leakage functions* $\mathsf{L}_1,\cdots,\mathsf{L}_{\ell-1}$.

Proof. By contradiction, let us assume that for some $c > 0$ and for infinitely many n's, there exists efficient $\tilde{\mathsf{D}}$ such that: $\Pr_{K_0,S}[\tilde{\mathsf{D}}(\mathsf{view}_\ell, X_\ell) = 1] - \Pr_{K_0,S}[\tilde{\mathsf{D}}(\mathsf{view}_\ell, U_n) = 1] \geq \frac{1}{n^c}$. We construct a 2-pass bit-agreement protocol as in Figure 5.

It follows from Equation (4) that no efficient passive adversary Eve (observing the communication) will be able to guess b_B (i.e. whether r is x_ℓ or uniform randomness) with more than negligible advantage. Furthermore, the bit-agreement also achieves correlation:

$$\boxed{\begin{array}{ccc}
\text{Alice} & & \text{Bob} \\
s \leftarrow U_n & \xrightarrow{p_0,\cdots,p_{\ell-1}} & k_0 \leftarrow U_n \\
p_0,\cdots,p_{\ell-1} \leftarrow \mathsf{G}(s,0),\cdots,\mathsf{G}(s,\ell-1) & & \text{Evaluate } \mathsf{SC} \text{ on } k_0, p_0,\cdots,p_{\ell-1} \\
 & & \text{to get } \mathsf{view}_\ell \setminus s \text{ and } x_\ell \\
 & & b_{\mathsf{B}} \leftarrow U_1 \\
 & \xleftarrow{r,\, \mathsf{view}_\ell \setminus s} & \text{if } b_{\mathsf{B}} = 0 \text{ then } r := x_\ell \\
b_{\mathsf{A}} \leftarrow \tilde{\mathsf{D}}(r, \mathsf{view}_\ell) & & \text{else if } b_{\mathsf{B}} = 1 \text{ then } r \leftarrow U_n
\end{array}}$$

Fig. 5. A bit agreement protocol from any PRFs F, G and leakage functions $\mathsf{L}_1, \cdots, \mathsf{L}_{\ell-1}$

$$\begin{aligned}\Pr[b_\mathsf{A}=b_\mathsf{B}] &= \underbrace{\Pr[b_\mathsf{B}=1]}_{=1/2}\Pr[b_\mathsf{A}=1|b_\mathsf{B}=1] + \underbrace{\Pr[b_\mathsf{B}=0]}_{=1/2}\underbrace{\Pr[b_\mathsf{A}=0|b_\mathsf{B}=0]}_{=1-\Pr[b_\mathsf{A}=1|b_\mathsf{B}=0]} \\ &= \frac{1}{2}\left(\Pr[b_\mathsf{A}=1|b_\mathsf{B}=1] + 1 - \Pr[b_\mathsf{A}=1|b_\mathsf{B}=0]\right) \\ &= \frac{1}{2}\left(1 + \Pr_{K_0,S}[\tilde{\mathsf{D}}(\mathsf{view}_\ell, X_\ell)=1] - \Pr_{K_0,S}[\tilde{\mathsf{D}}(\mathsf{view}_\ell, U_n)=1]\right) \geq \frac{1+\frac{1}{n^c}}{2},\end{aligned}$$

which implies 2-pass key agreement and PKE (by privacy amplification and parallel repetition [15]). Intuitively, the protocol can be seen as a bit-PKE. That is, Alice generates secret and public key pair $sk = s$ and $pk = (p_0, \cdots, p_{\ell-1})$ respectively, and sends her public key to Bob for him to encrypt his message b_B such that only Alice (with secret key sk) can decrypt (with non-negligible correlation). This completes the proof. □

As observed in [1], we can further extend this type of bit-PKE to a 1-out-of-2 Oblivious Transfer (OT) against curious-but-honest adversaries[5] as follows. For choice bit b, Alice first samples $pk_b := (p_0, \cdots, p_{\ell-1})$ and $pk_{1-b} \leftarrow U_{n\ell}$ and then sends pk_0, pk_1 to Bob. Bob, who holds two bits σ_0 and σ_1, uses the bit-PKE to encrypt σ_0 and σ_1 under pk_0 and pk_1, respectively. Finally, Alice recover σ_b and learns no information about σ_{1-b} (since it is computationally hidden by uniform randomness pk_{1-b}).

Additional remark about the protocol in Figure 5. In the non-uniform setting, any insecurity already implies efficient protocols for PKE and OT (using the hypothetical non-uniform $\tilde{\mathsf{D}}$), whereas in the uniform setting we will get practical and useful protocols, uniformly generated given the security parameter. See more discussion in [1].

3.3 Alternative Proof of Lemma 2

We will need the two following technical lemmata for the proof.

Theorem 3 (Dense Model Theorem [9,11]). *Let $(X, Z) \in \{0,1\}^n \times \{0,1\}^\lambda$ be random variables such that $\mathsf{CD}_t(X, U_n) < \varepsilon$ and let $\varepsilon_\mathsf{HILL} > 0$. Then we have:*

$$\mathbf{H}^{\mathsf{HILL}}_{2^\lambda\varepsilon+\varepsilon_\mathsf{HILL},t_\mathsf{HILL}}(X|Z) \;\geq\; n-\lambda, \qquad \text{where } t_\mathsf{HILL} \in \Omega(\varepsilon^2_\mathsf{HILL}\cdot t/n).$$

Lemma 4 (PRFs on Weak Keys and Inputs [6,27]). *If $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ is a $(2t, 2, \varepsilon)$-secure PRF, then for (K,Z) with $\widetilde{\mathbf{H}}_\infty(K|Z) \geq n-\lambda$, and independent P with $\mathbf{H}_\infty(P) \geq \log(1/\varepsilon)$, we have $\mathsf{CD}_t(\mathsf{F}(K,P), U_m \mid P, Z) \leq \sqrt{2^\lambda\cdot\varepsilon}$.*

[5] A 1-out-of-2 oblivious transfer refers to a protocol, where Alice has a bit b and Bob has two messages m_0 and m_1 such that Alice wishes to receive m_b without Bob learning b, while Bob wants to be assured that the Alice receives only one of the two messages.

Proof sketch. Similar to [9,27], we show the above by induction on ε_i and t_i. For $i = 1$, Equation (3) is trivially satisfied ($t_1 = \infty$ and $\varepsilon_1 = 0$). It remains to show that if Equation (3) holds for case i with parameter ε_i and t_i, then it must hold for case $i+1$ with $\varepsilon_{i+1} \leq \varepsilon_i + 2\sqrt{2^{3\lambda} \cdot \varepsilon}$ and $t_{i+1} = \min\{t_i - (t_{\mathsf{F}} + t_{\mathsf{L}_i}), \Theta(2^{3\lambda}\varepsilon \cdot t/n)\}$. By Definition 2, Equation (3) with (ε_i,t_i) refers to the fact that conditioned on $\mathsf{view}'_i \setminus P'_{i-1}$, there exists $\tilde{K}_{i-1}$ with $n - \lambda$ bits of average min-entropy such that K_{i-1} is (t_i, ε_i)-close to $\tilde{K}_{i-1}$. By our leakage assumptions, P'_{i-1} is independent of (K_{i-1},$\mathsf{view}'_i \setminus P'_{i-1}$), so if we apply F to $\tilde{K}_{i-1}$ and P'_{i-1}, Lemma 4 directly implies that:

$$\mathsf{CD}_{t/2}(\ (\tilde{K}_i, \tilde{X}_i) := \mathsf{F}(\tilde{K}_{i-1}, P'_{i-1})\ , U_{2n} \mid \mathsf{view}'_i\ ) \leq \sqrt{2^{\lambda} \cdot \varepsilon}.$$

Taking into account $\mathsf{L}_i(\tilde{K}_{i-1}, P'_{i-1})$, we know by Theorem 3 that:

$$\mathbf{H}^{\mathsf{HILL}}_{2\sqrt{2^{3\lambda}\cdot\varepsilon}, \Theta(2^{3\lambda}\varepsilon\cdot t/n)}(\ \tilde{K}_i, \tilde{X}_i \mid \mathsf{view}'_i, \mathsf{L}_i(\tilde{K}_{i-1}, P'_{i-1})\ )\ \geq\ 2n - \lambda,$$

which implies (using the chain rule for min-entropy) that $\tilde{K}_i$ has $n - \lambda$ bits of HILL pseudo-entropy (for the same parameters) conditioned on $\tilde{X}_i$. Note that this is almost what we want except that F is applied to $\tilde{K}_{i-1}$ rather than K_{i-1}. Hence, we need to pay $2\sqrt{2^{3\lambda} \cdot \varepsilon}$ for $\varepsilon_{i+1} - \varepsilon_i$, and lose t_{F}+t_{L_i} in complexity (to simulate the experiment). □

4 Leakage-Resilient PRFs

By minimizing their randomness requirements, the previous results improve the relevance of leakage-resilient stream ciphers. Besides, they also increases our confidence that simple constructions such as the first proposal in [37] are indeed secure against side-channel attacks. Hence, a natural question is to investigate whether a similar situation is observed for PRFs. In this context, three proposals have been analyzed in the literature. Standaert et al. first observed in [35] that a tree-based construction such as the one of Goldreich, Goldwasser and Micali [13] inherently brings improved resistance against side-channel attacks. They proved its leakage-resilience under a (non-standard) random oracle based assumption. Next, Dodis and Pietrzak proposed a similar tree-based design using an alternating structure, and proved its leakage-resilience in the standard model. Finally, Faust et al. replaced the alternating structure by public randomness (following the approach they used for the stream cipher in Figure 3) [10]. In this last case, a fresh p_i is required in each step of the PRF tree. The techniques described in the previous section can be directly applied to mitigate this requirement. That is, one can run a PRF on a counter and public seed to generate the p_i's. As in Lemma 3, either this construction is secure, or we can build a bit agreement protocol using the PRFs and leakage functions of the figure. While the randomness saving may be not substantial for a regular PRF (with input size linear in n), it will be desirable for variants that handle long (polynomial-size) inputs, e.g.

for Message Authentication Codes (MACs). Finally, we note that as in [10], the constructed leakage-resilient PRF is only secure against non-adaptive inputs.

Acknowledgements. Yu Yu was supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61172085, 61061130540, 61073174, 61103221, 11061130539, 61021004 and 61133014. François-Xavier Standaert is an associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in part by the ERC project 280141 on CRyptographic Algorithms and Secure Hardware (CRASH).

References

1. Barak, B., Dodis, Y., Krawczyk, H., Pereira, O., Pietrzak, K., Standaert, F.-X., Yu, Y.: Leftover Hash Lemma, Revisited. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 1–20. Springer, Heidelberg (2011)
2. Barak, B., Shaltiel, R., Wigderson, A.: Computational Analogues of Entropy. In: Arora, S., Jansen, K., Rolim, J.D.P., Sahai, A. (eds.) RANDOM 2003 and APPROX 2003. LNCS, vol. 2764, pp. 200–215. Springer, Heidelberg (2003)
3. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
4. Clavier, C., Coron, J.-S., Dabbous, N.: Differential Power Analysis in the Presence of Hardware Countermeasures. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)
5. Dodis, Y., Pietrzak, K.: Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 21–40. Springer, Heidelberg (2010)
6. Dodis, Y., Yu, Y.: Overcoming weak expectations. Short version appears in Information Theory Workshop ITW 2012 (2012), http://www.cs.nyu.edu/~dodis/ps/weak-expe.pdf
7. Dubrov, B., Ishai, Y.: On the randomness complexity of efficient sampling. In: Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC 2006), pp. 711–720 (2006)
8. Dziembowski, S.: On Forward-Secure Storage. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 251–270. Springer, Heidelberg (2006)
9. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008), pp. 293–302 (2008)
10. Faust, S., Pietrzak, K., Schipper, J.: Practical Leakage-Resilient Symmetric Cryptography. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 213–232. Springer, Heidelberg (2012)
11. Fuller, B., O'Neill, A., Reyzin, L.: A Unified Approach to Deterministic Encryption: New Constructions and a Connection to Computational Entropy. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 582–599. Springer, Heidelberg (2012)
12. 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)

13. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. In: Proceedings of the 25th Annual Symposium on Foundations of Computer Science (FOCS 1984), pp. 464–479 (1984)
14. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. 28(4), 1364–1396 (1999)
15. Holenstein, T.: Key agreement from weak bit agreement. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC 2005), pp. 664–673 (2005)
16. Hsiao, C.-Y., Lu, C.-J., Reyzin, L.: Conditional Computational Entropy, or Toward Separating Pseudoentropy from Compressibility. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 169–186. Springer, Heidelberg (2007)
17. Impagliazzo, R.: A personal view of average-case complexity. In: Proceedings of Structure in Complexity Theory Conference, pp. 134–147 (1995)
18. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: The 21st Annual ACM Symposium on Theory of Computing (STOC 1989), pp. 44–61 (1989)
19. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
20. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007)
21. Mangard, S., Popp, T., Gammel, B.M.: Side-Channel Leakage of Masked CMOS Gates. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 351–365. Springer, Heidelberg (2005)
22. Medwed, M., Standaert, F.-X.: Extractors against Side-Channel Attacks: Weak or Strong? In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 256–272. Springer, Heidelberg (2011)
23. Medwed, M., Standaert, F.-X., Joux, A.: Towards Super-Exponential Side-Channel Security with Efficient Leakage-Resilient PRFs. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 193–212. Springer, Heidelberg (2012)
24. Micali, S., Reyzin, L.: Physically Observable Cryptography (Extended Abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)
25. Petit, C., Standaert, F.-X., Pereira, O., Malkin, T., Yung, M.: A block cipher based pseudo random number generator secure against side-channel key recovery. In: Abe, M., Gligor, V.D. (eds.) ASIACCS, pp. 56–65. ACM (2008)
26. Pietrzak, K.: Composition Implies Adaptive Security in Minicrypt. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 328–338. Springer, Heidelberg (2006)
27. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
28. Pietrzak, K., Sjödin, J.: Weak Pseudorandom Functions in Minicrypt. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 423–436. Springer, Heidelberg (2008)
29. Quisquater, J.-J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
30. Reingold, O., Trevisan, L., Tulsiani, M., Vadhan, S.P.: Dense subsets of pseudorandom sets. In: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008), pp. 76–85 (2008)

31. Rivain, M., Prouff, E.: Provably Secure Higher-Order Masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010)
32. Standaert, F.-X.: How Leaky Is an Extractor? In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 294–304. Springer, Heidelberg (2010)
33. Standaert, F.-X., Malkin, T.G., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
34. Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The World Is Not Enough: Another Look on Second-Order DPA. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 112–129. Springer, Heidelberg (2010)
35. Standaert, F.-X., Pereira, O., Yu, Y., Quisquater, J.-J., Yung, M., Oswald, E.: Leakage resilient cryptography in practice. In: Towards Hardware Intrinsic Security: Foundation and Practice, pp. 105–139. Springer, Heidelberg (2010); Cryptology ePrint Archive, Report 2009/341 (2009), http://eprint.iacr.org/
36. Tiri, K., Verbauwhede, I.: Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 125–136. Springer, Heidelberg (2003)
37. Yu, Y., Standaert, F.-X., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 141–151. ACM (2010)
38. Zheng, C.J.: A uniform min-max theorem and its applications. STOC 2012, Poster (2012), http://cs.nyu.edu/~stoc2012/acceptedposters.pdf

Cryptanalytic Attacks on MIFARE Classic Protocol

Jovan Dj. Golić

Security Lab, Telecom Italia IT
Via Reiss Romoli 274, 10148 Turin, Italy
{jovan.golic}@it.telecomitalia.it

Abstract. MIFARE Classic is the most widely used contactless smart card in the world. It implements a proprietary symmetric-key mutual authentication protocol with a dedicated reader and a proprietary stream cipher algorithm known as CRYPTO1, both of which have been reverse engineered. The existing attacks in various scenarios proposed in the literature demonstrate that MIFARE Classic does not offer the desired 48-bit security level. The most practical scenario is the card-only scenario where a fake, emulated reader has a wireless access to a genuine card in the on-line stage of the attack. The most effective known attack in the card-only scenario is a differential attack, which is claimed to require about 10 seconds of average on-line time in order to reconstruct the secret key from the card. This paper presents a critical comprehensive survey of currently known attacks on MIFARE Classic, puts them into the right perspective in light of the prior art in cryptanalysis, and proposes a number of improvements. It is shown that the differential attack is incorrectly analyzed and is optimized accordingly. A new attack of a similar, differential type is also introduced. In comparison with the optimized differential attack, it has a higher success probability of about 0.906 and a more than halved on-line time of about 1.8 seconds.

Keywords: RFID, NFC, smart card attacks, key reconstruction attacks, stream ciphers, repeated nonce attacks, inversion atacks, resynchronization attacks, differential attacks.

1 Introduction

There are a number of proprietary algorithms and protocols used for data encryption and device authentication in RFID (Radio Frequency IDentification) and NFC (Near Field Communication) systems. The MIFARE Classic smart card, by NXP Semiconductors, is claimed to be the most widely used contactless smart card in the world, especially for access control to buildings and public transport. According to [6,7], this smart card covers more than 70% of the market share for access control worldwide. It is a memory card with several extra functionalities. It is capable of implementing a proprietary symmetric-key mutual authentication protocol and a proprietary encryption algorithm (stream cipher) known as CRYPTO1. They are also implemented on the dedicated contactless card reader. CRYPTO1 uses a preshared 48-bit secret key for encrypting messages between a card and a reader, including challenges and responses in the mutual authentication protocol. Data integrity on the RFID or NFC channel is not provided for. The protocol and CRYPTO1 are both reverse engineered in [14,7] and then analyzed in a number of scientific publications [14,12,6,7,4]. The proposed cryptanalytic attacks in various attack scenarios demonstrate that the MIFARE Classic smart card does not offer the 48-bit security level. Note that on a standard CPU, the brute-force attack on the 48-bit key can take several years, but less than an hour on the FPGA board COPACOBANA [13], at a cost of about 10000 USD. NXP Semiconductors has also introduced other smart cards for replacing MIFARE Classic, which use stronger authentication protocols and encryption algorithms, e.g., MIFARE Plus in 2008, based on 128-bit AES. According to [6], there are more than a billion of MIFARE and about 200 million of MIFARE Classic smart cards in use worldwide.

Attack scenarios considered include a passive scenario AttP and active scenarios AttAT and AttAR or their combinations. In AttP scenario, the attacker intercepts and records traces of valid transactions between a genuine card/tag and a genuine reader, and has the objective to decrypt some of the traces by cryptanalysis. There are a number of active scenarios where the attacker can initiate fake transactions between a tag and a reader. In AttAT scenario, the attacker uses a fake, emulated reader to access a genuine tag and, possibly in combination with AttP scenario, has the objective to perform an illegitimate transaction on the genuine tag, e.g., reading or modifying the stored data. In particular, in AttP or AttAT scenarios, the attacker may have the objective to reconstruct the genuine tag key. In AttAR scenario, the attacker uses a fake, emulated tag to access a genuine reader and has the objective to reconstruct the genuine reader key corresponding to the unique ID of the emulated tag. AttAT is the easiest scenario to implement in practice and is also called the tag-only or card-only scenario. In all the scenarios, the ProxMark instrument [15], with the open-source specification, programmed to handle the standard ISO/IEC 14443-A, can be used to emulate tags and readers and eavesdrop on valid transactions.

The reconstruction of the key totally breaks the system. The attacker can then perform any transaction on a genuine tag by using a fake reader with the genuine key. In particular, it can read or modify the stored data (e.g., read sensitive data or modify valuable data). Alternatively, the attacker can clone a tag, i.e., produce a fake tag with the genuine key or emulate a genuine tag and thus force a genuine reader into legitimate transactions or actions (e.g., access to building or access to any event requiring a ticket).

The main objectives of this paper are to critically analyze all known attacks on MIFARE Classic protocol in various scenarios and put them into the right perspective with respect to the prior art in cryptanalysis, propose their improvements, and introduce a novel attack, which appears to be the most effective currently known attack in the tag-only scenario, where the attacker uses a fake reader for a contactless access to a targeted genuine tag. For comparison of tag-only attacks, see Table 1. Note that the main practical limitation factor of these attacks is the on-line stage, which requires real-time access to the tag. The topic and the results are very interesting in practice, due the worldwide usage of MIFARE Classic smart cards and dedicated readers.

The MIFARE Classic protocol including the authentication protocol, the encryption algorithm CRYPTO1, and the error detection code are described in Section 2. The attack [12] in the combined AttP and AttAT scenario, which is independent of the structure of CRYPTO1, is discussed in Section 3. Two attacks [6] that work in AttP or AttAR scenario, namely, the time-memory-data tradeoff attack and the inversion attack are presented in Section 4 and Appendix A. Section 5 is dedicated to five attacks in AttAT, i.e., tag-only scenario. Three of them [7] are only outlined in Appendix B, as they are not very practical. The practical differential attack [4] is explained, critically analyzed, and optimized in Section 5 and Appendix C. A novel differential attack is proposed in Section 5 and Appendix C. All these cryptanalytic attacks relate to the mutual authentication protocol for one sector of the MIFARE Classic smart card. The attacks [6,7] aiming at reconstructing the keys for multiple sectors are presented in Section 6 and Appendix D. Conclusions are given in Section 7.

2 Description of MIFARE Classic Protocol

The EEPROM memory of the MIFARE Classic tag is divided into sectors, which are further divided into blocks of 16 bytes each. The last block of each sector contains two 48-bit secret keys and access conditions for the sector. The basic operations that can be performed on the memory data include read, write, increment, and decrement the stored value. The reader can access data in a given block only upon successful authentication for the sector containing that block, where the access conditions determine which of the two keys should be used and define the operations allowed for the sector. The first block of the first sector is a read-only block that contains special data including the unique 32-bit identifier (ID or UID) of the card, the parity byte computed on ID, and the manufacturer data. Secret keys stored on the tag can be specific to the tag or shared among a number of tags. In the latter case, a particular tag is identified by its unique ID stored in the read-only block. In the former case, in order to avoid storing all the keys in the reader memory, a key specific to the tag can be derived from a group master key and the unique ID.

The three-pass symmetric-key authentication protocol is of the challenge-response type, where the 32-bit challenge nonces n_T and n_R used by the tag and the reader are generated by the respective pseudorandom number generators. The tag nonce, n_T, is generated by a 16-bit linear feedback shift register (LFSR), which implies that it contains only 16 bits of entropy if the LFSR state is assumed to be uniformly random. The LFSR starts from the same state after powering up, has period 65535, and shifting its state every $9.44\,\mu$s it repeats its state after about 618 ms. The only randomness factor is thus a variable time when the tag nonce is produced. On the other hand, the reader pseudorandom number generator starts from the same state after every restart and produces a nonce n_R only upon invocation by the authentication protocol. In this case, the only randomness factor is a variable number of invocations after the restart. In principle, fresh nonces are important in order to avoid fake authentication by the replay attacks. However, if the nonces are treated as random in the protocol, but are in fact easily repeatable or predictable, then cryptanalytic attacks with a fake reader or a fake tag may be possible.

The tag and reader nonces serve for mutual authentication as well as for the initialization of CRYPTO1 for encrypting the data to be exchanged. CRYPTO1 is a stream cipher (keystream generator) with a structure of a nonlinear filter generator using a 48-bit LFSR and a nonlinear 20-bit Boolean function applied to the LFSR state to produce the keystream bits. The ciphertext bits are produced by XORing the plaintext and keystream bits. CRYPTO1 is initialized during the authentication process by using the challenge nonces n_T and n_R, which are bitwise XORed into the feedback path of the LFSR. Fresh nonces are important to ensure that the keystream is not repeated for the same sector key.

The main steps of the three-pass mutual authentication protocol are depicted in Fig. 1. In the preliminary two steps (which in reality include more auxiliary steps), the tag sends its unique ID and the reader sends back the index of the block (including that of the corresponding sector) to which it wishes to communicate to. The tag and the reader then select the key to be used for that sector, according to the access conditions. In the first pass, the tag sends n_T in the clear form. In the second pass, the reader sends back its response containing n_R and the answer a_R encrypted with two successive 32-bit keystream segments ks_1 and ks_2, respectively. In the third pass, the tag sends its answer a_T encrypted with the subsequent 32-bit keystream segment ks_3. We use the notation as in [7], i.e., $\{n_R\} = n_R \oplus ks_1$, $\{a_R\} = a_R \oplus ks_2$, and $\{a_T\} = a_T \oplus ks_3$. The answers to the challenges are defined by $a_R = \mathrm{suc}^2(n_T)$ and $a_T = \mathrm{suc}^3(n_T)$, where "suc" denotes the successor function associated with the 16-bit LFSR used for generating n_T, which maps 32 successive LFSR sequence bits into the next (non-overlapping) 32 successive LFSR sequence bits.

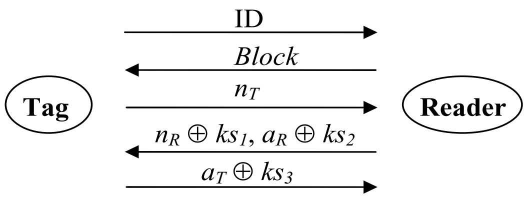


Fig. 1. MIFARE Classic authentication protocol

In the process, the same keystream is generated both by the tag and the reader, with the only difference that the reader uses n_R directly, whereas the tag first decrypts $\{n_R\}$ into n_R. This is achieved in the bitwise manner since ks_1 also depends on n_R, but in a specific way. Namely, as the current bit of ks_1

used to encrypt the current bit of n_R depends only on the previous bits of n_R, where the first bit of ks_1 does not depend on n_R at all, the bits of n_R can be recovered by the tag one at a time and then XORed into the feedback path of the LFSR. The LFSR in CRYPTO1 is first initialized with the 48-bit key as the initial state. Then, the LFSR is clocked 32 times during which $n_T \oplus \mathrm{ID}$ is XORed in the feedback loop of the LFSR, one bit at a time. Then, the LFSR is clocked another 32 times during which n_R is XORed in the feedback loop of the LFSR, one bit at a time. The first bit of ks_1 is generated after the insertion of the last bit of $n_T \oplus \mathrm{ID}$ and before the insertion of the first bit of n_R, whereas the last bit of ks_1 is generated before the insertion of the $32nd$ bit of n_R. The keystream segments ks_2 and ks_3, which are successively generated after ks_1, as well as the subsequent keystream bits are generated by clocking the LFSR autonomously, and thus depend on the key and all the bits of $n_T \oplus \mathrm{ID}$ and n_R.

Let $k = k_0k_1 \dots k_{47}$, $n_T = n_{T,0}n_{T,1} \dots n_{T,31}$, $n_R = n_{R,0}n_{R,1} \dots n_{R,31}$, and $u = u_0u_1 \dots u_{31}$ denote the 48-bit key and 32-bit tag and reader nonces and unique tag ID, respectively. Let $s = s_0s_1s_2 \dots$ denote the LFSR sequence and let $S_i = s_is_{i+1} \dots s_{i+47}$ and $L(S_i)$ denote the LFSR state and the feedback bit at time $i \geq 0$, respectively. The keystream sequence $z = z_0z_1z_2 \dots$ is defined by

$$s_i = k_i, \quad 0 \leq i \leq 47, \tag{1}$$

$$s_{48+i} = L(S_i) \oplus n_{T,i} \oplus u_i, \quad 0 \leq i \leq 31, \tag{2}$$

$$s_{80+i} = L(S_{32+i}) \oplus n_{R,i}, \quad 0 \leq i \leq 31, \tag{3}$$

$$s_{112+i} = L(S_{64+i}), \quad i \geq 0, \tag{4}$$

$$z_i = f(S_i), \quad i \geq 0, \tag{5}$$

where f denotes the filter function applied to the LFSR state as a whole. The three keystream segments used in the protocol are $ks_1 = z_{32}z_{33} \dots z_{63}$, $ks_2 = z_{64}z_{65} \dots z_{95}$, and $ks_3 = z_{96}z_{97} \dots z_{127}$. Effectively, f is defined as a balanced 20-bit Boolean function being a composition of five 4-bit Boolean functions and a 5-bit Boolean function. The tap positions forming inputs to f are taken from odd-indexed LFSR stages $9, 11, \dots, 47$, so that we can write $f(S_0)$ as $f(s_9, s_{11}, \dots, s_{47})$. It effectively depends on all 20 input bits and is linear neither in the first not the last input bit, i.e., it does not satisfy the sufficient condition [8] for the output sequence to preserve pure randomness of the input sequence to the filter function. In Section 5, it will be shown that this weakness opens a door for the most effective known attacks in the tag-only scenario.

If the additive inputs to the LFSR, $n_T \oplus u$ and n_R, are known, then it follows that the LFSR can also be clocked backwards, i.e., starting from the LFSR state at any time, one can determine all previous LFSR states, including the initial state defined by the key. As shown in [6,7], this so-called LFSR rollback also holds if the encryption $\{n_R\}$ is known instead of n_R, as a simple consequence of the fact that the filter function $f(S_0)$ does not effectively depend on the first, i.e., leftmost input bit s_0. We note that even when $f(S_0)$ effectively depends on s_0, the LFSR rollback is still feasible by the branching inversion/reversion attack [8,10]. Accordingly, the key can be easily recovered from any reconstructed internal state of CRYPTO1 and known transcripts of the authentication protocol.

The rationale for the mutual authentication is as follows. The correct encrypted answer $\{a_R \oplus ks_2\}$ can be produced by a genuine reader knowing the key and the nonces n_T and n_R. The correct encrypted answer $\{a_T \oplus ks_3\}$ can be produced by a genuine tag knowing the key and the nonces n_T and n_R, by first recovering n_R by decryption. The tag recovers a_R by decryption and verifies if it is equal to $\mathrm{suc}^2(n_T)$, thus authenticating the reader. The reader recovers a_T by decryption and verifies if it is equal to $\mathrm{suc}^3(n_T)$, thus authenticating the tag.

The mutual authentication protocol described above relates to the authentication for one sector. For authenticating access to multiple sectors of the same tag, the protocol is repeated by using new nonces n_T and n_R, with the only difference that, for each new sector, the new authentication command (Block) is encrypted with the previous sector key and the new n_T is bitwise encrypted by using the new key while it is bitwise inserted into the feedback path of the LFSR in the same way as n_R. The encrypted tag nonce sent by the tag is thus $n_T \oplus ks_0$, where $ks_0 = z_0 z_1 \ldots z_{31}$ denotes the preceding 32-bit keystream segment. The attacker thus knows n_T only for the first sector.

According to the standard ISO/IEC 14443-A, every plaintext byte sent is followed by the parity bit for error detection, computed as the binary complement of the XOR of all 8 bits in the byte. The parity bits in MIFARE Classic communication protocol are thus computed over the plaintext and, importantly, each parity bit is then encrypted with the same keystream bit subsequently reused to encrypt the next bit of the plaintext. The fact that the parity bits are computed over the plaintext instead of the ciphertext implies that the ciphertext itself reveals linear relations among the keystream bits (one relation per 8+1 ciphertext bits), which may be useful for ciphertext-only cryptanalytic attacks. The fact that the keystream bits are repeatedly used implies that the ciphertext also reveals linear relations among the plaintext bits other than those determined by the parity bits (one relation per pair of ciphertext bits corresponding to the parity bit of the current byte and the first bit of the next byte). In particular, this weakness can be used for reducing the uncertainty of new tag nonces used for authenticating access to multiple sectors, because such nonces are sent in the encrypted form and need to be guessed, as pointed out above. For example, the entropy of n_T given $n_T \oplus ks_0$ is thus reduced to 13=16-3 bits.

3 Attacks on Genuine Session and Genuine Tag

The attack [12] works in the combined AttP and AttAT scenario which requires access to a genuine authentication session and a genuine tag, by a fake reader. It neither uses knowledge of the encryption algorithm CRYPTO1 nor aims at reconstructing the key. It only aims at reconstructing portions of the keystream, which are repeatedly used for encrypting data in fake sessions initiated by a fake reader, on the condition that the tag can be forced to use the same tag nonce n_T as the one from the intercepted genuine session. This can be achieved by using either the periodic query or the reset query technique, due to the weakness of the tag pseudorandom number generator, where the only randomness factor is a variable time when the tag nonce is produced. Both techniques are also important for other attacks with fake readers or fake tags considered in this paper.

The periodic query technique is made possible by the short period of the tag pseudorandom number generator, which repeats its state roughly every 680 ms. Accordingly, the same value of n_T can be verifiably obtained (since n_T is transmitted in the clear), by forcing the genuine tag into several fake authentication sessions, called queries to the tag. The attacker can thus initiate fake authentication sessions periodically, with a precise timing, and thus ideally get the same n_T every 680 ms. In practice, it would take several attempts to get the same n_T each time. Alternatively, the reset query technique [14,6] is based on the fact that the tag pseudorandom number generator always starts shifting from the same initial state, after resetting, which can be achieved by switching-off the field and powering-up the passive tag by the fake reader. The attacker thus initiates the queries at a fixed time after resetting the tag and repeats this operation as fast as possible until the same value of n_T is obtained. According to [6], the required number of attempts to get the same n_T does not exceed ten each time. The reset query technique is significantly faster than the periodic query technique.

In fake sessions with repeated n_T, it is also needed to repeat the same reader nonce n_R, which is unknown to the fake reader. To this end, the fake reader simply replays the second pass of the authentication protocol from the recorded transcripts of the intercepted genuine session, in each fake session with the repeated n_T. The reader will thus be successfully authenticated to the tag and the tag will reproduce the same n_R and automatically generate the same keystream.

Since the repeated tag nonce enables the fake reader to successfully authenticate itself in the fake session by replaying the corresponding transcripts of the genuine session, the genuine tag will then proceed with the session as it were authentic. The desired keystream portions can be recovered from known plaintext portions obtained either from the genuine session or fake sessions, e.g., from the tag ID and manufacturer data, access conditions for the sector, and known tag responses to modified encrypted reader commands enabled by malleable bitwise XOR encryption. The recovered keystream portions, for a fixed value of the tag nonce and a fixed response of the fake reader in a genuine session, can then be used to perform illegitimate transactions on the tag, such as reading and modifying data stored in memory blocks. More precisely, any known or partially known 16-byte keystream block for any sector enables the attacker to read and modify the data at the same bit positions in any other block from the same sector.

4 Attacks on Genuine Session or Genuine Reader

The attacks [6] work in AttP scenario requiring interception of a genuine authentication session or in AttAR scenario requiring access to a genuine reader by a fake tag. In the latter case, the authentication session cannot be terminated, because the fake tag cannot be successfully authenticated to the reader as it does not know the key. The objective is to reconstruct the key for a single sector from the known keystream segments obtained from the recorded transcripts of the authentication protocol by using the known tag nonces n_T and the derived values

$a_R = \mathrm{suc}^2(n_T)$ and $a_T = \mathrm{suc}^3(n_T)$. Each known n_T gives rise to 64 keystream bits in AttP scenario and 32 or 64 keystream bits in AttAR scenario, where 64 bits correspond to ks_2 and ks_3 and 32 bits to ks_2. The keystream segment ks_3 can be recovered also in AttAR scenario, regardless of the fact that the fake tag does not send anything in the third pass of the protocol, if the reader proceeds by sending back either an encrypted "halt" command (because the tag authentication failed) or an encrypted "read" command (as if the tag authentication were successful). According to [6], this happens for most readers in practical use.

The structure of the stream cipher CRYPTO1 is needed in the attacks. The attacks first recover an internal state of CRYPTO1, e.g., the 48-bit LFSR state S_{64} at a time immediately after all the bits of n_T and n_R have been fed into the LFSR. This is the state from which the first bit of ks_2 is produced. The secret key is then reconstructed by the LFSR rollback, as explained in Section 2. The two attacks proposed in [6] include a time-memory-data tradeoff (TMDT) attack and an inversion attack. The former works in AttAR scenario and uses keystream segments from multiple (fake) authentication sessions with a genuine reader, while the latter works in AttP or AttAR scenario and uses the keystream segments from one or two authentication sessions.

Both the attacks are essentially known from previous publications, which is not mentioned in [6]. The TMDT attack can be regarded as an adaptation of the generic TMDT attack [1,9], for a stream cipher with 2^{48} states. More precisely, if the stored states correspond to a special form of tag nonces chosen by the fake tag, then the attack succeeds with probability 1 instead of a high probability typical of TMDT attacks. The required number of keystream segments are obtained from fake authentication sessions using variable n_T and random n_R produced by the genuine reader. We point out that the number of authentication sessions can be reduced at the expense of increasing the precomputation time, while keeping the same computation time and memory, by applying the TMDT attack [2]. The inversion attack can be regarded as an adaptation of the inversion attack with the decimation technique [8,10] to decimated keystream segments shorter than the LFSR length. Recall that the generic inversion attack on a nonlinear filter generator with the input memory size M and the greatest common divisor of the pairwise differences between the tap positions to the filter function being equal to d takes about $2^{M/d}$ steps (in CRYPTO1, $M = 38$ and $d = 2$). The adapted attack takes about 0.05 seconds and 8 MByte of memory on a standard CPU to recover the key. For comparison, it is claimed in [3] that the key can be reconstructed in about 12 seconds on the same CPU by an algebraic attack. More detailed descriptions of the attacks are given in Appendix A.

5 Attacks on Genuine Tag

The easiest attacks to implement are the tag-only or card-only attacks in AttAT scenario, with a fake reader having access to a targeted genuine tag. The fake reader forces the tag into multiple fake partial authentication sessions, called queries, in which it cannot be successfully authenticated to the tag. The objective is to reconstruct the key stored in the tag by using the transcripts of the partial authentication sessions and the known structure of CRYPTO1.

It would be impossible to obtain needed portions of known keystream from the tag, if it were not for a peculiar property of the authentication protocol, which can rightfully be called a bug or even a deliberately inserted weakness [7,4]. Namely, in the protocol, upon receiving a tag nonce in the clear, the fake reader sends two 32-bit ciphertexts, one standing for the encrypted reader nonce, $\{n_R\} = n_R \oplus ks_1$, and the other for the encrypted answer to the tag nonce, $\{a_R\} = a_R \oplus ks_2$. The fake reader also sends 8 encrypted parity bits $\{p\}$, corresponding to the 8 bytes sent. The attacker produces all the ciphertext directly, in a random or chosen manner depending on the attack to be conducted, without knowing the secret key. The reader's answer, n_R, decrypted by the tag will be wrong and the authentication will fail. The bug is that in this case, if all 8 decrypted parity bits in p happen to be correct, then the tag sends back to the reader a 4-bit ciphertext of a fixed 4-bit error message, encrypted with the first four bits of ks_3. This is a serious weakness, because it reveals to the attacker that the 8 parity bits are all correct and discloses 4 keystream bits. Moreover, the correct parity bits disclose to the attacker 8 independent linear combinations of keystream bits, which are here referred to as the keystream parity bits. Altogether, in the case of a successful query, the attacker thus gets 12 bits of information or entropy regarding the key, in the form of 4 keystream bits and 8 keystream parity bits. This can then be used for mounting key reconstruction attacks in the tag-only scenario, which are described in the sequel. Surprisingly, according to [4], unlicensed clone MIFARE Classic cards used in some countries always send out the encrypted error message, regardless of the values of the parity bits. This then greatly facilitates the attacks.

In the on-line stage of the attacks, the fake reader makes a number of queries to the genuine tag, in order to achieve a sufficient number of successful queries (with all 8 parity bits correct). The queries can use random or fixed tag nonces, where the latter, realized by the reset query technique, take about 50 times more time than the former. It is thus claimed in [7] that the attacker can perform about 1500 queries per second with a random tag nonce and about 30 queries per second with a fixed tag nonce realized by the reset query technique. In the off-line stage of the attacks, the collected keystream data is analyzed in order to reconstruct the key. The on-line stage is thus a practical bottleneck of the attacks and the required numbers of queries of one or the other type, together with some auxiliary, simple computations, determine the on-line complexity of the attacks. In addition, the attacks may also require significant precomputation time and storage in the setup stage.

For random tag nonces, the keystream is also random and cannot be controlled by the fake reader. Therefore, the best strategy for the fake reader to get a successful query is to choose randomly $\{n_R\}$, $\{a_R\}$, and $\{p\}$ until a successful query occurs, i.e., until all 8 parity bits in p are correct. On average, this requires 256 queries with random n_T. For fixed tag nonces, if $\{n_R\}$ is kept fixed by the fake reader, then the keystream and n_R will also be fixed. Further, if $\{a_R\}$ is also kept fixed, then a_R will be fixed and, hence, all 8 parity bits in p will be fixed too. To get a successful query, the best strategy for the fake reader is then to choose different instead of random values of $\{p\}$. The average number of queries is thus reduced to 128=256/2.

Each of the three attacks proposed in [7] has serious practical limitations, as shown in Table 1. Namely, the first attack has huge off-line time, the second attack has very large on-line time, and the third attack has huge precomputation time. Concise descriptions of the attacks, denoted as Attack 1, 2, and 3, are given in Appendix B, where it is shown that by using the queries with random tag nonces to obtain different tag nonces, the on-line time of Attack 3 can be reduced from about 2 minutes to about 7 seconds. We now concentrate on the fourth attack, proposed in [4] and denoted here as Attack 4. It overcomes all the limitations of the three attacks from [7]. In spirit, this attack is similar to Attack 2 from [7], but takes better advantage of differential properties of the nonlinear filter function f applied to the LFSR sequence. As a consequence, it requires a significantly smaller number of queries in the on-line stage and has a significantly smaller off-line time complexity. However, it is shown below that the analysis of the attack given in [4] is incorrect and that the attack can be simply improved by better usage of the queries with random and fixed tag nonces. As a result, the optimized attack, denoted as Attack 4* in Table 1, has a better performance.

The main idea of the differential attack [4] is for the fake reader to first get one successful query for some $\{n_T\}$, $\{n_R\}$, $\{a_R\}$, and $\{p\}$. Then, the fake reader performs a number of modifications of $\{n_R\}$ and, for each modification, performs further queries with fixed $\{n_T\}$, $\{n_R\}$, and $\{a_R\}$ and different $\{p\}$ in order to get a new successful query. The 32-bit encrypted reader nonce $\{n_R\}$ is modified by changing its last 3 bits and then fixed. For each of 7 possible changes, $\{a_R\}$ is randomly chosen and then fixed, and only the last 5 bits of $\{p\}$, which are (randomly) affected by the change of $\{n_R\}$ and $\{a_R\}$, are varied. On average, 16=32/2 such queries with a fixed nonce tag are needed for a successful query to occur. As a result of the on-line stage, the attacker thus obtains 8 successful queries yielding the known keystream data. The problem is that this keystream depends on unknown values of n_R. This can be overcome with a high probability, by using differential properties of f when shifted along the LFSR sequence.

Namely, it is claimed in [4] that with probability about 0.75, the 3 keystream bits used for the decryption of the last 3 bits of $\{n_R\}$ are independent of these 3 bits. As a consequence, each nonzero 3-bit change δ_3 of $\{n_R\}$ will result in the same change of the last 3 bits of n_R itself. Since the LFSR sequence depends linearly on n_R, this implies that for each value of δ_3, the subsequent LFSR sequence will change linearly in a way that depends only on this value. This means that it can be expressed as the bitwise XOR of the LFSR sequence corresponding to $\{n_R\}$ and a binary sequence that depends only on the known value of δ_3. For each of 8 values of δ_3, including the all-zero value, the attacker can use the 4 keystream bits resulting from the corresponding successful query. Then, in the off-line stage, an adapted variant [4] of the well-known resynchronization attack [5,11], where the IV corresponds to δ_3, can be used to obtain about 2^{16}

candidates for the LFSR state at the time when the last of the 4 keystream bits is generated. The relation with the resynchronization attack is not noticed in [4]. A detailed description is given in Appendix C. The 2^{16} candidate keys resulting from the LFSR rollback are then tested on other keystream data already collected in the on-line stage (i.e., 64 keystream parity bits), to produce the correct key. The total off-line time complexity of about 2^{16} steps takes practically zero time on a standard CPU. According to [4], the attack succeeds with probability about 0.75. In order to reconstruct the key, both on-line and off-line stages of the attack need to be repeated about $4/3 \approx 1.33$ times on average.

Our criticism of the differential attack [4] concerns the probability of the exploited differential property of f and the way the queries with random or fixed tag nonces are performed in the on-line stage.

Consider a general case where the last m, $m \leq 32$, bits of $\{n_R\}$ are changed. Then the last m bits of n_R obtained by the bitwise decryption of $\{n_R\}$ will change in the same way if the corresponding m keystream bits used for the decryption do not change. The fact overlooked in [4] is that the first (leftmost) of these m keystream bits does not change necessarily, because it depends only on the previous bits of $\{n_R\}$, which are not changed. The remaining $m - 1$ keystream bits are generated as $m - 1$ successive output bits of the filter function shifted along the LFSR sequence, i.e., by the $(m-1)$-bit augmented filter function of all the bits contained in $m-1$ (overlapping) successive LFSR states, which form the input to the augmented filter function. Therefore, any change of the last m bits of $\{n_R\}$ will result in the same change of the last m bits of n_R if and only if the $(m-1)$-bit output of the $(m-1)$-bit augmented filter function is independent of the last $m - 1$ input bits. Let π_{m-1} denote this probability, over the uniformly distributed inputs. In the attack [4], $m = 3$ and the relevant probability is then π_2. In [7], it is proved that $\pi_1 = 29/32 \approx 0.906$, where the 1-bit augmented filter function is f expressed as a function of all 48 LFSR state bits. Since the input bits to f that are effectively used for generating two successive output bits are distinct, it follows that $\pi_2 = (29/32)^2 \approx 0.821$. This is the correct probability for the differential attack [4], not 0.75.

It is interesting to note that $\pi_m = 0$ would hold for all m if f were linear in the first or the last input variable, i.e., it f satisfied the sufficient condition [8] for pure randomness of the output sequence. In other words, if f had satisfied this condition, then the considered differential attack would have been impossible. On the other hand, $\pi_m = 1$ would hold if f did not effectively depend on the last m LFSR state bits. It is fair to say that in CRYPTO1, f effectively depends on the last LFSR state bit and, hence, $\pi_m < 1$ holds for all m.

In the first phase of the on-line stage of the attack, to get one successful query, the fake reader can perform queries with a fixed or random tag nonce, where, on average, 128 queries are required in the former case and 256 in the latter. However, as mentioned above, the difference in the timings is significant. If q_r and q_f denote the timings of the queries using random and fixed tag nonces, respectively, then, according to [7], $q_f \approx 50q_r$, $1500q_r \approx 1\,\text{sec}$, and $30q_f \approx 1\,\text{sec}$. Accordingly, $256q_r \approx 5.12q_f \ll 128q_f$. Surprisingly, in [4], it is proposed to use queries with a fixed tag nonce. Moreover, if the attack fails in the first run, then, in order to reduce the number of queries in repeated runs, it is suggested in [4] to keep n_T and the first two bytes of $\{n_R\}$ the same as in the first run, which implies that the first two bits of $\{p\}$ will also be the same. In this case, on average, 32=64/2 instead of 128 queries with a fixed tag nonce are required in the repeated runs. The average on-line time of the attack is thus estimated to be about $256q_f$ in the first run and $160q_f$ in the repeated runs. Since the off-line time is negligible, the average time required for the success is estimated to be $(128 + 32/3 + (4/3)(8 \cdot 16))q_f \approx 310q_f \approx 10.33\,\text{sec}$.

It follows that this differential attack can be simply optimized by using the queries with a random tag nonce in the first phase of all the runs. Also, in other phases of the attack, further 16 queries with a fixed tag nonce need to be performed not 8 times, as proposed in [4], but only 7 times, for each nonzero 3-bit modification δ_3 of $\{n_R\}$. The average on-line time of each run then becomes $256q_r + (7 \cdot 16)q_f \approx 117.12q_f \approx 3.9\,\text{sec}$. The average time required for the success of the optimized attack Attack 4* is then about $117.12q_f/\pi_2 \approx 4.75\,\text{sec}$, since the attack needs to be repeated $1/\pi_2$ times on average.

We now propose a novel key reconstruction attack, denoted as Attack 5 in Table 1. It is in spirit similar to the differential Attack 4*, but requires a considerably smaller number of queries in the on-line stage, which is a bottleneck of tag-only attacks, and has a considerably higher probability of success. This is achieved at the cost of increasing the off-line time complexity, which nevertheless remains practically low. Our main insight making the tradeoff possible is that m can be reduced from 3 to 2 bits, according to the general considerations given above. The success probability then increases from $\pi_2 \approx 0.821$ to $\pi_1 \approx 0.906$.

The first phase of the on-line stage of the attack is the same as in Attack 4*. The fake reader on average makes 256 queries with random $\{n_T\}$, $\{n_R\}$, $\{a_R\}$, and $\{p\}$, in order to get one successful query. The fake reader then proceeds by making new queries using fixed n_T, modified and then fixed $\{n_R\}$, randomly chosen and then fixed $\{a_R\}$, and different $\{p\}$, where the first 3 bits of $\{p\}$ are kept the same and only the last 5 bits are varied. The modification of $\{n_R\}$ consists in changing its last $m = 2$ bits. On average, 16=32/2 such queries with a fixed n_T are needed for a successful query to occur. Since this has to be repeated 3 times, for all 3 nonzero 2-bit changes δ_2 of $\{n_R\}$, a total of $48 = 3 \cdot 16$ queries with a fixed n_T are needed on average. The average on-line time of the attack is then $256q_r + (3 \cdot 16)q_f \approx 53.12q_f \approx 1.77\,\text{sec} \approx 1.8\,\text{sec}$, which is more than 2 times smaller than in Attack 4*. Each of the 4 successful queries achieved in the on-line stage provides 12 bits of information about the 48-bit key in the form of 4 keystream bits and 8 keystream parity bits, to be used in the off-line stage.

With probability $\pi_1 \approx 0.906$, each nonzero 2-bit change δ_2 of $\{n_R\}$ will result in the same change of the last 2 bits of n_R. This means that the subsequent LFSR sequence can then be expressed as the bitwise XOR of the LFSR sequence corresponding to $\{n_R\}$ and a binary sequence that depends only on the known value of δ_2. Then, in the off-line stage, about 2^{32} candidates for the LFSR state at the time when the last of the 4 keystream bits resulting from each successful query is generated can be obtained by a variant of the resynchronization attack [5,11], where the IV corresponds to δ_2. A detailed description is given in Appendix C. The 2^{32} candidate keys resulting from the LFSR rollback are then tested on the $32 = 4 \cdot 8$ keystream parity bits collected in the on-line stage, to produce the correct key or a very small number of candidates. The total off-line time complexity of about 2^{32} steps takes about 5 minutes on a standard CPU. The attack thus succeeds with probability about $\pi_1 \approx 0.906$.

For reducing the number of candidate keys to only 1, the attacker may also make a small number of additional queries in the on-line stage in order to collect more bits of information about the key. For example, to obtain additional 12 bits of information, the attacker needs another successful query, which may be achieved in the random n_T scenario with an average of 256 additional queries. This effectively increases the on-line time to $58.24q_f \approx 1.94$ sec and is still about 2 times smaller than in Attack 4^*. Alternatively, the attacker can find the unique key at any later time in AttAR scenario, by accessing a genuine reader and then testing the candidate keys on the keystream segment ks_2 reconstructed from n_T and $\{a_R\}$. More precisely, for each assumed key, the attacker generates a 32-bit keystream segment by using n_T and $\{n_R\}$ and then compares it with ks_2.

If the attack fails, then the strategy of repeating the whole attack (on-line and off-line stages) as many times as needed until the key is reconstructed may not be practical, due to the off-line time of about 5 minutes. In any case, the attack would need to be repeated only about $\pi_1 \approx 1.1$ times on average. It is hence preferable for the attacker to decide in advance on the success probability, repeat the on-line stage a number of times that guarantees this probability, and only then perform the off-line stage using all the collected data. To obtain the success probability of at least 0.99, the on-line stage needs to be performed only twice, which takes about 3.5 seconds. In this case, additional successful queries are not needed for testing the candidate keys, because the on-line stage performed twice already provides the sufficient keystream. The off-line stage takes about 5 minutes with probability 0.906 or about 10 minutes otherwise, which is still about 5 minutes on average and thus remains very practical. For comparison, to achieve the success probability of at least 0.99, Attack 4^* needs to be repeated three times, which takes about 11.7 seconds, whereas the off-line stage can be done practically instantly. The on-line stage in the new Attack 5, with $m = 2$, is thus more than 3 times faster than in the optimized Attack 4^*, with $m = 3$.

Properties of the five presented tag-only attacks are summarized in Table 1. Times are given both in appropriate steps (clear from the context) and in time units, where off-line times relate to a standard CPU. TLU denotes a table lookup operation and f_{ev} denotes one evaluation of f.

6 Multiple Sector Attacks

The cryptanalytic attacks described in previous sections relate to the mutual authentication protocol for one sector. In this case, the tag nonce n_T is sent

Table 1. Summary of tag-only attacks

	Attack 1 [7]	Attack 2 [7]	Attack 3 [7]	Attack 4* [4]	**Attack 5** this paper
Setup time	0	0	2^{48}	0	0
Setup memory	0	0	$48 \cdot 2^{36}$ bit 384 GByte	0	0
On-line time	$1280q_r$ 1 sec	$28500q_f$ 15 min	$4230q_r + 128q_f$ 7 sec	$3(256q_r + 112q_f)$ 11.7 sec	$2(256q_r + 48q_f)$ 3.5 sec
Off-line time	$5 \cdot 2^{48}$ 3 year	$2^{32.8}$ 10 min	2^{24} TLU 20 sec	$2^{16} + 2^{26} f_{ev}$ ≈ 0	$2^{32} + 2^{25} f_{ev}$ 5 min
Off-line memory	≈ 0	≈ 0	≈ 0	168 Byte	42 KByte
Success rate	100%	100%	100%	99.4%	99.1%

in the clear form, which can be used for known keystream attacks (except in the tag-only scenario). As explained in Section 2, for authenticating access to multiple sectors of the tag in the same session, the same protocol is repeated with a difference that, for each new sector, the new authentication command (Block) is encrypted with the previous sector key and the new n_T is encrypted with the preceding keystream segment ks_0 generated from the new key. Therefore, n_T has to be guessed by the attacker.

Assume that the key for the first sector has been recovered by the attacker by one of the attacks described in Sessions 4 and 5, in AttP, AttAR, or AttAT scenario. The objective of the attacker is then to reconstruct the secret keys for other sectors. To this end, it is required to gather data from an authentication session for multiple sectors. This is possible in AttP and AttAT scenarios. AttAR scenario is not useful for the attacker, because a fake tag cannot encrypt a new n_T with the correct key for the next sector and the attacker cannot then obtain any correct keystream segment generated from the key for the next sector.

In AttP scenario [6], the attacker intercepts and records a genuine authentication session for multiple sectors, between a genuine tag and a genuine reader. Since the key for the first sector is already reconstructed, the attacker determines the next sector by decrypting the new authentication command. In AttAT scenario [7], a fake reader initiates a fake authentication session for multiple sectors with a genuine tag and repeatedly uses the previously reconstructed key for the first sector. The attacker is then successfully authenticated for the first sector and then proceeds with the authentication for the next sector. This is achieved by sending the authentication command (Block) encrypted with the key for the first sector, which is then successfully decrypted by the tag. In either case, the attacker obtains a correct keystream segment generated from the key for the next sector, on the condition that n_T can be effectively guessed. In AttAT scenario, the keystream segment length is 32 bits, from $n_T \oplus ks_0$, while in AttP scenario, the keystream segment length is 96 bits, from $n_T \oplus ks_0$, $a_R \oplus ks_2$, and $a_T \oplus ks_3$. The key for the next sector can then be easily reconstructed by using the techniques described in Section 4. The keys of other sectors can be reconstructed analogously, by proceeding one sector at a time. In [6], this attack is called the nested authentication attack.

The tag nonce can be guessed easily not only because it contains at most 16 bits of entropy and its effective entropy depends only on the imprecision of timing, but also because of the way the parity bits are encrypted, as pointed out in Section 2. Namely, any pair of ciphertext bits, one bit corresponding to the parity bit of the current byte and the other to the first bit of the next byte, reveals a linear relation among the plaintext bits due to the fact that the two keystream bits are the same. Each such relation reduces the plaintext uncertainty by 1 bit. More details can be found in Appendix D.

7 Conclusions

Attacks on the MIFARE Classic protocol are made possible by repeatable and predictable tag nonces, the weak structure of the nonlinear filter generator in CRYPTO1, the way the parity bits for error detection are genarated, and the fact that the tag can sent out an encrypted response even if the authentication of the reader fails. It is shown that the TMDT attack [6] in AttAR scenario can be regarded as an adaptation of the generic TMDT attack [1,9], whereas the inversion attack [6] in AttP or AttAR scenario can be regarded as an adaptation of the inversion attack with the decimation technique proposed in [8,10] for attacking nonlinear filter generators. The easiest attacks to implement are the tag-only attacks, in AttAT scenario, where a fake, emulated reader has a wireless access to a genuine tag in the on-line stage of the attack. Their main limitation factor in practice is the on-line time required. It is pointed out that each of the three attacks from [7] has serious practical limitations, namely, Attack 1 has huge off-line time, Attack 2 has very large on-line time of about 15 min, and Attack 3 has huge precomputation time. It is shown that by using the queries with random tag nonces to obtain different tag nonces, the on-line time of Attack 3 can be reduced from about 2 minutes to about 7 seconds.

The best known attack in the tag-only scenario is the differential attack [4], Attack 4, which is claimed to take about 10 seconds of average on-line time in order to reconstruct the secret key for one sector of the card. A correct analysis of this attack demonstrates that it can be optimized into Attack 4*, to reduce the average on-line time to about 4.75 seconds. On the basis of the conducted analysis, a new attack of a similar, differential type, denoted as Attack 5, is also proposed. It achieves a success probability of about 0.906 with the on-line time of about 1.8 seconds, whereas the optimized differential attack, Attack 4*, has the success probability of about 0.821 with the on-line time of about 3.9 seconds. The success probability and the on-line time of two independent runs of Attack 5 are about 0.991 and 3.5 seconds, respectively. For three independent runs of Attack 4*, they are about 0.994 and 11.7 seconds, respectively. This significant improvement is achieved at the cost of increasing the off-line time to about 5 minutes, which still remains very practical. It is explained that the off-line stage of both differential attacks can be regarded as an adaptation of the resynchronization attack [5,11].

As the worldwide MIFARE Classic infrastructure cannot be changed easily, the most effective countermeasure [4] against the tag-only attacks is putting the cards in electromagnetic-shield covers.

References

1. Babbage, S.: A space/time tradeoff in exhausting search attacks on stream ciphers. In: Proc. European Convention on Security and Detection, IEE Conference Publication No. 408, pp. 161–166 (May 1995)
2. Biryukov, A., Shamir, A.: Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 1–13. Springer, Heidelberg (2000)
3. Courtois, N.T., Nohl, K., O'Neil, S.: Algebraic attacks on the Crypto-1 stream cipher in MiFare Classic and Oyster cards. Cryptology ePrint Archive, Report 2008/166 (2008)
4. Courtois, N.T.: The darkside of security by obscurity - and cloning MiFare Classic rail and building passes, anywhere, anytime. In: Proc. Secrypt 2009, pp. 331–338 (2009)
5. Daemen, J., Govaerts, R., Vandewalle, J.: Resynchronization Weaknesses in Synchronous Stream Ciphers. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 159–167. Springer, Heidelberg (1994)
6. Garcia, F.D., de Koning Gans, G., Muijrers, R., van Rossum, P., Verdult, R., Wichers Schreur, R., Jacobs, B.: Dismantling MIFARE Classic. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 97–114. Springer, Heidelberg (2008)
7. Garcia, F.D., van Rossum, P., Verdult, R., Wichers Schreur, R.: Wirelessly pickpocketing a Mifare Classic card. In: Proc. 30th IEEE Symposium on Security and Privacy, Oakland, pp. 3–15 (2009)
8. Golić, J.Dj.: On the Security of Nonlinear Filter Generators. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 173–188. Springer, Heidelberg (1996)
9. Golić, J.Dj.: Cryptanalysis of Alleged A5 Stream Cipher. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 239–255. Springer, Heidelberg (1997)
10. Golić, J.Dj., Clark, A., Dawson, E.: Generalized inversion attack on nonlinear filter generators. IEEE Trans. Comput. C-49, 1100–1109 (2000)
11. Golić, J.Dj., Morgari, G.: On the Resynchronization Attack. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 100–110. Springer, Heidelberg (2003)
12. de Koning Gans, G., Hoepman, J.-H., Garcia, F.D.: A Practical Attack on the MIFARE Classic. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 267–282. Springer, Heidelberg (2008)
13. Kumar, S., Paar, C., Pelzl, J., Pfeiffer, G., Schimmler, M.: Breaking Ciphers with COPACOBANA –A Cost-Optimized Parallel Code Breaker. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 101–118. Springer, Heidelberg (2006)
14. Nohl, K., Evans, D., Starbug, Plötz, H.: Reverse-engineering a cryptographic RFID tag. In: Proc. USENIX Security 2008, pp. 185–193 (2008)
15. Proxmark III instrument, HW and SW, http://cq.cx/proxmark3.pl

A TMDT Attack and Adapted Inversion Attack from [6]

TMDT attack uses a precomputed table with 2^{48-x} entries (S_{64}, ks_2, ks_3), sorted by (ks_2, ks_3), and 2^x keystream segments, either (ks_2, ks_3) or only ks_2, obtained from 2^x (partial) authentication sessions with the genuine reader. For $x = 12$, the precomputation time and memory complexities are thus both proportional to 2^{36}. In the on-line stage, 4096 authentication sessions are needed for collecting the keystream data, which takes about 10 min. In the off-line stage, the key can be reconstructed by 4096 table lookup operations, on a hard disk, and by the LFSR rollback, i.e., almost instantly, provided that 64-bit keystream segments are used. With 32-bit keystream segments, the 4096 table lookups will, on average, yield 2^{16} $(2^{48}/2^{32})$ candidate states S_{64}, instead of only one. To get the correct key from the resulting 2^{16} candidate keys, one more authentication session is needed. We note that the required number of authentication sessions can be reduced at the expense of increasing the precomputation time, while keeping the same computation time and memory, by applying the TMDT attack [2].

Let M be the input memory size of a nonlinear filter generator defined as the difference between the last and the first tap position to the filter function (in CRYPTO1, $M = 38$). Then the time complexity of the generic inversion attack is 2^M steps. Further, let d be the greatest common divisor of the pairwise differences between the tap positions (in CRYPTO1, $d = 2$). Then the input memory size can be reduced d times by looking at d-decimations of d successive phase shifts of the LFSR sequence.

In order to deal with short keystream segments corresponding to decimated sequences, which are shorter than the LFSR length, one has to consider different phase shifts of the LFSR sequence simultaneously. This can be achieved by the technique [6] based on partitioning the LFSR recursion into $d = 2$ linear terms corresponding to 2-decimations of two successive phase shifts of the LFSR sequence. The values of these terms are computed along the decimated LFSR segments obtained by the inversion attack and then stored in two sorted tables, each containing about 2^{19} entries. The matches between the entries in the two tables then yield the reconstruced LFSR segments by interleaving. For 64-bit keystream segments, the time complexity is thus at most 2^{17} steps. For 32-bit keystream segments, the time complexity is at most 2^{20} steps and the number of candidate keys obtained is about 2^{16}, so that one more authentication session is needed to get the correct key. Altogether, it takes about 0.05 seconds and 8 MByte of memory on a standard CPU to recover the key.

B Three Tag-Only Attacks from [7]

Attack 1 is a brute-force attack that exhaustively tests all 2^{48} keys on the keystream data obtained from 4-5 successful queries with a random tag nonce, which provide 48-60 bits of entropy for the 48-bit key. To get one successful query, it is on average needed to perform 256 queries with a random nonce tag. The on-line stage thus takes less than 1 second to perform about 1280 queries with a random tag nonce.

Attack 2 uses a relatively large number of queries with a fixed tag nonce, i.e., on average, about 28500 such queries. The on-line stage thus takes about 15 minutes. The objective is to find a special value of the 32-bit encrypted reader nonce by adaptively choosing its values for a given value of the tag nonce. This is achieved by an iterative process with backtracking, each time changing the last bit of one byte of the encrypted reader nonce and checking a condition on the corresponding encrypted correct parity bits, upon obtaining the respective two successful queries by repeatedly changing the encrypted parity bits. The special value of the encrypted reader nonce, satisfying the condition on all four bytes, significantly reduces the number of possible LFSR states at a given time and thus reduces the number of candidate keys to about $2^{32.8}$. They are then checked in the off-line stage by using the keystream data already collected in the on-line stage, to find the correct key. The off-line stage approximately requires 10 minutes on a standard CPU.

Attack 3 uses a large partitioned table of about 384 GByte, stored on a hard disk, which is precomputed in time equivalent to that of the brute-force attack. It stores about 2^{36} LFSR states at a given time that result in the all-zero 32-bit encrypted reader nonce, all-zero 32-bit encrypted answer to the tag nonce, all 8 encrypted parity bits equal to zero, as well as 4 keystream bits encrypting the error message, all equal to zero. In the on-line stage, the attacker should make about 4096 queries with different tag nonces in order to satisfy the conditions for the LFSR state to belong to the precomputed table of 2^{36} entries. In [7], it is claimed that this requires about 2 minutes of on-line time, which is true if these queries use a fixed tag nonce. However, in view of the fact that the tag nonce can take 2^{16} different values, it follows that only 4230 queries with random tag nonces are on average sufficient to get 4096 different tag nonces. In addition, the attacker then makes about 128 queries with a fixed tag nonce in order to further restrict the LFSR state to a subtable of 2^{24} entries. The on-line stage can thus take only about 7 seconds, not 2 minutes as claimed in [7]. In the off-line stage, the resulting 2^{24} candidate keys are then tested on the keystream data already collected in the on-line stage, to produce the correct key. The off-line complexity of the attack is thus determined by 2^{24} table lookup operations, which takes less than about 20 seconds on a standard CPU.

C Adapted Resynchronization Attacks

The resynchronization attack [5,11] is applicable to stream ciphers with a linear next-state function, a nonlinear Boolean output function f depending on a relatively small or moderately large number of bits, n, and a linear reinitialization algorithm combining a k-bit secret key with an IV (initialization vector). Due to the linearity, the input to the output function at any given time can be linearly decomposed into two components, one depending on the key and the other depending on the IV. The attack then essentially consists in solving the corresponding equations of the form $z_t^i = f(X_t \oplus C_t^i)$, for a number of time instants t, where X_t is a linear function of the key and C_t^i is a linear function of the IV^i.

For each t, the solution is found by the exhaustive search. Accordingly, the key can be reconstructed from about n different IVs, by observing the outputs of f at about k/n time instants, and by evaluating f about $k2^n$ times.

Consider first the off-line stage of the differential Attack 4 [4] or Attack 4*. The 4 keystream bits resulting from each of 8 successful queries are the first 4 keystream bits of ks_3, i.e., z_i, $96 \leq i \leq 99$. Accordingly, for each such keystream bit z_i, the following 8 equations hold simultaneously with probability π_2: $z_i(\delta_3) = f(S_i \oplus \Delta_{3,i})$, where δ_3 is varied over all 8 possible values, $\Delta_{3,i}$ is a 48-bit vector depending only on i and δ_3 ($\Delta_{3,i} = 0$ if $\delta_3 = 0$), and S_i is the LFSR state at time i for $\delta_3 = 0$, for given n_T and $\{n_R\}$. Due to the fact [4] that f depends only on 20 bits of the state and that after 2 steps, f depends on the same 19 bits and 1 new bit, z_{96} and z_{98} (as well as z_{97} and z_{99}) depend on 21 LFSR bits and we have 16=8+8 nonlinear equations involving these 21 bits. As in the resynhcronization attack, by the exhaustive search over these 21 bits, each time evaluating f 16 times, the attacker finds and stores all 21-bit inputs that are consistent with these 16 equations. On average, the number of possible values for the 21 bits is thus $2^5 = 2^{21-16}$. In the same way, the attacker gets and stores about 2^5 possibilities for the other 21 LFSR sequence bits determining z_{97} and z_{99}. The two tables of 2^5 entries are thus computed in 2^{22} steps, each consisting of 16 evaluations of f, that is, almost instantly on a standard CPU.

Altogether, the attacker thus obtains about 2^{10} possibilities for the corresponding 42 successive bits of the LFSR sequence. By guessing the remaining 6 bits, the attacker thus obtains 2^{16} possibilities for the LFSR state S_{99}. By exhaustively examining all 2^{16} found states S_{99}, for given n_T and $\{n_R\}$, 2^{16} candidate keys are thus recovered via the LFSR rollback. The 2^{16} candidate keys are then tested on $64 = 8 \cdot 8$ keystream parity bits already collected in the on-line stage of the attack to yield the correct key. In fact, only 24 keystream parity bits obtained from 3 successful queries are sufficient. This can be done almost instantly on a standard CPU. The total off-line time of the attack is thus practically zero. The attack succeeds with probability $\pi_2 \approx 0.821$, and fails if no candidate key survives the testing.

Consider now the off-line stage of the new differential Attack 5. For each such keystream bit z_i, $96 \leq i \leq 99$, the following 4 equations hold simultaneously with probability π_1: $z_i(\delta_2) = f(S_i \oplus \Delta_{2,i})$, where δ_2 is varied over all 4 possible values, $\Delta_{2,i}$ is a 48-bit vector depending only on i and δ_2 ($\Delta_{2,i} = 0$ if $\delta_2 = 0$), and S_i is the LFSR state at time i for $\delta_2 = 0$, for given n_T and $\{n_R\}$. Now, similarly as in Attack 4, since there are 4 instead of 8 linear equations for each of these 4 keystream bits, the attacker then finds and stores $2^{13} = 2^{21-8}$ possibilities for each of the two interleaved 21-bit parts of the state S_{99}, in 2^{22} steps, each step consisting of 8 evaluations of f. The total memory required is then 42 KByte. The off-line time of this part of the attack, requiring 2^{25} evaluations of f, is practically zero. The attacker then exhaustively examines $2^{32} = 2^{26+6}$ possibilities for the LFSR state S_{99}, for given n_T and $\{n_R\}$, by recovering 2^{32} candidate keys via the LFSR rollback. The 2^{32} candidate keys are then tested on $32 = 4 \cdot 8$ keystream parity bits already collected in the on-line stage of the attack. This should suffice to reduce the number of candidate keys to only 1 or a very small number. The required 2^{32} steps of the off-line stage can be performed in about 5 minutes or less on a standard CPU. The attack succeeds with probability $\pi_1 \approx 0.906$, and fails if no candidate key survives the testing.

D Nested Authentication Attacks from [6,7]

It is here explained how the attacker can effectively guess a correct value of n_T used in the authentication protocol for the next sector, in the nested authentication attacks [6,7], where [6] deals with AttP scenario and [7] with AttAT scenario. There are only 2^{16} values of n_T to start with, due to the 16-bit LFSR used for generating 32-bit tag nonces. In fact, as pointed out in [6], the uncertainty is much smaller, because n_T is generated almost deterministically from the previously reconstructed value of the tag nonce, where the only residual uncertainty relates to the imprecision of timing. Another weakness which further reduces the uncertainty is that any pair of ciphertext bits encrypting the parity bit of the current byte and the first bit of the next byte reveals a linear relation among the plaintext bits due to the fact that the two keystream bits are the same. Each such relation reduces the plaintext uncertainty by 1 bit. Therefore, the initial uncertainty of n_T given $n_T \oplus ks_0$ is 13 (16-3) bits.

In AttP scenario, the uncertainty of n_T, given $n_T \oplus ks_0$, $a_R \oplus ks_2$, and $a_T \oplus ks_3$ is only 6 (16-10) bits, so that the attacker needs to check at most 64 values of n_T. This number is effectively much smaller in practice, as pointed out above. For any guessed value of n_T, the key for the next sector can be reconstructed from the corresponding 96-bit keystream segment (ks_0, ks_2, ks_3) by the techniques from Section 4, provided that the guess is correct. If the guess is incorrect, no 48-bit key can be reconstructed, because the corresponding 96-bit keystream segment will be incorrect. The keys of other sectors can be reconstructed analogously from already recovered keys of the previous sectors.

In AttAT scenario, as pointed out in [7], the attack proceeds along similar lines, with a difference that only the encryption $n_T \oplus ks_0$ can be used. This implies that at most 2^{13} values of n_T need to be checked, which may be significantly reduced by using the measured timing between two successive authentication rounds. For each guessed value of n_T, about 2^{16} candidate keys for the next sector can be reconstructed from the corresponding 32-bit keystream segment ks_0 by the techniques described in Section 4. More precisely, the techniques need to be slightly adapted to the fact that during the generation of ks_0, $n_T \oplus ID$ is shifted in the LFSR. The correct key for the next sector is then found by using another fake authentication session for multiple sectors initiated by the fake reader (or at most two such sessions), by computing the intersections between the obtained sets of candidate keys. With a high probability, only the correct guesses of the tag nonces will result in a unique value of the key, whereas for incorrect guesses, the intersections will be empty. The keys of other sectors can be reconstructed analogously. This means that in AttAT scenario, the keys of other sectors can be reconstructed much easier than for the first sector.

Asynchronous Computational VSS with Reduced Communication Complexity[⋆]

Michael Backes[1], Amit Datta[2], and Aniket Kate[3]

[1] Saarland University and MPI-SWS, Germany
backes@mpi-sws.org
[2] Carnegie Mellon University, U.S.A.
amitdatta@cmu.edu
[3] MMCI, Saarland University, Germany
aniket@mmci.uni-saarland.de

Abstract. Verifiable secret sharing (VSS) is a vital primitive in secure distributed computing. It allows an untrusted dealer to verifiably share a secret among n parties in the presence of an adversary controlling at most t of them. VSS in the synchronous communication model has received tremendous attention in the cryptographic research community. Nevertheless, recent interest in deploying secure distributed computing over the Internet requires going beyond the synchronous model and thoroughly investigating VSS in the asynchronous communication model.

In this work, we consider the communication complexity of asynchronous VSS in the computational setting for the optimal resilience of $n = 3t + 1$. The best known asynchronous VSS protocol by Cachin et al. has $O(n^2)$ message complexity and $O(\kappa n^3)$ communication complexity, where κ is a security parameter. We close the linear complexity gap between these two measures for asynchronous VSS by presenting two protocols with $O(n^2)$ message complexity and $O(\kappa n^2)$ communication complexity. Our first protocol satisfies the standard VSS definition, and can be used in stand-alone VSS scenarios as well as in applications such as Byzantine agreement. Our second and more intricate protocol satisfies a stronger VSS definition, and is useful in all VSS applications including multiparty computation and threshold cryptography.

Keywords: Verifiable Secret Sharing, Asynchronous Communication Model, Communication Complexity, Polynomial Commitments.

1 Introduction

The notion of secret sharing was introduced independently by Shamir [24] and Blakley [6] in 1979. For integers n and t such that $n > t \geq 0$, an (n, t)-*secret sharing* scheme is a method used by a *dealer* to share a secret s among a set of n parties in such a way that any subset of $t + 1$ or more parties can compute the secret s, but subsets of size t or fewer cannot.

[⋆] An extended version of this paper is available [3].

In many applications of secret sharing, parties may need to verify the correctness of the values dealt in order to prevent malicious behavior by the dealer. To satisfy this requirement, Chor et al. [12] introduced the concept of *verifiable secret sharing* (VSS). With its applicability to Byzantine agreement, multiparty computation (MPC) and threshold cryptography, VSS has remained an important area of cryptographic research for the last two decades [9,13,15,16,21,22].

Although the literature for VSS is vast, the notion of VSS in the asynchronous communication setting (no bounds on message transfer delays) has not yet received the deserved attention in terms of practical efficiency or theoretical lower bounds. Asynchronous VSS schemes with unconditional security have been developed [1,5,11,20]; however, these schemes are prohibitively expensive for any realistic use as they need $\Omega(\kappa n^5)$ bits of communication for κ-bit secrets. In the computational security setting, Cachin et al. [9], Zhou et al. [25], and recently Schultz et al. [23] suggested more practical asynchronous VSS schemes: asynchronous verifiable secret sharing (AVSS), asynchronous proactive secret sharing (APSS) and mobile proactive secret sharing (MPSS), respectively. Of these, AVSS [9] is the most generic and practical asynchronous VSS scheme and it forms the basis for many practical threshold cryptographic protocols such as [17]. AVSS assimilates a bivariate polynomial into Bracha's deterministic reliable broadcast protocol [8], which results into its $O(n^2)$ message complexity (number of messages transferred) and $O(\kappa n^4)$ communication complexity (number of bits transferred) for the optimal resiliency condition of $n = 3t+1$. Cachin et al. [9] further refined the AVSS protocol to reduce the communication complexity to $O(\kappa n^3)$. Nevertheless, a further reduction in the communication complexity is not possible using similar techniques, and a linear complexity gap between the message complexity and the communication complexity still remains.

In this work, we bridge this gap. We present two *efficient* asynchronous VSS schemes (eAVSS and eAVSS-SC) with different properties (and correspondingly different utilities) with $O(n^2)$ message complexity and $O(\kappa n^2)$ communication complexity.

1.1 Our Contributions

Kate, Zaverucha and Goldberg [18] define the concept of commitments to polynomials, and devise two schemes $\mathsf{PolyCommit}_{\mathrm{DLog}}$ and $\mathsf{PolyCommit}_{\mathsf{Ped}}$ that commit to a univariate polynomial of degree t (or less) using a single element of size $O(\kappa)$. Their schemes work in the bilinear pairing setting under the t-strong Diffie–Hellman (t-SDH) assumption [7]. We use their $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme and a collision-resistant hash function to achieve our goal of asynchronous VSS with $O(\kappa n^2)$ communication complexity. Although we choose the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme that provides unconditional hiding (secrecy) instead of the much simpler $\mathsf{PolyCommit}_{\mathrm{DLog}}$ scheme that provides computational hiding against the discrete logarithm (DLog) assumption, our protocols work with the $\mathsf{PolyCommit}_{\mathrm{DLog}}$ scheme with no modification.

Nevertheless, the schemes we present are not a straightforward adaptation of the PolyCommit schemes to the bivariate polynomial-based AVSS scheme [9], and not surprisingly, Kate et al. [18] left the applicability of PolyCommit to asynchronous VSS as an open problem. The reason for that, as we elaborate in Section 2.4, is that modifying the PolyCommit schemes to a scheme providing constant-size commitments to bivariate-polynomials used in asynchronous VSS seems difficult if not impossible.

We achieve our goal by taking an entirely different path, bypassing the open problem of obtaining constant-size commitments to bivariate polynomials. We realize asynchronous VSS in two steps: We first present a univariate polynomial-based asynchronous VSS scheme (eAVSS), which guarantees that at least $t+1$ honest parties receive proper shares of the secret, while the remaining honest parties are assured that at least $t+1$ honest parties have received correct shares and can reconstruct the shared secret. This construction is sufficient for stand-alone VSS and for applications such as asynchronous Byzantine agreement (ABA). For applications such as MPC and threshold cryptographic constructions, we then design an efficient stronger asynchronous VSS scheme (eAVSS-SC), which guarantees that every honest party receives its share during the sharing phase. In principle, this is possible by running $n+1$ instances of eAVSS; however, it asks for a broadcast of commitment vectors of size $O(\kappa n)$ which increases the communication complexity to $O(\kappa n^3)$. In eAVSS-SC, we overcome this barrier by aptly modifying the AVSS protocol flow and by hashing the commitments in the vector using a collision-resistant hash function and running a PolyCommit instance over the hashed values.

Our schemes have direct implications to the efficiency of all asynchronous VSS applications. Most prominently, using our eAVSS protocol in the modular ABA construction by Canetti and Rabin [11] it is possible to obtain the first $O(\kappa n^3)$ communication complexity ABA protocol, which is secure against the *adaptive* adversary in the *standard* model.

Organization. In Section 2, we describe our system model and provide a brief overview of the concepts of VSS, polynomial commitments and asynchronous VSS. In Section 3, we define and prove our basic asynchronous VSS protocol (eAVSS), while in Section 4, we define our main asynchronous VSS protocol (eAVSS-SC). In Section 5, we discuss a few interesting applications. An in-depth discussion of the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme and corresponding computational assumptions have been added in Appendix A.

2 Preliminaries

Our schemes work in the computational security setting. The adversary $\mathcal{A}$ is a probabilistic polynomial time (PPT) algorithm with respect to a security parameter κ unless stated otherwise. A function $\epsilon(\cdot) : \mathbb{N} \rightarrow \mathbb{R}^+$ is called *negligible* if for all $c > 0$ there exists a κ_0 such that $\epsilon(\kappa) < 1/\kappa^c$ for all $\kappa > \kappa_0$. Throughout the rest of this paper, $\epsilon(\cdot)$ denotes a negligible function.

We assume that the shared secret s lies over a finite field $\mathbb{F}_p$, where p is a κ-bit long prime. We use Shamir's *polynomial-based* secret sharing approach [24], where our polynomials belong to $\mathbb{F}_p[x]$ or $\mathbb{F}_p[x, y]$.

2.1 Asynchronous System Model

Following the adversary and communication model of AVSS given by Cachin et al. [9], we assume an asynchronous fully-connected network of n parties $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, where every pair of parties is connected by an authenticated and private communication link. A special party $P_d \in \mathcal{P}$ works as a dealer. The indices for the parties are chosen from $\mathbb{F}_p$. Without loss of generality, we assume these indices to be $\{1, \ldots, n\}$.

The adversary $\mathcal{A}$ is *t-bounded* and it can coordinate the actions of up to t out of n parties. The adversary $\mathcal{A}$ is further assumed to be *adaptive*, and may corrupt a party of its choice at any instance during a protocol execution as long as its total number of corruptions is bounded by t. A party is said to be *honest* if the adversary has not corrupted it. In our asynchronous setting, the adversary $\mathcal{A}$ controls the network and may delay messages between any two honest parties. However, it cannot read or modify these messages, and it also has to eventually deliver all the messages by honest parties.

2.2 Verifiable Secret Sharing—VSS

In many secret sharing applications, a dealer may behave maliciously. This led to the conception of VSS [12].

Definition 1. *An (n, t)-*VSS *scheme among n parties in $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ with a distinguished party $P_d \in \mathcal{P}$ consists of two phases: the* sharing *(Sh) phase and the* reconstruction *(Rec) phase.*

Sh phase. *A dealer P_d distributes a secret $s \in \mathbb{F}_p$ among parties in $\mathcal{P}$. At the end of the Sh phase, each honest party P_i holds a share s_i of the distributed secret s.*

Rec phase. *In this phase, each party P_i sends its secret share s'_i to every party in $\mathcal{P}$ and a reconstruction function is applied in order to compute the secret $s = \mathsf{Rec}(s'_1, s'_2, \ldots, s'_n)$ or output $\perp$ indicating that P_d is malicious. For honest parties $s'_i = s_i$, while for malicious parties s'_i may be different from s_i or even absent.*

An (n, t)-VSS scheme has the following security requirements:

Secrecy. *If the dealer is honest, the adversary who can compromise t parties does not have any more information about s except what is implied by the public parameters.*

Correctness. *If P_d is honest, the reconstructed value should be equal to the dealer's secret s.*

Commitment. *Even if P_d is dishonest, there exists a value $s^* \in \mathbb{F}_p \cup \{\perp\}$ at the end of the Sh phase, such that all honest parties output s^* at the end of the Rec phase.*

In this paper, we consider VSS schemes where any malicious behaviour by P_d can be identified by the honest parties in the Sh phase itself and the commitment property simplifies to the following: the reconstructed value z should be equal to a shared secret $s \in \mathbb{F}_p$ that gets fixed at the end of the Sh phase.

Many VSS applications (e.g., threshold cryptography and MPC) avoid participation by all parties once the Sh phase is over. It is required that messages from any $t+1$ honest parties (or any $2t+1$ parties) are sufficient to reconstruct the shared secret s. For these applications, we require a stronger commitment property that we refer as the *strong commitment* requirement.

Strong Commitment. *Even if P_d is dishonest, there exists a value $s^* \in \mathbb{F}_p$ at the end of the Sh phase, such that s^* is reconstructed regardless of the subset of parties (of size greater than $2t$) chosen by the adversary in the Rec phase.*

Some VSS schemes achieve a weaker (computational) secrecy guarantee.

Weak Secrecy. *A t-*limited *adversary who can compromise t parties cannot compute s during the Sh phase.*

We also give the following definitions for the complexity measures.

Definition 2 (Message Complexity). *The message complexity is defined as the total number of messages exchanged between the parties participating in a scheme.*

Definition 3 (Communication Complexity). *The communication complexity is defined as the total number of bits exchanged between the parties taking into consideration every message that has been transmitted.*

A variant of VSS considers dealer P_d to be an external party (i.e., $P_d \notin \mathcal{P}$) and allows the adversary to corrupt P_d and up to t additional parties in $\mathcal{P}$. All our protocols also work in this stronger setting.

Assuming a broadcast channel, Feldman [15] gave the first non-interactive and efficient VSS scheme and Pedersen [21,22] presented a modification to it. Both protocols obtain the strong commitment property. In terms of secrecy, Feldman VSS achieves the weak secrecy property, while Pedersen VSS achieves the stronger form.

2.3 Use of Commitments in VSS

A verification mechanism for a consistent dealing is fundamental to VSS. It is achieved using distributed computing techniques in the information-theoretic security setting. In the computational setting that we focus in this paper, the commitment schemes provide an efficient alternative.

A commitment scheme allows an entity, the *committer*, to publish a value, called the *commitment* (say $\mathcal{C}$), which binds her to a message s (*binding*) without revealing it (*hiding*). Later, she may *open* the commitment $\mathcal{C}$ and reveal the committed message m to a verifier, who can check that the message is consistent with the commitment. In particular, the computational VSS schemes utilize the commitments to the shared polynomials. Kate et al. [18] formalize this concept of polynomial commitments. Here, we present a refined version of their polynomial commitment ($\mathsf{PolyCommit}$) definition for polynomial of degree $\leq t$.

Definition 4. *A* $\mathsf{PolyCommit}$ *scheme consists of the following algorithms:*

$\mathbf{Setup}(1^\kappa, t)$ *generates system parameters* SP *to commit to a polynomial of degree $\leq t$. In these system parameters, let $\mathbb{G}$ be an algebraic structure for commitments.* Setup *is run by a trusted or distributed authority.* SP *can also be standardized for repeated use.*

$\mathbf{Commit}(\mathbf{SP}, \phi(x))$ *outputs a commitment $\mathcal{C}$ to a polynomial $\phi(x)$ for the system parameters* SP*, and some associated decommitment information d. (In some constructions, d can be null.)*

$\mathbf{Open}(\mathbf{SP}, \mathcal{C}, \phi(x), [d])$ *outputs the polynomial $\phi(x)$ used while creating the commitment, with decommitment information d.*

$\mathbf{VerifyPoly}(\mathbf{SP}, \mathcal{C}, \phi(x), [d])$ *verifies that $\mathcal{C}$ is a commitment to $\phi(x)$, created with decommitment information d. If so, the algorithm outputs* 1*, otherwise it outputs* 0.

$\mathbf{CreateWitness}(\mathbf{SP}, \phi(x), i, [d])$ *outputs $\langle i, \phi(i), w_i, d_i\rangle$, where w_i is a witness and d_i is the decommitment information for the evaluation $\phi(i)$ of $\phi(x)$ at the index i. This algorithm is optional.*

$\mathbf{VerifyEval}(\mathbf{SP}, \mathcal{C}, i, \phi(i), [d_i, w_i])$ *verifies that $\phi(i)$ is indeed the evaluation at the index i of the polynomial committed in $\mathcal{C}$. If so, the algorithm outputs* 1*, otherwise it outputs* 0.

Given $\mathsf{SP} \leftarrow \mathsf{Setup}(1^\kappa, t)$, a $\mathsf{PolyCommit}$ scheme satisfies the following properties:

Correctness. *Let $\mathcal{C} \leftarrow \mathit{Commit}(\mathit{SP}, \phi(x))$. For $\mathcal{C}$ generated by $\mathit{Commit}(\mathit{SP}, \phi(x))$, and all $\phi(x) \in \mathbb{Z}_p[x]$, any $\langle i, \phi(i), w_i, d_i\rangle$ generated using $\mathit{CreateWitness}(\mathit{SP}, \phi(x), i)$ is correctly verified by $\mathit{VerifyEval}(\mathit{SP}, \mathcal{C}, i, \phi(i), d_i, , w_i)$.*

Strong Correctness. $\forall \mathcal{A} : \Pr\{(\mathcal{C}, \langle \phi(x), d\rangle) \leftarrow \mathcal{A}(\mathit{SP}) : \deg(\phi(x)) > t\} = \epsilon(\kappa)$.

Polynomial Binding. $\forall \mathcal{A} : \Pr\{(\mathcal{C}, \langle \phi(x), d\rangle, \langle \phi'(x), d'\rangle) \leftarrow \mathcal{A}(\mathit{SP})\} = \epsilon(\kappa)$ *given* $\big[(\mathit{VerifyPoly}(\mathit{SP}, \mathcal{C}, \phi(x), d) = 1) \wedge (\mathit{VerifyPoly}(\mathit{SP}, \mathcal{C}, \phi'(x), d') = 1) \wedge (\phi(x) \neq \phi'(x))\big]$

Evaluation Binding. $\forall \mathcal{A} : \Pr\{(\mathcal{C}, \langle i, \phi(i), d_i, w_i\rangle, \langle i, \phi(i)', d_i', w_i'\rangle) \leftarrow \mathcal{A}(\mathit{SP})\} = \epsilon(\kappa)$ *given* $\big[(\mathit{VerifyEval}(\mathit{SP}, \mathcal{C}, i, \phi(i), d_i, w_i) = 1) \wedge (\mathit{VerifyEval}(\mathit{SP}, \mathcal{C}, i, \phi(i)', d_i', w_i') = 1) \wedge (\phi(i) \neq \phi(i)')\big]$

(Unconditional) Hiding. *For $\phi(x) \in_R \mathbb{Z}_p[x]$, given $\langle \mathit{SP}, \mathcal{C}\rangle$ and $\{\langle i_j, \phi(i_j), d_{i_j}, w_{\phi_{i_j}}\rangle : j \in [1, \deg(\phi)]\}$ such that $\mathit{VerifyEval}\ (\mathit{SP}, \mathcal{C}, i_j, \phi(i_j), d_{i_j}, w_{\phi_{i_j}}) = 1$ for each j, a computationally unbounded adversary $\hat{\mathcal{A}}$ has no information about $\phi(\hat{j})$ for any unqueried index $\hat{j}$.*

The above strong correctness property is not present in the original $\mathsf{PolyCommit}$ definition. We include it as restricting degree of the committed polynomial by a threshold t is required for VSS. Further, a weaker form of hiding is also possible, where a computationally bounded adversary $\mathcal{A}$ cannot compute $\phi(\hat{j})$ for any unqueried index $\hat{j}$. We consider the unconditional hiding property in the paper.

In literature, VSS protocols utilized commitments to the *coefficients* or *evaluations* of shared polynomials as polynomial commitments. They used two commitment schemes. Given g and h as two random generators of a multiplicative group of order p, Feldman VSS and its variants use a commitment scheme of the form g^s with computational hiding under the discrete logarithm (DLog) assumption and unconditional binding. Pedersen [21] presented another commitment of the form $g^s h^r$ with unconditional hiding but computational binding under the DLog assumption. The hiding property of the commitment scheme leads to the secrecy property of VSS, while the binding property leads to the correctness property of VSS. Both of these commitment schemes also trivially satisfy the commitment property of VSS by the fact that the size of a commitment to a polynomial $\phi(x) \in \mathbb{Z}_p[x]$ is equal to $\deg(\phi) + 1$. In the complexity terms, the size of commitment is $O(n)$ (since for optimal resiliency, $\deg(\phi) = t = \lfloor \frac{n-1}{2} \rfloor$). However, the commitment to a shared polynomial has to be broadcast to all parties, which results in a linear-size broadcast for Feldman VSS, and a linear complexity gap between the message and the communication complexities.

Kate et al. [18] close this gap for Feldman VSS and its variants using a commitment that commits to the entire univariate polynomial using a single element. In particular, they define two polynomial commitment ($\mathsf{PolyCommit}$) schemes: $\mathsf{PolyCommit}_{\mathrm{DLog}}$ and $\mathsf{PolyCommit}_{\mathsf{Ped}}$, both of which works in the bilinear pairing setting with $\Theta(t)$ system parameters. $\mathsf{PolyCommit}_{\mathrm{DLog}}$ attains hiding under the DLog assumption, binding under the t-strong Diffie-Hellman (t-SDH) assumption [7], and strong correctness under the t-polynomial Diffie-Hellman assumption (refer to Appendix A for references to any assumption). Using a technique similar to Pedersen commitments, they also define $\mathsf{PolyCommit}_{\mathsf{Ped}}$, which attains unconditional hiding and computational binding under the t-SDH assumption. These constructions are based on an algebraic property of polynomials $\phi(x) \in \mathbb{F}_p[x]$ that $(x - i)$ *perfectly* divides the polynomial $\phi(x) - \phi(i)$ for any $i \in \mathbb{F}_p$.

In this work, we extend the utility of the $\mathsf{PolyCommit}$ concept to asynchronous VSS. We choose the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme for our protocol as it provides unconditional hiding and include the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ construction in Appendix A.

2.4 Asynchronous VSS

The asynchronous communication setting places no bounds on message delays. Consequently, there is no trivially available broadcast channel, and Feldman VSS and its variants do not guarantee a correct completion. This gives rise to the concept of asynchronous VSS for optimal resilience of $n = 3t + 1$.

An asynchronous VSS protocol requires the liveness and agreement properties along with the secrecy, correctness and commitment properties defined in Section 2.2

Definition 5. *An* asynchronous VSS *protocol having $n \geq 3t + 1$ parties with a t-limited Byzantine adversary satisfies the following conditions:*

Liveness. *If the dealer P_d is honest in the* Sh *phase, then all honest parties complete the* Sh *phase.*

Agreement. *If some honest party completes the* Sh *phase, then all honest parties will eventually complete the* Sh *phase. If all honest parties subsequently start the* Rec *phase, then all honest parties will complete the* Rec *phase.*

Correctness, Commitment and Secrecy. *as defined in Section 2.2.*

For VSS applications such as MPC, we need VSS that has identical secrecy, correctness, liveness and agreement properties as in Definition 5, but a stronger *commitment* property as defined in Section 2.2. In other words, there exists a t-degree polynomial $f(x)$ such that a share s_i held by every honest party P_i at the end of the sharing phase is equal to $f(i)$.

As discussed in the introduction, three computational VSS schemes have been suggested for the asynchronous setting: AVSS [9], APSS [25], and MPSS [23]. Of these, AVSS [9] provides the first and the most practical asynchronous VSS scheme. In the AVSS methodology, secret sharing is integrated into a reliable broadcast primitive [8]. This results into its $O(n^2)$ messages complexity. Here, the commitments to the secret and its shares are broadcast, and the shares themselves are appropriately appended to the broadcast commitments so that parties receive their shares while maintaining their secrecy. To overcome an adversarial dealer that does not provide some honest party with its correct share, parties send sub-shares to each other along with the broadcasted commitment. The victim party then computes its share from the received sub-shares. AVSS implements this using bivariate polynomial-based secret sharing, which leads to a commitment (or broadcast) of size $\Theta(\kappa n^2)$ and correspondingly $O(\kappa n^4)$ bits of communication. In the same paper, Cachin et al. improve their AVSS scheme by reducing the commitment-size to $\Theta(\kappa n)$, which results in $O(\kappa n^3)$ bits of communication. A linear gap between the message complexity and the communication complexity still remains.

A Mismatch between AVSS and PolyCommit*.* It is tempting to consider filling this gap for AVSS using a bivariate PolyCommit scheme that commits to an entire bivariate polynomial using a constant-size commitment; however, this does not seem to be possible with the existing PolyCommit methodology. PolyCommit schemes use the algebraic property that, for $\phi(x) \in \mathbb{F}_p[x]$, $(x - i)$ perfectly divides the polynomial $\phi(x) - \phi(i)$ for any $i \in \mathbb{F}_p$. However, such a perfect and direct relation is not known between a bivariate polynomial $\phi(x, y)$ and its evaluations $\phi(i, j)$ for any $i, j \in \mathbb{F}_p$.[1] Therefore, we will have to use two-stage properties involving univariate polynomials (e.g., $(x-i)(y-j)$ perfectly divides the polynomial $\phi(x, y)-\phi(i, y)-\phi(x, j)+\phi(i, j)$ for any $i, j \in \mathbb{F}_p$). However, this does not work either because even though the t-SDH problem to find $\langle c, g^{\frac{1}{\alpha+c}} \rangle$ for any value of $c \in \mathbb{Z}_p$ given $\langle g, g^{\alpha}, g^{\alpha^2}, \dots, g^{\alpha^t} \rangle$ is conjectured to be hard, its exponential version to find a pair $\langle g^c, g^{\frac{1}{\alpha+c}} \rangle$ is easy.

[1] This is equivalent to derivatives in calculus, where complete derivation of a multi-variable equation is not possible and partial derivatives are employed.

A closer look at AVSS reveals that further reducing the commitment-size in the hash-based approach of Cachin et al. using a univariate PolyCommit scheme also does not work: Cachin et al. hash the shares (or the univariate polynomials) for n parties and the secret. These $n+1$ hashed values sent to each party constitute a polynomial of degree n instead of degree t of the underlying bivariate polynomial. This requires an honest party to wait for (constant-size) messages from all n parties in AVSS, which is impossible in the asynchronous setting.

As a result, we have to work towards our goal of asynchronous VSS with $O(\kappa n^2)$ in a different way. In the next section, we provide an asynchronous VSS that satisfies the basic VSS definition, and extend it to a stronger version with applicability in all known VSS applications in Section 4.

3 eAVSS: Asynchronous VSS Protocol

In this section, we present a protocol (eAVSS) with $O(n^2)$ message complexity and $O(\kappa n^2)$ communication complexity and that satisfies Definition 5 of asynchronous VSS. The eAVSS protocol guarantees that at least $t+1$ honest parties receive proper shares of the secret committed using a t-degree univariate polynomial during the Sh phase, while the remaining honest parties are assured that there are at least $t+1$ honest parties that have received correct shares and can complete the Rec phase. The protocol is sufficient for applications such as Byzantine agreement and stand-alone VSS. The protocol construction is significantly simpler than the AVSS protocol [9] and it has a protocol flow similar to a VSS protocol for non-homomorphic commitments [4].

3.1 Construction

We assume a $\mathsf{PolyCommit}_{\mathsf{Ped}}$ commitment Setup instance $\mathsf{SP} \leftarrow \mathsf{Setup}(1^\kappa, t)$. We choose $\mathsf{PolyCommit}_{\mathsf{Ped}}$ due to its unconditional hiding property and the constant size of the commitments. It can, however, be replaced by any polynomial commitment scheme.

The dealer P_d starts off the protocol by choosing a *univariate* polynomial $\phi(x)$ with $\phi(0) = s$, and computing a commitment $\langle \mathcal{C}, d\rangle \leftarrow \mathsf{Commit}(\mathsf{SP}, \phi(x))$ and corresponding witnesses $w_j \leftarrow \mathsf{CreateWitness}(\mathsf{SP}, \phi(x), d, j)$ for $j \in [1, n]$. In $\mathsf{PolyCommit}_{\mathsf{Ped}}$, the decommitment information d is a t-degree polynomial, which is represented as $\widehat{\phi}(x)$ in the rest of the paper. P_d then sends $(\mathsf{send}, \mathcal{C}, \phi(j), \widehat{\phi}(j), w_j)$ messages to all parties and the parties verify their shares against the received commitment $\mathcal{C}$. In the rest of the protocol, the parties try to agree on $\mathcal{C}$. Unlike AVSS, the parties in eAVSS do not exchange their common evaluations of a bivariate polynomial; they only verify consistency of the received shares (if any) with $\mathcal{C}$ locally. If the dealer is dishonest, some honest parties may not receive shares consistent with $\mathcal{C}$; however, they still help to reach an agreement on $\mathcal{C}$ once they are assured that at least $t+1$ honest parties have received shares and witnesses consistent with $\mathcal{C}$. We describe the protocol in Figure 1. Note that commitment $\mathcal{C}$ is set to $\perp$ initially. An honest party accepts only one message

Sh Phase

Dealer P_d with the secret s

- Select a polynomial $\phi(x)$ of degree t such that $\phi(0) = s$.
- Compute a commitment $\langle \mathcal{C}, \widehat{\phi}(x)\rangle \leftarrow \mathsf{Commit}(\mathsf{SP}, \phi(x))$ and witnesses $w_j \leftarrow$ $\mathsf{CreateWitness}$ $(\mathsf{SP}, \phi(x), \widehat{\phi}(x), j)$ for $j \in [1, n]$.
- Send $(\mathsf{send}, \mathcal{C}' = \mathcal{C}, \phi(j), \widehat{\phi}(j), w_j)$ to every party P_j.

Every party P_i

- On receiving a message $(\mathsf{send}, \mathcal{C}', \phi(i), \widehat{\phi}(i), w_i)$ from P_d, run $\mathsf{VerifyEval}(\mathsf{SP}, \mathcal{C}',$ i, $\phi(i)$, $\widehat{\phi}(i)$, $w_i)$. If the verification succeeds, set $\mathcal{C} = \mathcal{C}'$ and send $(\mathsf{echo}, \mathcal{C}')$ to all parties.
- On receiving $(\mathsf{echo}, \mathcal{C}')$ messages from $(n - t)$ parties:
  - For $\mathcal{C}' = \mathcal{C}$, send $(\mathsf{ready}, \mathsf{share}, \mathcal{C}')$ to all parties;
  - For $\mathcal{C}' \neq \mathcal{C}$, discard $\langle \phi(j), \widehat{\phi}(j), w_j\rangle$, set $\mathcal{C} = \mathcal{C}'$, and send $(\mathsf{ready}, \mathsf{no\text{-}share}, \mathcal{C}')$ to all parties.
- If a ready message has not been sent, then on receiving $(\mathsf{ready}, *, \mathcal{C}')$ messages from $t + 1$ parties:
  - For $\mathcal{C}' = \mathcal{C}$, send $(\mathsf{ready}, \mathsf{share}, \mathcal{C}')$ to all parties;
  - For $\mathcal{C}' \neq \mathcal{C}$, discard $\langle \phi(j), \widehat{\phi}(j), w_j\rangle$, set $\mathcal{C} = \mathcal{C}'$, and send $(\mathsf{ready}, \mathsf{no\text{-}share}, \mathcal{C}')$ to all parties.
- On receiving $(\mathsf{ready}, *, \mathcal{C}')$ messages from at least $(n-t)$ parties such that share flags are set in at least $t+1$ of those, complete the Sh phase with commitment $\mathcal{C} = \mathcal{C}'$ (and $\langle \phi(i), \widehat{\phi}(i), w_i\rangle$ if present).

Rec Phase

Every party P_i

- Send $(\mathsf{rec\text{-}share}, \phi(i), \widehat{\phi}(i), w_i)$ to all parties, if it has sent a $(\mathsf{ready}, \mathsf{share}, \mathcal{C})$ message in the Sh phase.
- On receiving $t + 1$ $\mathsf{rec\text{-}share}$ messages verified using $\mathsf{VerifyEval}(\mathsf{SP}, \mathcal{C}, j, \phi(j),$ $\widehat{\phi}(j), w_j)$, interpolate the secret as $s = \phi(0)$.

Fig. 1. Protocol eAVSS for Asynchronous VSS ($n \geq 3t + 1$)

of a kind from any other party, and without loss of generality, we assume that every party chooses only the first message.

The protocol requires $O(n^2)$ messages as decided by its echo and ready messages. Use of PolyCommit ensures that all messages are of a constant size, and results in $O(\kappa n^2)$ communication complexity.

3.2 Analysis

Theorem 1. *Given a PolyCommit scheme that satisfies Definition [4], eAVSS is an asynchronous VSS protocol that satisfies Definition [5].*

Proof. To prove the theorem, we show that protocol eAVSS satisfies liveness, agreement, correctness, commitment, secrecy properties of asynchronous VSS according to Definition 5. Our analysis is based on the properties of the polynomial commitment scheme used.

We start by proving the following two claims.

Claim. If some honest party has agrees on $\mathcal{C}$, then every honest party will eventually agree on $\mathcal{C}$.

Proof. We first prove by contradiction that if P_i be the first honest party to send ready message containing $\mathcal{C}$, then a ready message sent by every other honest party P_j will contain $\mathcal{C}$. Assume an honest party P_j sends a ready message with $\overline{\mathcal{C}}$ such that $\mathcal{C} \neq \overline{\mathcal{C}}$. Being first honest party to send a ready message with $\mathcal{C}$ party P_i must have received $(\mathsf{echo}, \mathcal{C})$ from at least $n-t$ parties of which at least $n-2t$ were honest. P_j can send $\overline{\mathcal{C}}$ only after one of the following two events and in both cases we arrive at a contradiction:

1. P_j can send $(\mathsf{ready}, \overline{\mathcal{C}})$ after receiving $(\mathsf{echo}, \overline{\mathcal{C}})$ from at least $n-t$ parties. As $n \geq 3t+1$, $(n-t)+(n-t)-n = n-2t \geq t+1$ parties must have sent echo with both $\mathcal{C}$ and $\overline{\mathcal{C}}$. This implies that at least one *honest* party sent echo messages of two types, which is impossible.
2. P_j can also send $(\mathsf{ready}, \overline{\mathcal{C}})$ after receiving $n-2t$ $(\mathsf{ready}, *, \overline{\mathcal{C}})$ messages. For $n \geq 3t+1$, $n-2t \geq t+1$. Therefore, there is at least one honest party (say P_k), who sent $\overline{\mathcal{C}}$ in its ready message to P_j. This means that one of the events (1) or (2) must have occurred with the honest party P_k. If we argue in a recursive manner, we reach some honest party who must have experienced event (1), which is a contradiction.

Therefore, no two honest parties will send ready messages containing different commitments.

A honest party agrees on $\mathcal{C}$ only after receiving at least $n-t$ ready messages such that at least $t+1$ contain share. Therefore, $n-2t \geq t+1$ honest parties must have sent ready message and at least one honest party must have sent a ready message containing share. ready messages from $t+1$ or more parties will eventually reach all remaining honest parties and they will send ready messages with the same $\mathcal{C}$, as discussed above. As the number of honest parties is at least $n-t$, every honest party will receive at least $n-t$ ready messages.

It, however, remains to be shown that every honest party will eventually receive at least $t+1$ ready messages with the share flag. From the above paragraph, we know that at least one honest party must have sent a ready message for $\mathcal{C}$ after receiving $n-t$ echo messages for $\mathcal{C}$ and, out of those, at least $n-2t \geq t+1$ are sent by honest parties. As an honest party sends an echo message only after receiving a verified send message from the dealer, at least $t+1$ honest parties must have received their shares from the dealer. As every honest party eventually sends a ready message, these $t+1$ parties will also certainly send ready messages and importantly, they will contain the share flag. Therefore, every honest party will eventually receive $n-t$ ready messages for $\mathcal{C}$ and at least $t+1$ among them will have share flags and thereby agree on $\mathcal{C}$.

Claim. If some honest party agrees on $\mathcal{C}$, then there exists a subset of at least $n-2t \geq t+1$ honest parties such that each of those holds an evaluation of a degree-t polynomial consistent with $\mathcal{C}$.

Proof. From the proof of Claim 3.2, $n-2t$ honest parties will eventually send out ready messages for $\mathcal{C}$ with share; these $n-2t$ honest parties have received verifiable send messages for $\mathcal{C}$ from the dealer P_d. Note that these honest parties never update $\mathcal{C}$, and eventually agree on the same $\mathcal{C}$ by Claim 3.2. Due to the strong correctness and polynomial binding properties of $\mathsf{PolyCommit}_{\mathsf{Ped}}$, there is a unique t-degree of polynomial $\phi(x)$ committed by $\mathcal{C}$. Therefore, evaluations available with these $n-2t \geq t+1$ parties implicitly defines $\phi(x)$ that is consistent with $\mathcal{C}$.

Liveness. If the dealer P_d is honest, then every honest party will eventually receive verifiable send message sent by P_d and will send an echo message and then a ready message. As there are $n-t \geq 2t+1$ honest parties, they will finally agree on $\mathcal{C}$ and complete the Sh phase.

Agreement. A party completes its Sh phase as soon as it agrees on a commitment $\mathcal{C}$. Claim 3.2 suggests that if an honest party agrees on $\mathcal{C}$, then every honest party will eventually agree on $\mathcal{C}$. Therefore, if one honest party completes the Sh phase, then every honest party will complete its Sh phase.

For agreement in the Rec phase, Claim 3.2 shows that there is a subset of at least $t+1$ honest parties each holding an evaluation of a degree-t polynomial $\phi(x)$ that is consistent with $\mathcal{C}$. As every honest party participates in the Rec phase, $t+1$ correct evaluations of $\phi(x)$ associated with $\mathcal{C}$ are available in the Rec phase, and the secret $s = \phi(0)$ can be interpolated by every honest party.

Correctness. Assume that the dealer has shared a secret s using a polynomial $\phi(x)$, and has remained honest throughout the execution of the Sh phase. Let $\mathcal{C}$ be the commitment to $\phi(x)$ sent by the dealer. Given correctness of the polynomial commitment scheme, all honest parties will receive correct shares of the secret s that is consistent with $\mathcal{C}$. Therefore, as we discussed above for agreement, the same secret s will be reconstructed by the parties.

Commitment. We prove the commitment by contradiction. Assume that two different honest parties P_i and P_j reconstruct different s' and s'' such that $s' \neq s''$, The maximum possible degree of the committed polynomial is t due to strong correctness of $\mathsf{PolyCommit}_{\mathsf{Ped}}$. Therefore, each of them must have agreed upon different commitments (say) $\mathcal{C}'$ and $\mathcal{C}''$ in the Sh phase. However, this contradicts with Claim 3.2. Therefore, a unique value $s^* \in \mathbb{F}_p$ will be reconstructed by all honest parties.

Secrecy. To prove secrecy, we need to show that if dealer P_d is honest, then the adversary $\mathcal{A}$ gains no information about the secret s. A t-limited adversary will be able to obtain t messages of the form $(\mathsf{send}, \mathcal{C}, w_i, \phi(i))$. Due to the hiding property for polynomial commitments, given only t such messages it is impossible to reconstruct polynomial $\phi(x)$ (of degree t) and correspondingly the dealer's secret $s = \phi(0)$.

4 eAVSS-SC: AVSS Protocol with Strong Commitment

Although protocol eAVSS in Section 3 does not attain the strong commitment property, it can be used as a component of a VSS protocol that satisfies it. The most intuitive way to realize such a VSS scheme is to make P_d execute $(n+1)$ correlated instances of eAVSS, where the secret s is shared using the first instance (say) eAVSS_0 and the associated shares or polynomial evaluations for all n parties in eAVSS_0 are themselves shared using n instances eAVSS_j for $j \in [1, n]$. Once all eAVSS_j instances complete their Sh phases, a subset of $t+1$ or more honest parties provide every P_j its share in eAVSS_0 by running the Rec phase of eAVSS_j, and by sending their verifiable shares of eAVSS_j to only P_j. It is possible to combine send, echo and ready messages for all $n+1$ instances to keep the message complexity the same as that of AVSS and eAVSS, i.e., $O(n^2)$. However, to broadcast all associated commitments, the communication complexity becomes $O(\kappa n^3)$, which is no better than that of AVSS [9]. In protocol eAVSS-SC, we overcome this drawback using a collision-resistant hash function.

4.1 Construction

Here, the dealer P_d shares the secret s using a symmetric bivariate polynomial $\phi(x, y)$ such that $\phi(0, 0) = s$. The dealer commits to this bivariate polynomial using the univariate PolyCommit scheme twice. In Section 2.4, we observed that constant-size commitments to bivariate polynomials seem difficult, if not impossible. Here, we overcome this hurdle using PolyCommit over the hashed univariate PolyCommit values.[2] We provide an expository description of protocol eAVSS-SC in Figure 2. Notice that although we use send, echo, and ready messages similar to AVSS, our message structures and their utilities are significantly different from those of AVSS. These message structures are crucial to adopt a univariate $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme to our asynchronous VSS scheme, which uses bivariate polynomials.

The protocol requires two $\mathsf{PolyCommit}_{\mathsf{Ped}}$ instances: $\mathsf{SP}_1 \leftarrow \mathsf{Setup}(1^\kappa, t)$ and $\mathsf{SP}_2 \leftarrow \mathsf{Setup}(1^\kappa, n)$. P_d runs $n+1$ eAVSS instances with polynomials $\phi(x, 0)$, $\phi(x, 1)$, ..., $\phi(x, n)$. Let $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n$ be the commitments for these $n+1$ instances. P_d also computes an n-degree polynomial $h_{\mathcal{C}}(x)$ from $H(\mathcal{C}_0), H(\mathcal{C}_1), \ldots, H(\mathcal{C}_n)$, where $H : \mathbb{G} \rightarrow \mathcal{F}_p$ is a collision-resistant hash function and broadcasts a commitment ζ to $h_{\mathcal{C}}(x)$. The dealer cannot cheat with $\phi(x, y)$ as the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme is binding and the hash function is collision-resistant. When all honest parties agree on ζ, they implicitly agree on $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n$. As $t+1$ or more honest parties have received all required shares $\phi_i(x) = \phi(x, i)$ and $\hat{\phi}_i(x) = \hat{\phi}(x, i)$, and commitments $\boldsymbol{\mathcal{C}}$, they can provide all parties their required shares, commitments and witnesses in a verifiable manner using the homomorphic property of $\mathsf{PolyCommit}_{\mathsf{Ped}}$. We optimize this final step by attaching the required shares, witnesses and commitments to the ready messages.

[2] Note that our scheme is not a generic constant-size commitment scheme for bivariate polynomials and some care has to be taken before applying it in other applications; e.g., our scheme cannot be applied to the main as well as the refined AVSS protocols [9] without making their computational complexity exponential $O\binom{n}{t}$.

Sh Phase

Dealer P_d with the secret s

- Choose a symmetric t-degree bivariate polynomial $\phi(x,y)$ such that $\phi(0,0) = s$ and $\phi(i,j) = \phi(j,i)$.
- Commit to $\phi(x,y)$ using a vector $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_j\}_{j\in[0,n]}$, where $\langle \mathcal{C}_j, \hat{\phi}_j(x)\rangle \leftarrow \mathsf{Commit}(\mathsf{SP}_1, \phi_j(x))$, $\phi_j(x) = \phi(x,j)$ and $\hat{\phi}(x,y)$ is symmetric. Also, compute witness vectors $\overrightarrow{\mathcal{W}_j} = \{w_j^k\}_{k\in[0,n]}$ for every party P_j such that $w_j^k \leftarrow \mathsf{CreateWitness}$ $(\mathsf{SP}_1, \phi_k(x), \hat{\phi}_k(x), j)$.
- Compute an n-degree polynomial $h_{\mathcal{C}}(x)$ from $H(\mathcal{C}_0), H(\mathcal{C}_1), \ldots, H(\mathcal{C}_n)$, where $H : G \to \mathcal{F}_p$ is a collision-resistant hash function and commit to it $\langle \zeta, \widehat{h}_{\mathcal{C}}(x)\rangle \leftarrow \mathsf{Commit}(\mathsf{SP}_2, h_{\mathcal{C}}(x))$
- Send (send, $\zeta' = \zeta$, $\boldsymbol{\mathcal{C}'} = \boldsymbol{\mathcal{C}}$, $\widehat{h}_{\mathcal{C}}(x)$, $\overrightarrow{\mathcal{W}_j}$, $\phi_j(x)$, $\hat{\phi}_j(x)$) to every party P_j.

Every party P_i

- On receiving (send, ζ', $\boldsymbol{\mathcal{C}'}$, $\widehat{h}_{\mathcal{C}}(x)$, $\overrightarrow{\mathcal{W}_i}$, $\phi_i(x)$, $\hat{\phi}_i(x)$) from P_d, verify its correctness:
  - interpolate the complete $\boldsymbol{\mathcal{C}'}$ from any of its $t+1$ elements to assert the degree t of the polynomial;
  - compute $h_{\mathcal{C}}(x)$ from $\boldsymbol{\mathcal{C}'}$ and $\mathsf{VerifyPoly}(\mathsf{SP}_2, \zeta', h_{\mathcal{C}}(x), \widehat{h}_{\mathcal{C}}(x))$;
  - $\mathsf{VerifyPoly}(\mathsf{SP}_1, \mathcal{C}'_i, \phi_i(x), \hat{\phi}_i(x))$;
  - $\mathsf{VerifyEval}(\mathsf{SP}_1, \mathcal{C}'_j, i, \phi_j(i)\big[=\phi_i(j)\big], \hat{\phi}_j(i)\big[=\hat{\phi}_i(j)\big], w_i^j)$ for every $j \in [0,n]$.

  Upon a successful verification, set $\zeta = \zeta'$ and $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{C}'}$, compute witnesses $w_j^i \leftarrow \mathsf{CreateWitness}$ $(\mathsf{SP}_1, \phi_i(x), \hat{\phi}_i(x), j)$ for $j \in [1,n]$ and $w_i^{\mathcal{C}} \leftarrow \mathsf{CreateWitness}(\mathsf{SP}_2, h_{\mathcal{C}}(x), \widehat{h}_{\mathcal{C}}(x), i)$. Send a message ($\mathsf{echo}$, ζ') to all parties.
- On receiving (echo, ζ') from at least $(n-t)$ parties:
  - If $\zeta' = \zeta$, send (ready, $\zeta', \mathsf{share}, \phi_i(j), \hat{\phi}_i(j), w_j^i, \mathcal{C}_i, \widehat{h}_{\mathcal{C}}(i), w_i^{\mathcal{C}}$) to every party P_j;
  - If $\zeta' \neq \zeta$, discard $\langle \boldsymbol{\mathcal{C}}, \overrightarrow{\mathcal{W}_i}, \phi_i(x), \hat{\phi}_i(x)\rangle$, set $\zeta = \zeta'$, and send (ready, ζ', $\mathsf{no\text{-}share}$) to all parties.
- If a ready message has not been sent, then on receiving (ready, ζ', $*$) messages from $(t+1)$ parties:
  - If $\zeta' = \zeta$, send (ready, $\zeta', \mathsf{share}, \phi_i(j), \hat{\phi}_i(j), w_j^i, \mathcal{C}_i, \widehat{h}_{\mathcal{C}}(i), w_i^{\mathcal{C}}$) to every party P_j;
  - If $\zeta' \neq \zeta$, discard $\langle \boldsymbol{\mathcal{C}}, \overrightarrow{\mathcal{W}_i}, \phi_i(x), \hat{\phi}_i(x)\rangle$, set $\zeta = \zeta'$, and send (ready, ζ', $\mathsf{no\text{-}share}$) to all parties.
- On receiving (ready, ζ', $*$) messages from at least $(n-t)$ parties such that at least $(t+1)$ of those messages contain $\langle \mathsf{share}, \phi_j(i), \hat{\phi}_j(i), w_i^j, \mathcal{C}_j, \hat{h}_{\mathcal{C}}(j), w_{\mathcal{C}}^j\rangle$ successfully verified using $\mathsf{VerifyEval}(\mathsf{SP}_1, \mathcal{C}_j, i, \phi_j(i), \hat{\phi}_j(i), w_i^j)$ and $\mathsf{VerifyEval}(\mathsf{SP}_2, \zeta', j, H(\mathcal{C}_j), \hat{h}_{\mathcal{C}}(j), w_{\mathcal{C}}^j)$, interpolate
  - shares $\phi_0(i)$ and $\hat{\phi}_0(i)$ from respectively $(t+1)$ $\phi_j(i)$ and $(t+1)$ $\hat{\phi}_j(i)$ values,
  - commitment $\mathcal{C}_0$, witness w_i^0 from respectively $(t+1)$ $\mathcal{C}_j$ and $(t+1)$ w_i^j values.

  Complete the Sh phase with $(\zeta = \zeta', \mathcal{C}_0, \phi_0(i), \hat{\phi}_0(i), w_i^0)$ as output.

Rec Phase

Every party P_i

- Send a message ($\mathsf{rec\text{-}share}$, $\phi_0(i)$, $\hat{\phi}_0(i)$, w_i^0) to every party P_j.
- On receiving $t+1$ $\mathsf{rec\text{-}share}$ messages that have been verified using $\mathsf{VerifyEval}(\mathsf{SP}_1, \mathcal{C}_0, \phi_0(j), \hat{\phi}_0(j), w_j^0)$, interpolate shares $\phi_0(j)$ to obtain secret s.

Fig. 2. Protocol eAVSS-SC for Asynchronous VSS with Stronger Commitment

From the protocol description, it is apparent that the message complexity is $O(n^2)$. As we use the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme that commits to univariate polynomials using a single element, the communication complexity is $O(\kappa n^2)$. Note that although the size of send messages is $O(\kappa n)$, only n such messages are delivered; thus, the communication complexity does not exceed $O(\kappa n^2)$.

For simplicity of the description, we define our protocol with a *symmetric* bivariate polynomial. It is easily possible to avoid this symmetry requirement in the protocol without any asymptotic increase in the complexity measures.

4.2 Analysis

Theorem 2. *Given a PolyCommit scheme that satisfies Definition 4, eAVSS-SC is an asynchronous VSS protocol that satisfies Definition 5 with the strong commitment property.*

Proof (Proof Outline). We have to prove that protocol eAVSS satisfies the asynchronous VSS properties in Definition 5 along with the strong commitment property. Our analysis is based on the following two claims and the properties of the PolyCommit scheme. We present our proof sketch here, while the complete proof appears in [3].

Claim. If some honest party agrees on ζ, then every honest party will eventually agree on ζ.

Claim. All honest servers complete the Sh phase with the same PolyCommit commitment $\mathcal{C}_0$.

Proof. Assume two honest parties terminate with $\mathcal{C}_0{}'$ and $\mathcal{C}_0{}''$ such that $\mathcal{C}_0{}' \neq \mathcal{C}_0{}''$. From Claim 4.2, we know that all honest parties agree on the same ζ. As ζ commits to $h_{\mathcal{C}}(x)$, an n-degree polynomial interpolated by hashing $n+1$ elements of $\boldsymbol{\mathcal{C}}$, the adversary has to break the evaluation binding property of the polynomial commitment or the collision resistance property of hash function to obtain two different $\mathcal{C}_0$ values that culminate the same ζ. This is not possible in PPT and there is a contradiction. Therefore, we prove that all honest servers complete the Sh phase with the same $\mathcal{C}_0$.

Liveness follows from the protocol flow and correctness of the PolyCommit scheme. Agreement in the Sh phase is evident from claims 4.2 and 3.2, while agreement in reconstruction follows from agreement during the Sh phase. Correctness follows directly from correctness of the PolyCommit scheme and collision-resistance of the hash function. Strong Commitment is apparent from agreement of eAVSS-SC and Claim 3.2. Secrecy follows from the hiding property of PolyCommit.

4.3 Lower Bounds

We observe that the $\Omega(n^2)$ message complexity of our eAVSS and eAVSS-SC protocols as well as the AVSS protocol is optimal.[3] This can be proved in two

[3] With the stronger cryptographic assumptions such as PKI or ZK proofs more efficient schemes can be possible; however, we only assume commitment schemes here.

steps: first, it is known that a VSS protocol is sufficient to implement reliable broadcast [19]; next, extending a result by Dolev and Reischuk [14] for Byzantine agreement to reliable broadcast. The latter proves that if a reliable broadcast protocol terminates, the number of messages exchanged by honest parties is lower bounded by $max\{(n-t), (1+t/2)^2\}$ in presence of a commitment scheme. The above two claims show that the message complexity of asynchronous VSS is lower-bounded by $\Omega(n^2)$ for optimal resiliency condition $n = 3t+1$ and $t > 2$. We thoroughly prove this result in the extended version of this paper [3].

Note that when the shared secret is of size κ (the computational security parameter), the lower bound of $\Omega(n^2)$ message complexity *intuitively* transfers to a lower bound of $\Omega(\kappa n^2)$ on the asynchronous VSS communication complexity. Nevertheless, proving this thoroughly presents an interesting challenge. If proven, it will show that our eAVSS and eAVSS-SC protocols are not only optimal in terms of message complexity but also in terms of communication complexity.

5 Applications

Our eAVSS and eAVSS-SC schemes have direct implications to all asynchronous VSS applications. We briefly discuss some important applications here.

Using our eAVSS-SC protocol in proactive VSS [9] reduces its communication complexity by a linear factor to $O(\kappa n^3)$. The same reduction also applies to distributed key generation required for threshold cryptography, and its group and threshold modification primitives [17]. Using our eAVSS protocol in the asynchronous Byzantine agreement (ABA) framework of Canetti and Rabin [10, 11], it is possible to obtain the first $O(\kappa n^3)$ communication complexity ABA protocol, which is secure against the *adaptive* adversary in the *standard* model without the random oracle assumption (see [9, Sec. 3.5] for details).

Finally, our commitment methodology may also find applications in some other bivariate polynomial-based protocols; however, one has to be careful as it is not a full-fledged bivariate polynomial commitment scheme.

References

1. Abraham, I., Dolev, D., Halpern, J.Y.: An Almost-surely Terminating Polynomial Protocol for Asynchronous Byzantine Agreement with Optimal Resilience. In: Proceedings of ACM PODC 2008, pp. 405–414 (2008)
2. Au, M.H., Susilo, W., Mu, Y.: Practical Anonymous Divisible E-Cash from Bounded Accumulators. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 287–301. Springer, Heidelberg (2008)
3. Backes, M., Datta, A., Kate, A.: Asynchronous Computational VSS with Reduced Communication Complexity. Cryptology ePrint Archive, Report 2012/619 (2012), http://eprint.iacr.org/2012/619
4. Backes, M., Kate, A., Patra, A.: Computational Verifiable Secret Sharing Revisited. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 590–609. Springer, Heidelberg (2011)

5. Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous Secure Computation. In: Proceedings of ACM STOC 1993, pp. 52–61 (1993)
6. Blakley, G.R.: Safeguarding Cryptographic Keys. In: Proceedings of the National Computer Conference, pp. 313–317 (1979)
7. 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)
8. Bracha, G.: An Asynchronous [(n-1)/3]-Resilient Consensus Protocol. In: Proceedings of PODC 1984, pp. 154–162 (1984)
9. Cachin, C., Kursawe, K., Lysyanskaya, A., Strobl, R.: Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems. In: Proceedings of ACM CCS 2002, pp. 88–97 (2002)
10. Canetti, R.: Studies in Secure Multiparty Computation and Applications. Ph.D. thesis, The Weizmann Institute of Science (1996)
11. Canetti, R., Rabin, T.: Fast Asynchronous Byzantine Agreement with Optimal Resilience. In: Proceedings of ACM STOC 1993, pp. 42–51 (1993)
12. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In: Proceedings of IEEE FOCS 1985, pp. 383–395 (1985)
13. Cramer, R., Damgård, I., Dziembowski, S.: On the Complexity of Verifiable Secret Sharing and Multiparty Computation. In: Proceedings of STOC 2000, pp. 325–334 (2000)
14. Dolev, D., Reischuk, R.: Bounds on Information Exchange for Byzantine Agreement. Journal of ACM 32(1), 191–204 (1985)
15. Feldman, P.: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: Proceedings of IEEE FOCS 1987, pp. 427–437 (1987)
16. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive Secret Sharing or: How to Cope with Perpetual Leakage. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995)
17. Kate, A., Goldberg, I.: Distributed Key Generation for the Internet. In: Proceedings of 29th IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 119–128 (2009)
18. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-Size Commitments to Polynomials and Their Applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (2010)
19. Katz, J., Koo, C.-Y., Kumaresan, R.: Improving the Round Complexity of VSS in Point-to-Point Networks. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 499–510. Springer, Heidelberg (2008)
20. Patra, A., Choudhary, A., Rangan, C.P.: Efficient Asynchronous Byzantine Agreement with Optimal Resilience. In: Proceedings of ACM PODC 2009, pp. 92–101 (2009)
21. Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
22. Pedersen, T.P.: A Threshold Cryptosystem without a Trusted Party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
23. Schultz, D.A., Liskov, B., Liskov, M.: MPSS: Mobile Proactive Secret Sharing. ACM Trans. Inf. Syst. Secur. 13(4), 34 (2010)
24. Shamir, A.: How to Share a Secret. Commun. ACM 22(11), 612–613 (1979)
25. Zhou, L., Schneider, F.B., van Renesse, R.: APSS: Proactive Secret Sharing in Asynchronous Systems. ACM Trans. Inf. Syst. Secur (TISSec) 8(3), 259–286 (2005)

A Protocol $\mathsf{PolyCommit}_{\mathsf{Ped}}$

In this section, we instantiate the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme that commits to a univariate polynomial using a single group element. $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is based on the algebraic property of polynomials $\phi(x) \in \mathbb{F}_p[x]$: $(x - i)$ perfectly divides the polynomial $\phi(x) - \phi(i)$ for $i \in \mathbb{F}_p$. Further, it uses an additional random polynomial $\hat{\phi}(x)$ to achieve unconditional hiding.

Setup$(1^\kappa, t)$ computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p (providing κ-bit security) such that there exists a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and for which the t-SDH assumption holds. We denote the generated bilinear pairing group as $\mathcal{G} = \langle e, \mathbb{G}, \mathbb{G}_t \rangle$. Choose two generators $g, h \in_R \mathbb{G}$. Let $\alpha \in_R \mathbb{F}_p^*$ be SK, generated by a (possibly distributed) trusted authority. Setup also generates a $(2t+2)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t}, h, h^\alpha, \ldots, h^{\alpha^t} \rangle \in \mathbb{G}^{2t+2}$ and outputs $\mathsf{SP} = \langle \mathcal{G}, g, g^\alpha, \ldots, g^{\alpha^t}, h, h^\alpha, \ldots, h^{\alpha^t} \rangle$. Note that SK is not required by the other algorithms of the commitment scheme, and it can be discarded by the authority if t is fixed.

Commit$(\mathsf{SP}, \phi(x))$ chooses $\hat{\phi}(x) \in_R \mathbb{F}_p[x]$ of degree t and computes the commitment $\mathcal{C} = g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)} \in \mathbb{G}$ for the polynomial $\phi(x) \in \mathbb{F}_p[X]$ of degree t or less. For $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$ and $\hat{\phi}(x) = \sum_{j=0}^{\deg(\hat{\phi})} \hat{\phi}_j x^j$, it outputs $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j} \prod_{j=0}^{\deg(\hat{\phi})} (h^{\alpha^j})^{\hat{\phi}_j}$ as the commitment to $\phi(x)$.

Open$(\mathsf{SP}, \mathcal{C}, \phi(x), \hat{\phi}(x))$ outputs the committed polynomials $\phi(x)$ and $\hat{\phi}(x)$.

VerifyPoly$(\mathsf{SP}, \mathcal{C}, \phi(x), \hat{\phi}(x))$ verifies that $\mathcal{C} \overset{?}{=} g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)}$.
If $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j} \prod_{j=0}^{\deg(\hat{\phi})} (h^{\alpha^j})^{\hat{\phi}_j}$ for $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$ and $\hat{\phi}(x) = \sum_{j=0}^{\deg(\hat{\phi})} \hat{\phi}_j x^j$, the algorithm outputs 1, else it outputs 0. Note that this only works when both $\deg(\phi)$ and $\deg(\hat{\phi}) \le t$.

CreateWitness$(\mathsf{SP}, \phi(x), \hat{\phi}(x), i)$ computes $\psi_i(x) = \frac{\phi(x) - \phi(i)}{(x-i)}$, $\hat{\psi}_i(x) = \frac{\hat{\phi}(x) - \hat{\phi}(i)}{(x-i)}$, and outputs $\langle i, \phi(i), \hat{\phi}(i), w_i \rangle$. Here, the witness $w_i = g^{\psi_i(\alpha)} h^{\hat{\psi}_i(\alpha)}$.

VerifyEval$(\mathsf{SP}, \mathcal{C}, i, \phi(i), \hat{\phi}(i), w_i)$ verifies that $\phi(i)$ is the evaluation at the index i of the polynomial committed to by $\mathcal{C}$. If $e(\mathcal{C}, g) \overset{?}{=} e(w_i, g^\alpha / g^i) e(g^{\phi(i)} h^{\hat{\phi}(i)}, g)$, the algorithm outputs 1, else it outputs 0.

Suppose $h = g^\lambda$ for some unknown λ. Then $\mathsf{VerifyEval}$ is correct because

$$\begin{aligned} e(w_i, g^\alpha / g^i) e(g^{\phi(i)} h^{\hat{\phi}(i)}, g) &= e(g^{\psi_i(\alpha) + \lambda \hat{\psi}_i(\alpha)}, g^{(\alpha - i)}) e(g, g)^{\phi(i) + \lambda \hat{\phi}(i)} \\ &= e(g, g)^{(\psi_i(\alpha)(\alpha - i) + \phi(i)) + \lambda(\hat{\psi}_i(\alpha)(\alpha - i) + \hat{\phi}(i))} \\ &= e(g, g)^{\phi(\alpha) + \lambda \hat{\phi}(\alpha)} = e(g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)}, g) = e(\mathcal{C}, g) \end{aligned}$$

The hiding property of $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is unconditional. The polynomial binding property is based on the DLog assumption, while the evaluation binding property is based on the t-Strong Diffie-Hellman (t-SDH) assumption [7]. The strong correctness property follows from the t-polynomial Diffie-Hellman (t-polyDH) assumption [2,18].

Proxy Re-Encryption in a Stronger Security Model Extended from CT-RSA2012

Toshiyuki Isshiki[1,2], Manh Ha Nguyen[2,⋆], and Keisuke Tanaka[2]

[1] NEC Corporation, Japan
t-issiki@bx.jp.nec.com
[2] Tokyo Institute of Technology, Japan
{nguyen9,keisuke}@is.titech.ac.jp

Abstract. Proxy re-encryption (PRE) realizes delegation of decryption rights, enabling a proxy holding a re-encryption key to convert a ciphertext originally intended for Alice into an encryption of the same message for Bob, and cannot learn anything about the encrypted plaintext. PRE is a very useful primitive, having many applications in distributed file systems, outsourced filtering of encrypted spam, access control over network storage, confidential email, digital right management, and so on. In CT-RSA2012, Hanaoka et al. proposed a chosen-ciphertext (CCA) security definition for PRE, and claimed that it is stronger than all the previous works. Their definition is a somewhat strengthened variant of the replayable-CCA one, however, it does not fully capture the CCA security notion. In this paper, we present a full CCA security definition which is extended from theirs. We then propose the first PRE scheme with this security in the standard model (i.e. without the random oracle idealization). Our scheme is efficient and relies on mild complexity assumptions in bilinear groups.

Keywords: unidirectional proxy re-encryption, chosen-ciphertext attack, pairings.

1 Introduction

Proxy re-encryption (PRE), first introduced by Blaze, Bleumer, and Strauss in [5], allows a proxy to transform ciphertexts computed under the public-key of Alice (the delegator) into other ciphertexts for Bob (the delegatee). The proxy, however, learns nothing about the underlying messages encrypted, and has no knowledge of the secret keys of the delegators and the delegatees.

PRE schemes have applications in digital rights management (DRM) [22], distributed file storage systems [3], law enforcement [15], encrypted email forwarding [5], and outsourced filtering of encrypted spam [3]. In all these cases, the gist is that the process of re-encryption, i.e., decrypting under one key for encryption under another key, should not allow the re-encryptor module to compromise the secrecy of encrypted messages.

⋆ Supported by Ministry of Education, Culture, Sports, Science and Technology.

1.1 Background

According to the direction of transformation, PRE can be classified into two types: *unidirectional* and *bidirectional* [15]. In unidirectional PRE, the proxy can only transform ciphertexts from Alice to Bob. While in bidirectional PRE, the proxy can transform ciphertexts in both directions. PRE can also be categorized into *multi-hop* PRE, in which the ciphertexts can be transformed from Alice to Bob and then to Charlie and so on, and *single-hop* PRE, in which the ciphertexts can only be transformed once [15]. In this work, we only consider unidirectional single-hop PRE schemes.

In 1998, Blaze, Bleumer, and Strauss [5] (whose work is sometimes dubbed BBS) proposed the first bidirectional PRE scheme. It is based on a simple modification of the Elgamal encryption scheme [13]. This scheme is efficient and semantically secure under the decision Diffie-Hellman (DDH) assumption.

In 2005, Ateniese, Fu, Green, and Hohenberger [2,3] showed the first examples of unidirectional PRE schemes based on bilinear maps. Moreover, they obtained the master key security property in that the proxy is unable to collude with the delegatees in order to expose the delegator's secret. The constructions [2,3] are also efficient, semantically secure assuming the intractability of decisional variants of the bilinear Diffie-Hellman problem [7]. These PRE schemes only ensure the chosen-plaintext security, which seems definitely insufficient for many practical applications.

In 2007, Canetti and Hohenberger [10] gave a definition of security against chosen ciphertext attacks (CCA) for PRE schemes and described an efficient construction satisfying this definition (for bidirectional schemes but easily adaptable for the unidirectional case, as explained in [18]). Their security analysis takes place in the standard model (without the random oracle heuristic [4]). Like the BBS scheme [5], their construction is bidirectional and they left as an open problem to come up with a CCA-secure unidirectional scheme.

In 2008, Libert and Vergnaud [18] partially resolved this problem by presenting a single-hop unidirectional PRE scheme without random oracles. They proved that their scheme is secure against the replayable chosen-ciphertext attack (RCCA) [9]. Here the RCCA security is a weaker variant of the CCA security in the sense that a harmless mauling of the challenge ciphertext is tolerated.

Recently, Shao, Liu, and Zhou [21] proposed a construction of PRE and claimed that their scheme achieves key privacy without losing the CCA security in the standard model, which is an open problem left by Ateniese et al. [1]. In this paper, however, we will show that their scheme is vulnerable to chosen-ciphertext attack, and present a concrete attack to break the CCA security of it (see Section 4 for more details).

In CT-RSA2012, Hanaoka, Kawai, Kunihiro, Matsuda, Weng, Zhang, and Zhao [14] proposed a CCA security definition for PRE, and claimed that it is stronger than all the previous works. Their definition is a somewhat strengthened variant of the replayable-CCA one, however, it does not fully capture the CCA security notion.

1.2 Our Contributions

First, we present a CCA security definition for PRE, which naturally extends that of [8] by giving an adversary all the possible resources as the observation in [14]. Then, we point out why the second-level-ciphertext security of the previous works, which include ours, is stronger than that of Hanaoka et al. [14]. Therefore, the security definition proposed in [14] does not fully capture the CCA security notion. Moreover, our security model is the strongest one to date. See **Table 1** for details of the comparison about our security model with related previous works.

As the main goal of this paper, we propose a unidirectional PRE scheme which is secure in our full CCA security model without random oracles. This is also the first CCA-secure unidirectional PRE scheme in the standard model. Our scheme is efficient and relies on a mild complexity assumption in bilinear groups. Additionally, our scheme also achieves the security in the sense of master secret security which is proposed by Ateniese et al. [3].

Table 1: Comparison with the previous unidirectional PRE schemes.
("—" means that the CCA security of the corresponding scheme is broken.)

Authors	Security Model	RO-free	Assumption
Ateniese *et al.* [3]	CPA	$\times$	eDBDH
Libert-Vergnaud [18]	RCCA	○	3-wDBDHI
Canard *et al.* [8]	CCA	$\times$	CDH
Shao *et al.* [21]	—	○	5-EDBDH & DDH
Hanaoka *et al.* [14]	weak CCA	○	DBDH
Ours	CCA	○	6-AmDBDH & 2-AmCDH

1.3 Roadmap

The paper is organized as follows: we review the properties of bilinear maps and the intractability assumptions that our scheme relies on in Section 2. We recall the concept of unidirectional PRE and its security model in Section 3. In Section 4, we present a concrete attack to break the CCA security of Shao-Liu-Zhou's scheme. Section 5 describes the main new scheme, gives the intuition behind its construction and a security proof. We conclude the paper in Section 6.

2 Preliminaries

Notations. We use $x \overset{\mathrm{R}}{\leftarrow} \mathcal{S}$ to denote that an element x is chosen uniformly at random from $\mathcal{S}$.

2.1 Bilinear Maps and Complexity Assumptions

Groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p are called *bilinear map groups* if there is a mapping $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties: (1) bilinearity: $\mathbf{e}(g^a, h^b) = \mathbf{e}(g, h)^{ab}$ for any $(g, h) \in \mathbb{G} \times \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, (2) efficient computability for any input pair, (3) non-degeneracy: $\mathbf{e}(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

In this paper, our proposed scheme shall use a variant of the modified decisional bilinear Diffie-Hellman assumption (a.k.a. mDBDH assumption) [10,20] named 6 augmented modified decisional bilinear Diffie-Hellman assumption. The details of this assumption is as follows.

Definition 1. (6-AmDBDH Problem). *Given a group $\mathbb{G}$ of prime order p with generator g. The 6-augmented modified decisional bilinear Diffie-Hellman (6-*AmDBDH*) problem in groups $(\mathbb{G}, \mathbb{G}_T)$ is, given $(g, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, Q) \in \mathbb{G}^{10} \times \mathbb{G}_T$, where $a, b, c, z \xleftarrow{R} \mathbb{Z}_p$, decide whether $Q = \mathbf{e}(g, g)^{ab/c^2}$. For an adversary $\mathcal{A}$, we define its advantage in solving the* 6-AmDBDH *problem in groups $(\mathbb{G}, \mathbb{G}_T)$ as*

$$\mathrm{Adv}_{\mathcal{A}}^{6\text{-AmDBDH}} \overset{def}{=}$$
$$\Big|\Pr[\mathcal{A}(g, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, \mathbf{e}(g, g)^{ab/c^2}) = 1 | a, b, c \xleftarrow{\mathrm{R}} \mathbb{Z}_p]$$
$$- \Pr[\mathcal{A}(g, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, \mathbf{e}(g, g)^z) = 1 | a, b, c, z \xleftarrow{\mathrm{R}} \mathbb{Z}_p]\Big|.$$

We say that the (t, ϵ)-6-AmDBDH *assumption holds in $(\mathbb{G}, \mathbb{G}_T)$ if no t-time adversary $\mathcal{A}$ has advantage at least ϵ in solving the 6-*AmDBDH *problem in $(\mathbb{G}, \mathbb{G}_T)$.*

The hardness of the 6-AmDBDH problem is implied by that of the general Diffie-Hellman problem in the generic bilinear group model [6]. In particularly, a generic attacker's advantage in solving the 6-AmDBDH problem (on which our CCA-secure PRE scheme is based) is less than 16 times of that in solving the DBDH problem. The details are given in the full version of this paper.

We also use a variant of the computational Diffie-Hellman assumption (a.k.a. CDH assumption) named 2 augmented modified computational Diffie-Hellman assumption, which is almost identical to the CDH assumption except the introduction of the additional terms $g^{ba^2}, g^{\frac{b}{a}}$.

Definition 2 (2-AmCDH). *Given a group $\mathbb{G}$ of prime order p with generator g. The 2-augmented modified computation Diffie-Hellman (2-*AmCDH*) problem is defined as $\mathrm{Adv}_{\mathcal{A}}^{2\text{-AmCDH}} = \Pr[\mathcal{A}(g, g^a, g^b, g^{ba^2}, g^{\frac{b}{a}}) = g^{ab}]$, where the probability is taken over the random choices of a, b and those made by $\mathcal{A}$. We say that the (t, ϵ)-2-AmCDH assumption holds in $\mathbb{G}$ if no t-time algorithm $\mathcal{A}$ has advantage at least ϵ in solving the 2-AmCDH problem in $\mathbb{G}$.*

2.2 The Gap Hashed Diffie-Hellman Assumption

Let Gen be a polynomial-time algorithm that on input 1^k returns the description of a multiplicative cyclic group $\mathbb{G}$ of prime order p, where $2^k < p < 2^{k+1}$, and a random generator g of $\mathbb{G}$. Gen furthermore outputs the description of a random hash function $H : \mathbb{G} \to \{0,1\}^{\ell(k)}$ that outputs $\ell(k)$ bits for a fixed polynomial $\ell(\cdot)$. Throughout the paper we use $\mathcal{HG} = (\mathbb{G}, g, p, H)$ as shorthand for the description of the hash group obtained by running Gen.

The gap hashed Diffie-Hellman (GHDH) assumption [16] states, roughly, that the two distributions $(g^x, g^y, H(g^{xy}))$ and (g^x, g^y, R) are computationally indistinguishable when x, y are drawn at random from $\mathbb{Z}_p$ and R is drawn at random from $\{0,1\}^{\ell(k)}$. This assumption should hold relative to an oracle that efficiently solves the DDH problem (See [16] for more details).

More formally let Gen be a parameter generation algorithm. To an adversary $\mathcal{B}$ we associate the following experiment.

Experiment $\mathbf{Exp}^{\text{ghdh}}_{\mathsf{Gen},\mathcal{B}}(1^k)$,
$\mathcal{HG} = (\mathbb{G}, g, p, H) \leftarrow_R \mathsf{Gen}(1^k)$,
$x, y \leftarrow_R \mathbb{Z}_p^*$, $Q_0 \leftarrow_R \{0,1\}^{\ell(k)}$, $Q_1 \leftarrow_R H(g^{xy})$,
$\gamma \leftarrow_R \{0,1\}$,
$\gamma' \leftarrow_R \mathcal{B}^{\text{DDHsolve}_{\mathbb{G}}(\cdot,\cdot,\cdot,\cdot)}(1^k, \mathcal{HG}, g^x, g^y, Q_\gamma)$,
If $\gamma \neq \gamma'$ then return 0 else return 1.

Here the oracle $\text{DDHsolve}_{\mathbb{G}}(g, g^a, g^b, g^c)$ returns 1 iff $ab = c \mod p$. Note that, since we make use of bilinear map, $\mathbf{e}(\cdot,\cdot)$ can be seen as the oracle $\text{DDHsolve}_{\mathbb{G}}$. We define the advantage of $\mathcal{B}$ in the above experiment as

$$\text{Adv}^{\text{ghdh}}_{\text{Gen},\mathcal{B}}(k) = \left| \Pr\left[\mathbf{Exp}^{\text{ghdh}}_{\mathsf{Gen},\mathcal{B}}(1^k) = 1\right] - \frac{1}{2} \right|.$$

We say that the *GHDH assumption relative to group generator* Gen holds if $\text{Adv}^{\text{ghdh}}_{\text{Gen},\mathcal{B}}(k)$ is a negligible function in k for all polynomial-time adversaries $\mathcal{B}$.

2.3 Target-Collision Resistant Hash Function

Definition 3. (Target-Collision Resistant Hash Function). *Let $H : X \to Y$ be a hash function. For an algorithm $\mathcal{A}$, define its advantage as*

$$\text{Adv}^{\text{TCR}}_{H,\mathcal{A}} = \Pr[x \leftarrow X, x' \leftarrow \mathcal{A}(x) : x' \neq x \wedge H(x') = H(x)].$$

We say that H is target-collision resistant (TCR) if for any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, its advantage $\text{Adv}^{\text{TCR}}_{H,\mathcal{A}}$ is negligible.

2.4 Symmetric Encryption

We review the formal notion of symmetric encryption and its security definition as follows.

Definition 4 (Symmetric Encryption). *Let $\mathcal{K}_D$ be the key space. A symmetric encryption scheme, denoted by* **SYM**, *consists of the following algorithms:*

- **SYM.Enc:** *Taking a key $K \in \mathcal{K}_D$ and a plaintext M as input, this algorithm encrypts M into a ciphertext e. We write $e \leftarrow \mathbf{SYM}.\mathbf{Enc}(K, M)$.*
- **SYM.Dec:** *Taking $K \in \mathcal{K}_D$ and e as input, this algorithm decrypts e into M. Note that M can be $\perp$. We write $M \leftarrow \mathbf{SYM}.\mathbf{Enc}(K, e)$.*

Definition 5 (IND-CCA Security of Symmetric Encryption).
Let $\mathbf{SYM}$ *be a symmetric encryption scheme as defined in Definition 4. Consider a game played with an attacker $\mathcal{A}$:*

Phase 1. *The game chooses $K \overset{\$}{\leftarrow} \mathcal{K}_D$.*

Phase 2. *$\mathcal{A}$ issues encryption queries, each of which is denoted by M. On receiving this, the game computes $e \xleftarrow{\$} \mathbf{Enc}(K, M)$ and gives e to $\mathcal{A}$. $\mathcal{A}$ also issues decryption queries, each of which is denoted by e. On receiving this, the game computes $M \leftarrow \mathbf{Dec}(K, e)$ and gives M to $\mathcal{A}$.*

Challenge. *$\mathcal{A}$ issues a challenge query (a pair of plaintexts) (m_0, m_1) such that $|m_0| = |m_1|$. On receiving this, the game picks $b \xleftarrow{\$} \{0,1\}$, computes $e^* \xleftarrow{\$} \mathbf{SYM}.\mathbf{Enc}(K,$
$m_b)$ and gives e^* to $\mathcal{A}$.*

Phase 3. *$\mathcal{A}$ continues to issue encryption and decryption queries as in Phase 2. However, a restriction here is that $\mathcal{A}$ is not allowed to issue e^* as decryption query. The game responds to $\mathcal{A}$'s queries in the same way as it did in Phase 2.*

Guess. *$\mathcal{A}$ outputs its guess $b' \in \{0,1\}$.*

We define $\mathcal{A}$'s advantage by $\mathrm{Adv}^{\text{IND-CCA}}_{\text{SYM},\mathcal{A}}(n) = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

3 Models and Security Notions

In this section, we first review the concept of unidirectional single-hop PRE. Then, we present a new CCA security definition for PRE, which naturally extends that of [8] by giving an adversary all the possible resources as the observation in [14]. We also discuss the second-level-ciphertext security of Hanaoka et al. [14], and show that their security model is strictly weaker than ours.

3.1 Unidirectional Single-Hop Proxy Re-Encryption

Definition 6. (Unidirectional Single-Hop PRE [18]). *A unidirectional single-hop PRE scheme is a tuple of algorithms $\Pi = (\mathbf{Setup}, \mathbf{KGen}, \mathbf{ReKey}, \mathbf{Enc_1}, \mathbf{Enc_2}, \mathbf{ReEnc}, \mathbf{Dec_1}, \mathbf{Dec_2})$ for message space $\mathcal{M}$:*

- $\mathbf{Setup}(1^\lambda) \to PP$. *On input security parameter 1^λ, the setup algorithm outputs the public parameters PP.*
- $\mathbf{KGen}(PP) \to (pk, sk)$. *On input parameters, the key generation algorithm outputs a public key pk and a secret key sk.*
- $\mathbf{ReKey}(PP, sk_i, pk_j) \to rk_{i \to j}$. *Given a secret key sk_i and a public key pk_j, this algorithm outputs a unidirection re-encryption key $rk_{i \to j}$.*
- $\mathbf{Enc}_1(PP, pk, m) \to CT$. *On input a public key pk and a message $m \in \mathcal{M}$, the encryption algorithm outputs a first level ciphertext CT that cannot be re-encrypted for another party.*
- $\mathbf{Enc}_2(PP, pk, m) \to CT$. *On input a public key pk and a message $m \in \mathcal{M}$, the encryption algorithm outputs a second level ciphertext CT that can be re-encrypted into a first level one (intended for a possibly different receiver) using the suitable re-encryption key.*
- $\mathbf{ReEnc}(PP, rk_{i \to j}, CT_i) \to CT_j$. *Given a re-encryption key from i to j and an original ciphertext for i, the re-encryption algorithm outputs a first level ciphertext for j or the symbol $\perp$.*

– $\mathbf{Dec}_1(PP, sk, CT) \to m$. *Given a secret key sk and a first level ciphertext CT, the decryption algorithm outputs a message $m \in \mathcal{M}$ or the symbol $\perp$.*
– $\mathbf{Dec}_2(PP, sk, CT) \to m$. *Given a secret key sk and a second level ciphertext CT, the decryption algorithm outputs a message $m \in \mathcal{M}$ or the symbol $\perp$.*

To lighten notations, from now, we will omit the public parameters PP as the input of the algorithms.

For all $m \in \mathcal{M}$ and all pair $(pk_i, sk_i), (pk_j, sk_j)$ these algorithms should satisfy the following conditions of correctness:

$$\mathbf{Dec}_1(sk_i, \mathbf{Enc}_1(pk_i, m)) = m;$$
$$\mathbf{Dec}_2(sk_i, \mathbf{Enc}_2(pk_i, m)) = m;$$
$$\mathbf{Dec}_1(sk_j, \mathbf{ReEnc}(\mathbf{ReKey}(sk_i, pk_j), \mathbf{Enc}_2(pk_i, m))) = m.$$

3.2 Security Models for Unidirectional Single-Hop Proxy Re-Encryption

In this section, we present a CCA security definition for PRE, which naturally extends that of [8,14].

Definition 7. (Game Framework of Chosen-Ciphertext Security).

Setup. *The challenger $\mathcal{C}$ takes a security parameter λ and executes the setup algorithm to get the system parameter PP. $\mathcal{C}$ executes the key generation algorithm n_{un} times resulting a list L_{un} of public/private keys $\mathcal{PK}_{un}, \mathcal{SK}_{un}$ and executes the key generation algorithm n_{corr} times resulting a list L_{corr} of public/private keys $\mathcal{PK}_{corr}, \mathcal{SK}_{corr}$. Next, $\mathcal{C}$ picks a challenge user's key pair $(pk_{i^*}, sk_{i^*}) \leftarrow$ $\mathbf{KGen}(PP)$. $\mathcal{A}$ gets PP, $\mathcal{PK} = (\mathcal{PK}_{un}, \mathcal{PK}_{corr})$, $\mathcal{SK}_{corr}$, and the challenge public key pk_{i^*}.*

Phase 1. *$\mathcal{A}$ adaptively queries to oracles $\mathcal{O}_{rk}$, $\mathcal{O}_{re}$, $\mathcal{O}_{dec_1}$, and $\mathcal{O}_{dec_2}$:*
– *$\mathcal{O}_{rk}$ takes (pk_i, pk_j) and returns a re-encryption key $rk_{i\to j} \leftarrow \mathbf{ReKey}(sk_i, pk_j)$.*
– *$\mathcal{O}_{re}$ takes public keys pk_i, pk_j, and a second level ciphertext CT_i, then returns a re-encryption of CT_i from pk_i to pk_j.*
– *$\mathcal{O}_{dec_1}$ takes a public key pk and a first level ciphertext CT, then returns the decryption of CT using the private key with respect to pk if $pk \in \mathcal{PK} \cup \{pk_{i^*}\}$; otherwise returns symbol $\perp$.*
– *$\mathcal{O}_{dec_2}$ takes a public key pk and a second level ciphertext CT, then returns the decryption of CT using the private key with respect to pk if $pk \in \mathcal{PK} \cup \{pk_{i^*}\}$; otherwise returns $\perp$.*

Challenge. *When $\mathcal{A}$ decides that Phase 1 is over, it also decides which type of ciphertext is for the challenge: first level (original or re-encrypted) or second level. In the cases that challenge ciphertext is original (first level or second level) one, $\mathcal{A}$ outputs two equal-length $m_0, m_1 \in \mathcal{M}$. Challenger $\mathcal{C}$ flips a random coin $\sigma \in \{0, 1\}$, and sends to $\mathcal{A}$ a challenge ciphertext CT^* depending on pk_{i^*} and m_σ. In the case that challenge ciphertext is a re-encrypted ciphertext, $\mathcal{A}$ outputs a (corrupted or not) public key $pk_{i'}$, and two "good messages" CT_0, CT_1 which can be re-encrypted from $pk_{i'}$ to pk_{i^*}. Challenger $\mathcal{C}$ flips a random coin $\sigma \in \{0, 1\}$, and sends to $\mathcal{A}$ a challenge ciphertext $CT^* \leftarrow \mathbf{ReEnc}(rk_{i'\to i^*}, CT_\sigma)$.*

Phase 2. *$\mathcal{A}$ issues queries as in Phase 1.*
Guess. *Finally, $\mathcal{A}$ outputs a guess $\sigma' \in \{0, 1\}$.*

The precise conditions of the attacks to second and first level ciphertexts are described separately as follows.

CCA Security of Second Level Ciphertext. Intuitively speaking, in this model the adversary $\mathcal{A}$ challenges with an untransformed ciphertext encrypted by $\mathbf{Enc}_2$ for a target user i^*. In a PRE scheme, however, $\mathcal{A}$ can ask for the re-encryption of many ciphertexts or even a set of re-encryption keys. These queries are allowed as long as they would not allow $\mathcal{A}$ to decrypt trivially. For examples, $\mathcal{A}$ should not get the re-encryption key from user i^* to user j if the secret key of user j has been compromised; however, $\mathcal{A}$ can certainly get a re-encryption of the challenge ciphertext from user i^* to user j as long as j is an honest user and the decryption oracle of user j has not been queried with the resulting transformed ciphertext. This explains the intuition behind the notion of derivative and the associated restrictions.

Definition 8 (2nd-level-CCA Security [11]). *For 2nd-level-CCA security, the adversary $\mathcal{A}$ plays the CCA game with the challenger $\mathcal{C}$ as in Definition 7, where the challenge ciphertext is formed by $CT^* \leftarrow \mathbf{Enc}_2(pk_{i^*}, m_\sigma)$, and $\mathcal{A}$ has the following additional constraints: 1. $\mathcal{O}_{rk}(pk_{i^*}, pk_j)$ is only allowed if pk_j is uncorrupt key.*

2. If $\mathcal{A}$ issues $\mathcal{O}_{re}(pk_i, pk_j, CT_i)$ where pk_j is corrupted key, (pk_i, CT_i) cannot be a derivative *of (pk_{i^*}, CT_{i^*}) (to be defined later).*

3. $\mathcal{O}_{dec_1}$ is only allowed if (pk, CT) is not a derivative *of (pk_{i^*}, CT_{i^*}).*
We define $\mathcal{A}$'s advantage in attacking the PRE scheme at level 2 as $\mathrm{Adv}^{\text{second-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda) = \left|\Pr[\sigma' = \sigma] - 1/2\right|$. A unidirectional PRE scheme is defined to be 2nd-level-CCA secure, if for any PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}^{\text{second-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda)$ is negligible.

Definition 9 (Derivative for Chosen-Ciphertext Security [11]). *Derivatives of (pk_{i^*}, CT_{i^*}) in the CCA setting is defined as below:*

1. *Reflexivity: (pk_{i^*}, CT_{i^*}) is a derivative of itself.*
2. *Derivative by re-encryption: If $\mathcal{A}$ has issued a re-encryption query (pk, pk', CT) and obtained the resulting re-encryption ciphertext CT', then (pk', CT') is a derivative of (pk, CT).*
3. *Derivative by re-encryption key: If $\mathcal{A}$ has issued a re-encryption key generation query (pk, pk') to obtain the re-encryption key rk, and $CT' \leftarrow \mathbf{ReEnc}(rk, CT)$, then (pk', CT') is a derivative of (pk, CT).*

CCA Security of First Level Ciphertext. The above definition provides adversaries with a second level ciphertext in the challenge phase. A complementary definition of security captures their inability to distinguish first level ciphertexts as well.

For single-hop schemes, $\mathcal{A}$ is granted access to all the re-encryption keys in this definition. Since first level ciphertexts cannot be re-encrypted, there is indeed no reason to keep attackers from obtaining all the honest-to-corrupt re-encryption keys. The re-encryption oracle becomes useless since all the re-encryption keys are available to $\mathcal{A}$.

Definition 10 (1st-level-CCA Security). *For 1st-level-CCA security, the adversary $\mathcal{A}$ plays the CCA-PRE game with the challenger $\mathcal{C}$ as in Definition 7, where the challenge ciphertext is formed as follows.*

1. In the case of original ciphertext, $CT^* \leftarrow \mathbf{Enc}_1(pk_{i^*}, m_\sigma)$.

2. In the case of re-encrypted ciphertext, $CT^* \leftarrow \mathbf{ReEnc}(rk_{i' \to i^*}, CT_\sigma)$.

$\mathcal{A}$ has the only constraint that: $\mathcal{O}_{dec_1}(pk_{i^*}, CT^*)$ is not allowed.

We define $\mathcal{A}$'s advantage in attacking the PRE scheme at level 1 as $\mathrm{Adv}^{\texttt{first-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda) = \left|\Pr[\sigma' = \sigma] - 1/2\right|$. A unidirectional PRE scheme is defined to be 1st-level-CCA secure, if for any PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}^{\texttt{first-CCA}}_{\mathrm{PRE},\mathcal{A}}(\lambda)$ is negligible.

Definition 11 (PRE-CCA Security). *We say a PRE scheme is PRE-CCA secure if the scheme is 1st-level-CCA and 2nd-level-CCA secure.*

Master Secret Security. Master Secret Security is considered in Ateniese et al. [3] which captures the intuition that, even if a dishonest proxy colludes with the delegatee, they still cannot derive the delegator's private key in full. The attack mode is quite simple and can be covered by the nontransformable / first-level ciphertext security (see *e.g.* [18]). The reason behind is easy to see there is no restriction in the re-encryption key generation queries, and decryption is easy when the adversary can derive the delegator's private key in full.

3.3 Discussion on the Previous Security Models [8,14]

In comparison with the security model of [8], ours is strengthened by giving an adversary all the possible resources as the observation in [14]. In particular, we allow the adversary to make both first and second level decryption queries.

In [14], Hanaoka et al. proposed a variant of the CCA security definition for unidirectional PRE, which naturally extends the RCCA one given in [18]. Then, they presented the first generic construction of a CCA-secure (in the sense of their definition) PRE scheme. On the discussion of difference from previous security definitions they explained why theirs is stronger than that of the RCCA security in [18]. They also showed that the observation of omitting second level decryption queries in previous definitions is incorrect (see [14] for details). However they did not give any comparison about the strength of those definitions with theirs. In this section, we will point out a gap between their definition with others, which includes ours. In particular, we will show that their security model of second level ciphertext is even weaker than ours by constructing a PRE scheme which is secure in their security model, but insecure in ours.

Using a secure PRE scheme $\mathbf{\Pi}$ (in the sense of their definition, which we denote by wCCA security) as a building block, we construct a PRE scheme $\mathbf{\Pi}'$ as follows:

– The second level encryption algorithm for $\mathbf{\Pi}'$ first runs the second level encryption algorithm for $\mathbf{\Pi}$, generating a second level ciphertext $\hat{C}$, and outputs $CT = (\hat{C}||0)$ (i.e., 0 is attached).
– The second level decryption algorithm $\mathbf{\Pi}'.\mathbf{Dec}_2$ for $\mathbf{\Pi}'$ with input $CT = (\hat{C}||a)$ does: if $a = 0$ then decrypts $\hat{C}$ with the underlying second level decryption algorithm $\mathbf{\Pi}.\mathbf{Dec}_2$; otherwise rejects by outputting the symbol $\perp$.

– The re-encryption algorithm $\mathbf{\Pi}'.\mathbf{ReEnc}$ with input $CT = (\hat{C}||a)$ first re-encrypts $\hat{C}$ with the underlying second level decryption algorithm to obtain re-encrypted ciphertext $r\hat{C}$, then outputs $rCT = (r\hat{C}||a)$ as re-encrypted ciphertext.
– The first level decryption algorithm $\mathbf{\Pi}'.\mathbf{Dec}_1$ for $\mathbf{\Pi}'$ ignores the last bit and decrypts $\hat{C}$ with the underlying first level decryption algorithm.
– The other algorithms for $\mathbf{\Pi}'$ are the same as those for $\mathbf{\Pi}$.

Next, we show that an adversary $\mathcal{A}$ can break the 2nd-level-CCA security (the security in the sense of our definition) of $\mathbf{\Pi}'$ by doing as follows:

1. After receiving the challenge ciphertext $CT_{i^*} = (\hat{C}_{i^*}||0)$, $\mathcal{A}$ queries $\mathcal{O}_{rk}$ to obtain a valid re-encryption key $rk_{i^* \to j}$ from challenger to uncorrupted user j.
2. $\mathcal{A}$ re-encrypts new ciphertext $CT'_{i^*} = (\hat{C}_{i^*}||1)$ using the above re-encryption key, and obtains a re-encrypted ciphertext $rCT_j \leftarrow \mathbf{ReEnc}(rk_{i^* \to j}, CT'_{i^*})$.
3. $\mathcal{A}$ issues a decryption oracle query under pk_j to decrypt the re-encrypted ciphertext rCT_j, and the result is the message encrypted in CT_{i^*}. Note that, $CT'_{i^*}(= (\hat{C}_{i^*}||1)) \neq CT_{i^*}(= (\hat{C}_{i^*}||0))$, so (pk_j, CT_j) is not a derivative of (pk_{i^*}, CT_{i^*}) and this decryption query is not restricted in our security model.

On the other hand, the wCCA-security of the underlying scheme $\mathbf{\Pi}$ guarantees that $\mathbf{\Pi}'$ is wCCA-secure.

From the above example, it is easy to see the gap between their security definition and ours is the restriction on the first level decryption queries that : if $\mathcal{A}$ has asked a re-encryption key query $(pk_{i^*}, pk_i \in \mathcal{PK})$ previously and $\mathbf{Dec}_1(sk_i, \hat{c}) \in \{m_0, m_1\}$, then the challenger returns the special symbol **test** to $\mathcal{A}$. This restriction covers the third condition in definition of Derivative (Definition 9). Because of this restriction, we succeed in constructing the above PRE scheme. Therefore, the security definition proposed in [14] indeed only captures a somewhat strengthened notion of the RCCA security.

4 Analysis of the Shao-Liu-Zhou's Scheme

Shao, Liu, and Zhou [21] proposed a construction of PRE and claimed that their scheme achieves key privacy without losing the CCA security in the standard model. Unfortunately, this scheme is actually not CCA secure. In this section, we describe our attack to break its CCA security (the details are showed in the full version of this paper). See [21] for more details of Shao-Liu-Zhou's scheme, which we denote by SLZ (due to the lack of space, we do not show it here).

Before describing our attack, we briefly explain how the re-encryption key is generated in the SLZ scheme. There are three components in the re-encryption key, where only the first component $(rk^{(1)}_{pk,pk'} = (h^{1/y})^{xr})$ is computed using the secret key of the delegator (i.e., $x \in sk$), where $h^{1/y}$ is from the delegatee's the public key and r is random chosen by the delegator. The other components $(rk^{(2)}_{pk,pk'}$ and $rk^{(3)}_{pk,pk'})$ are computed using parameters, the public key of the delegatee, and randoms chosen by the proxy.

It is easy to see that, everyone can compute the re-encryption key from the delegator to himself (i.e. $rk_{pk,pk} = (h^{1/x})^{xr})$ without any knowledge of the delegator's secret key. Specifically, the adversary first chooses random $r \xleftarrow{R} \mathbb{Z}_p$ then computes the first component as h^r (because $h^r = h^{(x/x)\cdot r}$). The two latter are not depend on the delegator's secret key, so it is easily computed as in the re-key generation algorithm. Using the computed re-encryption key , the adversary can re-encrypt the challenge ciphertext, then revokes the re-encrypted ciphertext to the decryption oracle $\mathcal{O}_{dec_1}$ to obtain the plaintext (note that, this is not restricted in the CCA security model).

5 The Proposed PRE Scheme

In this section, we first propose a new PRE scheme, and then show that it meets the 1st-level-CCA and the 2nd-level-CCA security.

5.1 Construction

To achieve the full CCA security, we start from the following observations which are important and necessary principles for designing CCA-secure unidirectional PRE schemes:

1. The validity of the original ciphertexts can be publicly verifiable by everyone including the proxy; otherwise, it will suffer from an attack as illustrated in [12].
2. It should also be impossible for the adversary to transform the second level ciphertext to the first level one without knowledge of delegator's secret key or re-encryption key; otherwise, it will suffer from an attack as applied to the SLZ scheme (Section 4).
3. For the first level ciphertext CT_j re-encrypted from a second level ciphertext CT_i, it should not exhibit any component of CT_i; otherwise, it will fail in achieving the 1st-level-CCA security.

We will explain how our scheme follows these principles in the following description of our scheme.

Setup: $(p, g, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, H) \xleftarrow{\text{R}} \text{Setup}(1^\lambda)$, where $(\mathbb{G}, g, p, H)$ is random parameters obtained by running the parameter algorithm $\mathsf{Gen}(1^\lambda)$, and $H : \mathbb{G} \to \{0,1\}^{\ell(n)}$ is a random instance of a hash function such that the GHDH assumptions holds relative to Gen. Choose $g_1, h, u, v, d \in_{\text{R}} \mathbb{G}$, and two collision-resistant hash functions $\mathsf{TCR} : \mathbb{G} \times \mathbb{G}_T \to \mathbb{Z}_p$, $\mathsf{TCR}' : \mathbb{G} \to \mathbb{Z}_p$. **SYM** is a symmetric encryption scheme of which the key space is $\{0,1\}^{\ell(n)}$. Return $PP = (p, \mathbb{G}, \mathbb{G}_T, g, h, g_1, u, v, d, \mathbf{e}, H, \mathsf{TCR}, \mathsf{TCR}')$.

KGen: On input $PP = (p, \mathbb{G}, \mathbb{G}_T, g, h, g_1, u, v, d, \mathbf{e}, H, \mathsf{TCR}, \mathsf{TCR}')$, choose $x, y \xleftarrow{\text{R}} \mathbb{Z}_p$, and output (pk, sk), where $pk = (g^x, g_1^{x^2}, g^y)$, $sk = (x, y)$.

ReKey: On input a private key $sk_i = (sk_{i,1}, sk_{i,2})$ and a public key $pk_j = (pk_{j,1}, pk_{j,2}, pk_{j,3})$, the algorithm outputs $rk_{i\to j} = pk_{j,2}^{1/sk_{i,1}} (= g_1^{sk_{j,1}^2/sk_{i,1}} = g_1^{x_j^2/x_i})$.

Enc$_1$: Given $pk_i = (pk_{i,1}, pk_{i,2}, pk_{i,3})$ and a message $m \in \mathbb{G}_\text{T}$, randomly choose $r, R, r', s \xleftarrow{\text{R}} \mathbb{Z}_p$ and compute $C_2 = h^r$, $C_3 = \mathbf{e}(g, g_1)^r \cdot m$, $t = \mathsf{TCR}(C_2, C_3)$, $C_4 = (u^t v^s d)^r$, $C_5 = s$, $C_6 = pk_{i,2}^{rR}$, $C_7 = pk_{i,2}^{R}$, $C_8 = g^{1/R}$, $CT_i' = C_2||C_3||...||C_8$, $A = g^{r'}$, $t' = \mathsf{TCR}'(A)$, $B = (pk_{i,3}^{t'} \cdot h)^{r'}$, $C \leftarrow \mathbf{SYM}.\mathbf{Enc}(H(pk_{i,3}^{r'}), CT_i')$. Finally, output $CT_i = (A, B, C)$.

Enc$_2$: Given $pk_i = (pk_{i,1}, pk_{i,2}, pk_{i,3})$ and a message $m \in \mathbb{G}_T$, choose $r, s \xleftarrow{R} \mathbb{Z}_p$ and compute: $C_1 = pk_{i,1}^r$, $C_2 = h^r$, $C_3 = \mathbf{e}(g, g_1)^r \cdot m$, $t = \mathsf{TCR}(C_2, C_3)$, $C_4 = (u^t v^s d)^r$, $C_5 = s$. Finally, output $CT = (C_1, C_2, C_3, C_4, C_5)$.

ReEnc: On input ReKey $rk_{i \to j}$, an original ciphertext CT_i and a pair of public keys pk_i, pk_j, this algorithm does as follows:
Compute $t = \mathsf{TCR}(C_2, C_3)$, and check the validity of the ciphertext CT_i by testing whether the following equalities hold (if all of the verifications pass, we write $Check(CT_i) = 1$):

$$\mathbf{e}(h, C_1) = \mathbf{e}(C_2, pk_{i,1}) \tag{1}$$
$$\mathbf{e}(h, C_4) = \mathbf{e}(C_2, u^t v^s d) \tag{2}$$

If one of the above verifications fail, output $\perp$ indicating an invalid ciphertext. Otherwise, choose $R, r' \xleftarrow{R} \mathbb{Z}_p$, and compute $C_6 = C_1^R$; $C_7 = pk_{i,1}^R$; $C_8 = rk_{i \to j}^{1/R}$; $CT_i' = C_2 || C_3 || ... || C_8$; $A = g^{r'}$, $t' = \mathsf{TCR}'(A)$, $B = (pk_{i,3}^{t'} \cdot h)^{r'}$, $C \leftarrow$ **SYM.Enc**$(H(pk_{i,3}^{r'}), CT_i')$. Finally, output $CT_j = (A, B, C)$.

Dec$_2$: Given sk_i and a ciphertext CT, do as follows:
Parse $CT = (C_1, C_2, C_3, C_4, C_5)$, if this is not the case, output $\perp$ and halt. Compute $t = \mathsf{TCR}(C_2, C_3)$, and check the validity of the ciphertext CT by testing whether $Check(CT) = 1$. If the verifications fail, output $\perp$ and halt. Otherwise, output $m = \frac{C_3}{\mathbf{e}(C_1, g_1)^{1/sk_{i,1}}}$.

Dec$_1$: Given (pk_j, sk_j), and a ciphertext CT_j, do as follows:
Compute $t' = \mathsf{TCR}(A)$, if $\mathbf{e}(A, pk_{i,3}^{t'} \cdot h) \neq \mathbf{e}(g, B)$ then output $\perp$ indicating an invalid ciphertext. Otherwise, do the following.
Compute $CT_i' = $ **SYM.Dec**$(H(A^{sk_{j,2}}), C)$;
Parse $CT_i' = C_2 || C_3 || ... || C_8$, if this is not the case, output $\perp$ and halt.
Else, compute $t = \mathsf{TCR}(C_2, C_3)$ and check the validity of the ciphertext CT_i by testing whether the following equalities hold:

$$\mathbf{e}(h, C_4) = \mathbf{e}(C_2, u^t v^{C_5} d) \tag{3}$$
$$\mathbf{e}(C_7, C_8) = \mathbf{e}(pk_{j,2}, g) \tag{4}$$
$$\mathbf{e}(h, C_6) = \mathbf{e}(C_2, C_7) \tag{5}$$

If one of the above verifications fails, output $\perp$ indicating an invalid ciphertext. Otherwise, output $m = \frac{C_3}{\mathbf{e}(C_6, C_8)^{1/sk_{j,1}^2}}$.

5.2 Security Analysis

The intuition of the CCA security of our scheme can be seen from the below properties.

1. The validity of the original ciphertexts can be publicly verifiable by everyone including the proxy; otherwise, it will suffer from an attack as illustrated in [12]. For our scheme, the ciphertext component C_4, C_5 in the original ciphertext $CT = (C_1, C_2, C_3, C_4, C_5)$ can be viewed as a signature signing the message C_1, C_2, C_3, that is how we get public verifiability.

2. It should be impossible for the adversary to transform the second level ciphertext to the first level one without knowledge of delegator's secret key or re-encryption key; otherwise, it only yields the RCCA security. In our scheme, the component $C_8 = rk_{i \to j}^{1/R}$ is computed using the re-encryption key and completely hidden in C, so the adversary cannot transform the second level ciphertext to ciphertext re-encrypted by **ReEnc** if he has no knowledge of the re-encryption key. The adversary also cannot transform the second level ciphertext to ciphertext encrypted by $\mathbf{Enc}_1$ without any knowledge of delegator's secret key or random component r used in the original ciphertext, because the component $C_6 = pk_{i,2}^{rR}$ is computed independently of $\mathbf{Enc}_2$, and completely hidden in C.
3. It should be impossible for the adversary to compute the re-encryption key from the target user i^* to itself (i.e., $rk_{i^* \to i^*}$), otherwise it will suffer from an attack as applied to the SLZ scheme (Section 4). This follows from Lemma 1.
4. For the first level ciphertext CT_j re-encrypted from a second level ciphertext CT_i, it should not exhibit any component of CT_i; otherwise, it will fail in achieving the CCA-security of **ReEnc** (i.e., the 1st-level-CCA security). In our scheme, all of the components from the original ciphertext are hidden in C. Furthermore, we use the KEM/DEM scheme of Kiltz [16] in the re-encryption algorithm to guarantee the CCA security of re-encrypted ciphertext.
5. In our scheme, **ReEnc** and $\mathbf{Dec_2}$ use the same algorithm of checking the validity of the second-level ciphertext CT_i (i.e. $Check(CT_i)$). So in the security game, providing the adversary with a second level decryption oracle is useless. Indeed, ciphertexts encrypted under public keys from $\mathcal{PK}_{un}$ can be re-encrypted for corrupted users by using the re-encryption oracle. Besides, the second level ciphertext under pk_{i^*} can be translated for other honest users by using $rk_{i^* \to j}$ (where $pk_j \in \mathcal{PK}_{un}$) and the resulting ciphertext can be queried for decryption at the first level by using $\mathcal{O}_{dec_1}$. This does not contradict the observation of Hanaoka et al. [14].

Lemma 1. *$\mathcal{A}$ cannot make $rk_{i^* \to i^*} = g_1^{sk_{i^*,1}}$ without knowledge of secret key $sk_{i^*,1}$, assuming the 2-AmCDH problem is hard.*

Proof (Sketch). Suppose there exists an adversary $\mathcal{A}$ who can compute $rk_{i^* \to i^*} = g_1^{x_{i^*}}$. We build an algorithm $\mathcal{B}$ which is, given a 2-AmCDH instance $(g, g^a, g^b, g^{ba^2}, g^{\frac{b}{a}}) \in \mathbb{G}^5$, solving the 2-AmCDH problem using $\mathcal{A}$.

Setup: $\mathcal{B}$ chooses $c, u, v, d \overset{\text{R}}{\leftarrow} \mathbb{Z}_p$, sets $g_1 := g^b$, and computes $h = g^c$. The other parameters are chosen as in the algorithm **Setup**. The public parameters are $PP = (p, \mathbb{G}, \mathbb{G}_T, g, h, g_1, u, v, d, \mathbf{e}, H, \mathsf{TCR}, \mathsf{TCR}')$.

– For the challenge key, $\mathcal{B}$ chooses randomly $x_{i^*}, y_{i^*} \overset{\text{R}}{\leftarrow} \mathbb{Z}_p$, and sets $pk_{i^*} = ((g^a)^{x_{i^*}}, (g^{ba^2})^{x_{i^*}^2}, g^{y_{i^*}})$ (meaning that $sk_{i^*} = (ax_{i^*}, y_{i^*})$).

– For corrupted-keys and uncorrupted-keys, $\mathcal{B}$ chooses randomly $x_i, y_i \overset{\text{R}}{\leftarrow} \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, (g^b)^{x_i^2}, g^{y_i}), sk_i = (x_i, y_i)$.

ReKey Oracle $\mathcal{O}_{rk}(pk_i, pk_j)$**:** $\mathcal{B}$ does as follows.

1. If $pk_i, pk_j \in \mathcal{PK}_{corr}$ or $pk_i, pk_j \in \mathcal{PK}_{un}$ then $\mathcal{B}$ returns $rk_{i \to j} = g_1^{x_j^2/x_i}$.
2. If $pk_i \in \mathcal{PK}_{un} \wedge pk_j \in \mathcal{PK}_{corr}$, then $\mathcal{B}$ returns $rk_{i \to j} = g_1^{x_j^2/x_i}$.

3. If $pk_i \in \mathcal{PK}_{corr} \wedge pk_j \in \mathcal{PK}_{un} \cup \{pk_{i^*}\}$ then $\mathcal{B}$ returns $rk_{i\to j} = pk_{j,2}^{1/x_i}$.
4. If $pk_i = pk_{i^*}, pk_j \in \mathcal{PK}_{un}$ then $\mathcal{B}$ returns $rk_{i^*\to j} = (g^{\frac{b}{a}})^{x_j^2/x_{i^*}} = g_1^{sk_{j,1}^2/sk_{i^*,1}}$.
5. If $pk_i \in \mathcal{PK}_{un}, pk_j = pk_{i^*}$ then $\mathcal{B}$ returns $rk_{i\to i^*} = (g^{ba^2})^{x_{i^*}^2/x_i} = g_1^{sk_{i^*,1}^2/sk_{i,1}}$.
6. If $pk_i = pk_{i^*}, pk_j \in \mathcal{PK}_{corr}$ then $\mathcal{B}$ outputs $\perp$.

First Decryption Oracle $\mathcal{O}_{dec_1}(pk_i, CT_i)$**:** If $pk_i \neq pk_{i^*}$, $\mathcal{B}$ does as the $\mathbf{Dec}_1$ to decrypt the ciphertext CT_i using the secret key $sk_i = (x_i, y_i)$. Otherwise, $\mathcal{B}$ does as follows. $\mathcal{B}$ first computes $t' = \mathsf{TCR}(A)$, and checks if $\mathbf{e}(A, pk_{i,3}^{t'} \cdot h) \overset{?}{=} \mathbf{e}(g, B)$. If this verification fails, output $\perp$ indicating an invalid ciphertext. Otherwise, it does the following.

Compute $CT_i' = \mathbf{SYM}.\mathbf{Dec}(H(A^{y_i}), C)$, and parse $CT_i' = C_2||C_3||...||C_8$. Then, $\mathcal{B}$ computes $t = \mathsf{TCR}(C_2, C_3)$ and does as follows:

1. Check the validity of the ciphertext CT_i as Eq. (3) - (5) in $\mathbf{Dec}_1$.
2. Check if $(C_2, C_3) \overset{?}{=} (C_2^*, C_3^*)$ and $t \neq t^*$. If so, abort and output a random bit.
3. Check if $t + sx_v + x_d = 0$. If so, $\mathcal{B}$ aborts and outputs a random bit.

If all of these verifications pass, then there exists $r \in \mathbb{Z}_p$ such that $C_1 = pk_{i,1}^r, C_4 = (u^t v^s d)^r, s = C_5$. If $(C_2, C_3, C_4, C_5) = (C_2^*, C_3^*, C_4^*, C_5^*)$, $\mathcal{B}$ outputs $\perp$ which deems CT_i as a derivative of the challenge pair (pk_{i^*}, CT^*). We now assume $C_3 \neq C_3^*$. Since $C_2 = h^r = g^{cr}$, we have $C_2^{1/c} = g^r$. Therefore $\mathcal{B}$ can compute $m = \frac{C_3}{\mathbf{e}(g^r, g_1)}$, then returns it to $\mathcal{A}$.

Re Encryption Oracle $\mathcal{O}_{re}(rk_{i\to j}, CT_i)$**:** $\mathcal{B}$ first checks the validity of CT_i, then executes the algorithm $\mathbf{ReEnc}$ with the re-encryption key computed as in $\mathcal{O}_{rk}(pk_i, pk_j)$ and the ciphertext CT_i. Finally returns $CT_j \leftarrow \mathbf{ReEnc}(rk_{i\to j}, CT_i)$.

Challenge. $\mathcal{A}$ outputs two equal-length messages $m_0, m_1 \in \mathbb{G}_T$. $\mathcal{B}$ flips a coin $\sigma \overset{\text{R}}{\leftarrow} \{0, 1\}$ and encrypts m_σ using the public key pk_{i^*}.

Whenever, $\mathcal{A}$ outputs $rk_{i^*\to i^*} = g_1^{sk_{i^*,1}}$ (meaning that $rk_{i^*\to i^*} = g^{abx_{i^*}}$), then $\mathcal{B}$ outputs $g^{ab} = k_{i^*\to i^*}^{\frac{1}{x_{i^*}}}$ as the answer of the 2-AmCDH problem.

This completes the description of the simulation.

It is easy to see that the simulation is perfect, therefore we have the probability that $\mathcal{A}$ can compute $rk_{i^*\to i^*}$ is bound by $\mathrm{Adv}_{\mathcal{B}}^{\text{2-AmCDH}}(\lambda)$.

The Lemma follows. □

Theorem 1. *Our scheme meets the 2nd-level-CCA security, assuming the hash function H is target collision resistant, the 6-AmDBDH assumption holds in groups $(\mathbb{G}, \mathbb{G}_T)$, and the 2-AmCDH problem is hard.*

Proof (Sketch). We prove that our proposed scheme is 2nd-level-CCA secure under the 6-AmDBDH assumption. We build an algorithm $\mathcal{B}$ which is, given a modified 6-AmDBDH instance $(g, g^a, g^b, g^c, g^{a/c}, g^{c^2}, g^{ac}, g^{ac^2}, g^{ac^3}, g^{ac^4}, Q) \in \mathbb{G}^{10} \times \mathbb{G}_T$, deciding whether $Q = \mathbf{e}(g, g)^{ab/c^2}$, using the adversary $\mathcal{A}$ who can break the 2nd-level-CCA security of the scheme.

In the setup phase, $\mathcal{B}$ chooses $\omega, x_v, x_d, y_u, y_v, y_d \overset{\text{R}}{\leftarrow} \mathbb{Z}_p$, sets $g_1 := g^a$, and computes $h = (g^{c^2})^{\omega}, u = g \cdot (g^{c^2})^{y_u}, v = g^{x_v} \cdot (g^{c^2})^{y_v}, d = g^{x_d} \cdot (g^{c^2})^{y_d}$. The others parameters are chosen as in the algorithm **Setup**. The public parameters are $PP = (p, \mathbb{G}, \mathbb{G}_T, g, h, g_1, u, v, d, \mathbf{e}, H, \mathsf{TCR}, \mathsf{TCR}')$.

For the challenge key, $\mathcal{B}$ chooses $x_{i^*}, y_{i^*} \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$, and defines $pk_{i^*} = ((g^{c^2})^{x_{i^*}}, (g^{ac^4})^{x_{i^*}^2}, g^{y_{i^*}})$ (meaning that $sk_{i^*} = (c^2 x_{i^*}, y_{i^*})$). For uncorrupted-key, $\mathcal{B}$ chooses $x_i, y_i \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$, and defines $pk_i = ((g^c)^{x_i}, (g^{ac^2})^{x_i^2}, g^{y_i})$ (meaning that $sk_i = (cx_i, y_i)$). For corrupted-keys, $\mathcal{B}$ chooses $x_i, y_i \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$, and defines $pk_i = (g^{x_i}, (g^a)^{x_i^2}, g^{y_i})$, $sk_i = (x_i, y_i)$. By this setting, $\mathcal{B}$ can easily simulate the actions of the re-key oracle $\mathcal{O}_{rk}$.

To answer re-encryption oracle queries, $\mathcal{B}$ does as follows.

(1) if $pk_i \neq pk_{i^*}$ or $pk_j \notin \mathcal{PK}_{corr}$: $\mathcal{B}$ computes re-key as above and uses it to re-encrypt the queried message;

(2) If $pk_i = pk_{i^*}, pk_j \in \mathcal{PK}_{corr}$: $\mathcal{B}$ makes use of the technique used in [17,19] to compute g^r. Using g^r, $\mathcal{B}$ computes: $C_6 = (g^r)^{x_{i^*} R'} (= (g^{c^2 x_{i^*} r})^{R'/c^2} = C_1^R)$, where $R = R'/c^2$; $C_7 = g^{x_{i^*} R'} (= (g^{c^2 x_{i^*}})^{R'/c^2} = pk_{i^*,1}^R)$; $C_8 = (g_1^{x_j^2/x_{i^*}})^{1/R'} (= (g_1^{(sk_{j,1}^2/sk_{i^*,1})})^{c^2/R'} = rk_{i^* \to j}^{1/R})$; $CT'_{i^*} = C_2||C_3||...||C_8$; $A = g^{r'}$, $t' = \mathsf{TCR}'(A)$, $B = (pk_{i^*,3}^{t'} \cdot h)^{r'}$, $C \leftarrow \mathbf{SYM}.\mathbf{Enc}(H(pk_{i^*,3}^{r'}), CT'_{i^*})$. Output $CT_j = (A, B, C)$.

To answer decryption oracle queries, $\mathcal{B}$ first computes g^r as the above way (using the technique used in [17,19]), then easily computes $m = \frac{C_3}{\mathbf{e}(g^r, g_1)}$.

In the challenge phase, we make use of the technique used in [17,19] to allow $\mathcal{B}$ to successfully generate the challenge ciphertext for $\mathcal{A}$.

The simulation is perfect, thus as long as Lemma 1, the theorem follows. □

Theorem 2. *Our scheme meets the 1st-level-CCA security, assuming* TCR' *is a target collision resistant hash function, the GHDH problem is hard, and* **SYM** *is CCA-secure.*

In the algorithms $\mathbf{Enc}_1$ and **ReEnc**, we make use of the encryption algorithm of Kiltz's KEM/DEM scheme [16] to mask all of computed components including the second level ciphertext. Therefore, the 1st-level-CCA security of our scheme is implied by the CCA security of Kiltz's KEM/DEM scheme.

The detail proof of Lemma 1 and the above theorems are given in the full version of this paper.

6 Conclusions

We have proposed a full CCA security definition for unidirectional single-hop PRE, which naturally extends that of [8,14], and presented the first PRE scheme that is secure in the sense of this security. Our scheme relies on mild complexity assumptions in bilinear groups without random oracles. It would be interesting to construct a scheme without bilinear maps in the standard model.

Acknowledgements. We thank the anonymous reviewers for their helpful comments.

References

1. Ateniese, G., Benson, K., Hohenberger, S.: Key-Private Proxy Re-encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 279–294. Springer, Heidelberg (2009)

2. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: NDSS (2005)
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. 9(1), 1–30 (2006)
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACMCCS 1993, pp. 62–73. ACM Press (1993)
5. Blaze, M., Bleumer, G., Strauss, M.: Divertible Protocols and Atomic Proxy Cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
6. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
7. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
8. Canard, S., Devigne, J., Laguillaumie, F.: Improving the security of an efficient unidirectional proxy re-encryption scheme. Journal of Internet Services and Information Security 1(2), 140–160 (2011)
9. Canetti, R., Halevi, S., Katz, J.: A Forward Secure Public Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 254–271. Springer, Heidelberg (2003)
10. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: ACM Conference on Computer and Communications Security, pp. 185–194. ACM Press (2007)
11. Chow, S.S.M., Weng, J., Yang, Y., Deng, R.H.: Efficient Unidirectional Proxy Re-Encryption. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 316–332. Springer, Heidelberg (2010)
12. Deng, R., Weng, J., Liu, S., Chen, K.: Chosen-Ciphertext Secure Proxy Re-encryption without Pairings. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 1–17. Springer, Heidelberg (2008)
13. El Gamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
14. Hanaoka, G., Kawai, Y., Kunihiro, N., Matsuda, T., Weng, J., Zhang, R., Zhao, Y.: Generic Construction of Chosen Ciphertext Secure Proxy Re-Encryption. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 349–364. Springer, Heidelberg (2012)
15. Ivan, A., Dodis, Y.: Proxy cryptography revisited. In: NDSS. The Internet Society (2003)
16. Kiltz, E.: Chosen-Ciphertext Secure Key-Encapsulation Based on Gap Hashed Diffie-Hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)
17. Lai, J., Deng, R.H., Liu, S., Kou, W.: Efficient CCA-Secure PKE from Identity-Based Techniques. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 132–147. Springer, Heidelberg (2010)
18. Libert, B., Vergnaud, D.: Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008)
19. Nishimaki, R.: A CCA-secure proxy re-encryption scheme with short ciphertexts. In: SCIS 2011, 3F3-2 in Japanese (2011)
20. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
21. Shao, J., Liu, P., Zhou, Y.: Achieving key privacy without losing CCA security in proxy re-encryption. J. Syst. Software (2011), doi:10.1016/j.jss.2011.09.034
22. Smith, T.: DVD jon: Buy DRM-less tracks from Apple iTunes (January 2005), http://www.theregister.co.uk/2005/03/18/itunes_pymusique

Solving BDD by Enumeration: An Update

Mingjie Liu[1] and Phong Q. Nguyen[2]

[1] Beijing International Center for Mathematical Research, Peking University and Tsinghua University, Institute for Advanced Study, China
[2] INRIA, France and Tsinghua University, Institute for Advanced Study, China
http://www.di.ens.fr/~pnguyen/

Abstract. Bounded Distance Decoding (BDD) is a basic lattice problem used in cryptanalysis: the security of most lattice-based encryption schemes relies on the hardness of some BDD, such as LWE. We study how to solve BDD using a classical method for finding shortest vectors in lattices: enumeration with pruning speedup, such as Gama-Nguyen-Regev extreme pruning from EUROCRYPT '10. We obtain significant improvements upon Lindner-Peikert's Search-LWE algorithm (from CT-RSA '11), and update experimental cryptanalytic results, such as attacks on DSA with partially known nonces and GGH encryption challenges. Our work shows that any security estimate of BDD-based cryptosystems must take into account enumeration attacks, and that BDD enumeration can be practical even in high dimension like 350.

1 Introduction

There is growing interest in cryptography based on hard lattice problems (see the survey [11]). The field started with the seminal work of Ajtai [1] back in 1996, and recently got a second wind with Gentry's breakthrough work [7] on fully-homomorphic encryption. It offers asymptotical efficiency, potential resistance to quantum computers and new functionalities. Most of the provably-secure lattice-based constructions are based on either of the following two average-case problems:

– the Small Integer Solutions (SIS) problem proposed by Ajtai [1], which allows to build one-way functions, collision-resistant hash functions, signature schemes and identification schemes: see [11].
– the Learning with Errors (LWE) problem introduced by Regev [17] (see the survey [18]), which allows to build public-key encryption [17,16], and more powerful primitives such as ID-based encryption [8] and even fully-homomorphic encryption [3].

Both SIS and LWE are provably as hard as certain worst-case lattice problems (see [1,11] for SIS, and [17,16] for LWE), which allows to design many cryptographic schemes with security related to the hardness of lattice problems, without actually dealing explicitly with lattices.

Due to the importance of LWE, it is very important to know what are the best attacks on LWE, especially if one is interested in selecting concrete parameters. And this issue is independent of LWE (quantum or not) reductions [17,16] to lattice problems. At CT-RSA '11, Lindner and Peikert [10] generalized Babai's Nearest Plane algorithm [2] to solve Bounded Distance Decoding (BDD), and claimed that this was the best attack known on Search-LWE. Given a lattice and a target vector unusually close to the lattice, BDD asks to find the closest lattice vector to the target: this basic lattice problem has many applications in cryptanalysis (see [14]), and LWE is simply a particular case of BDD. Despite its importance, it is not obvious at the moment what is the best algorithm for solving BDD in practice: several parameters impact the answer, *e.g.* the dimension, the size and shape of the error error. Until now, the largest BDD cryptanalytical instances ever solved in practice were tackled using the so-called embedding method that heuristically reduces BDD to the unique-shortest vector problem (USVP) (see *e.g.* [13]). And in the past few years, there has been significant process in practical lattice reduction [6,4].

Our Results. We present lattice attacks on Search-LWE which are significantly better than the Lindner-Peikert attack [10]: in practice, the speedup can be as big as 2^{32} for certain parameters considered in [10] (see Table 1). First, we revisit the Lindner-Peikert BDD algorithm (which turns out to be essentially Schnorr's random sampling [19]) by rephrasing it in the pruned-enumeration framework of Gama-Nguyen-Regev (GNR) [6]. This allows us to present a simple randomized variant which already performs much better than the original LP algorithm [10] in the case of LWE, independently of the efficiency of lattice reduction: our randomization is similar to GNR's extreme pruning [6], *i.e.* we use several random reduced bases.

Next, we consider GNR pruned-enumeration algorithms [6] to solve BDD: this provides even better attacks on Search-LWE, and seems to be the method of choice in practice for the general BDD case. We illustrate this point by reporting improved experimental results for attacks on DSA with partially known nonces [15] and the solution of GGH encryption challenges [13,9]. Though enumeration is a classical method to solve BDD, it was unknown how efficient in practice was GNR pruned-enumeration in BDD applications. In the DSA case, pruned enumeration can recover the DSA secret key in a few hours, given each 2 least significant bits of the nonces corresponding to 100 DSA signatures: previously, the best lattice experiment [15] required 3 bits. For GGH encryption challenges, we provide the first partial secret-key recovery in dimensions 200-300: this is a proof-of-concept, and the rest of the secret key could easily be recovered by simply repeating our experiments a few times. And we re-solved the 350-dimensional message-recovery challenge using much weaker lattice reduction: in this case, enumeration is clearly preferable to the embedding method, even in very high dimension like 350.

Our work shows that any security estimate on BDD must take into account enumeration attacks, and that the Lindner-Peikert BDD algorithm [10] does not seem to offer any practical advantage over GNR pruning [6], despite having appeared later.

Road Map. The paper is organized as follows. In Section 2, we provide background. In Section 3, we revisit the Lindner-Peikert algorithm and present our randomized variant. Finally, in Section 4, we consider GNR pruned enumeration [6] to solve BDD, and apply it to LWE, GGH encryption challenges and attacks on DSA with partially known nonces. In Appendix, we provide pseudocode for BDD enumeration.

2 Background

We use row notation for vectors. We denote by $\|\mathbf{v}\|$ the Euclidean norm of a vector $\mathbf{v}$, and by $\mathrm{Ball}_m(R)$ the m-dimensional closed ball of radius R, whose volume is $V_m(R) \sim (\sqrt{\frac{2\pi e}{m}})^m R^m$. We use S^{m-1} to denote the m-dimensional unit sphere. The *fundamental parallelepiped* $\mathcal{P}_{1/2}(B)$ of a matrix $B = (\mathbf{b}_1, \ldots, \mathbf{b}_m)$ is $\{\sum_{i=1}^m x_i \mathbf{b}_i : -\frac{1}{2} \leq x_i < \frac{1}{2}\}$. The *volume* $\mathrm{vol}(L)$ of a lattice L is the m-dimensional volume of $\mathcal{P}_{1/2}(B)$ for any basis B of L.

Orthogonalization. The Gram-Schmidt orthogonalization of B is denoted by $B^* = (\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_m^*)$, where $\mathbf{b}_i^* = \pi_i(\mathbf{b}_i)$, with π_i being the orthogonal projection over $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1})^\perp$. Thus, $\pi_i(L)$ is an $(m+1-i)-$dimensional lattice generated by the basis $(\pi_i(\mathbf{b}_i), \ldots, \pi_i(\mathbf{b}_m))$ with volume $\mathrm{vol}(\pi_i(L)) = \prod_{j=i}^m \|\mathbf{b}_j^*\|$.

Gaussian Heuristic. The Gaussian Heuristic provides a heuristic estimate on the number of lattice points inside a set as follows: Given a "nice" lattice L and a "nice" set S, the number of points in $S \cap L$ is heuristically $\approx \mathrm{vol}(S)/\mathrm{vol}(L)$.

Lattice reduction. Lattice reduction algorithms aim at finding bases with short and nearly orthogonal vectors. Their output quality is usually measured by the Hermite factor $\|\mathbf{b}_1\|/(\mathrm{vol}(L))^{\frac{1}{m}}$, where $\mathbf{b}_1$ is the first vector of the output basis. The experiments of Gama and Nguyen [5] show that the Hermite factor of the best algorithms known is exponential δ^m in the dimension m in practice, and the recent work of Chen and Nguyen [4] provides a correspondence between the exponentiation base δ (the root Hermite factor) and the running time of the best state-of-the-art implementation (BKZ 2.0). This is related to the geometric series assumption (GSA) proposed by Schnorr [20]: for fixed parameters, the norms of the Gram-Schmidt vectors $\mathbf{b}_i^*$ decrease roughly geometrically with i, say $\|\mathbf{b}_i^*\| / \|\mathbf{b}_{i+1}^*\| \approx q$, in which case the root Hermite factor δ is $\approx \sqrt{q}$. In [10], Lindner and Peikert used different running-time estimates of lattice reduction than [4]:

- One is a numerical extrapolation based on their experiments with the BKZ implementation of the NTL library. However, NTL's implementation of BKZ dates back from 1997 and does not take into account recent progress, such as extreme pruning [6]. The state-of-the-art implementation of BKZ developed by Chen and Nguyen [4] achieves several exponential speedups compared to NTL's implementation.
- The other one is used in the tables of [10]. It is simply a conservative lower bound of the first one. They divide the running time by some arbitrary (large) constant, and change (conservatively) the slope of the curve.

We believe it is preferable to use the Chen-Nguyen estimates [4], but any comparison with the LWE algorithm of [10] must also take into account the lattice reduction estimates of [10] for completeness.

Discrete Gaussian Distribution. Let $s > 0$ be real. The discrete (centered) Gaussian distribution over L has density $D_{L,s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x})}{\rho_s(L)}$ where $\mathbf{x} \in L$, $\rho_s(\mathbf{x}) = e^{-\pi\|\mathbf{x}/s\|^2}$ and $\rho_s(L) = \sum_{\mathbf{y}\in L} \rho_s(\mathbf{y})$. Over $\mathbb{Z}^n$, most of the mass is within the ball of radius $O(s\sqrt{n})$.

Bounded Distance Decoding (BDD). Given L and a target $\mathbf{t}$ very "close" to L, BDD asks to find $\mathbf{u} \in L$ minimizing $\|\mathbf{u} - \mathbf{t}\|$. There are many ways to formalize what is meant by very "close": here, we assume that there exists a unique $\mathbf{u} \in L$ such that $\|\mathbf{u} - \mathbf{t}\| \leq \gamma \mathrm{vol}(L)^{1/m}$ for some small given $\gamma > 0$: the smaller γ, the easier BDD. The vector $\mathbf{u} - \mathbf{t}$ is called the BDD error.

The Learning with Errors problem (LWE). The input of LWE is a pair $(A, \mathbf{t} = \mathbf{s}A + \mathbf{e})$ where $A \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{s} \in \mathbb{Z}_q^n$ are chosen uniformly at random, $\mathbf{e} \in \mathbb{Z}_q^m$ is chosen according to some public distribution χ depending on a parameter $\alpha \in (0,1)$. In the original LWE article [17], χ is the integral rounding of a continuous Gaussian distribution: namely, χ is the distribution of the random variable $\lfloor qX \rceil$ mod q, where X is a (continuous) normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$ reduced modulo 1. The Lindner-Peikert article [10] uses instead the discrete Gaussian distribution $\chi = D_{\mathbb{Z}^m,\alpha q}$, and mention (without proof) that LWE hardness results also hold for this LWE variant.

Given $(A, \mathbf{t})$, Search-LWE asks to recover $\mathbf{s}$, while Decision-LWE asks to distinguish $(A, \mathbf{t})$ from a uniformly random $(A, \mathbf{t})$. Regev [17] proved that if $\alpha q \geq 2\sqrt{n}$, Search-LWE is at least as hard as quantumly approximating (Decision)-SVP or SIVP to within $\tilde{O}(n/\alpha)$ in the worst case for dimension n. Under suitable constraints on LWE parameters, Search-LWE can be reduced to Decision-LWE [17,16].

Search-LWE can be viewed as a BDD-instance in the m-dimensional lattice $\Lambda_q(A) = \{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y} = \mathbf{s}A \mod q \text{ for } \mathbf{s} \in \mathbb{Z}_q^n\}$, and with BDD error $\mathbf{e}$ (provided that α is sufficiently small). With overwhelming probability over A, $\mathrm{vol}(\Lambda_q(A)) = q^{m-n}$. This standard lattice attack can be improved using the so-called sublattice attack (see [5,11]), which replaces m by some $m' \leq m$ to optimize the use of current reduction algorithms. Given a root Hermite factor δ, we expect to achieve

$\|\mathbf{b}^*_{m'}\| \approx \delta^{-m'} q^{\frac{m'-n}{m'}}$, where $m' = \sqrt{n \log q / \log \delta}$ maximizes the norm of $\mathbf{b}^*_{m'}$. As long as $m' < m$, one applies a lattice attack to the sublattice with dimension m', otherwise one uses the full lattice.

Beta distribution. The density function of the Beta distribution of parameters $a, b > 0$ is $x^{a-1}(1-x)^{b-1}/B(a,b)$, where $B(a,b)$ is the beta function $\frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$. The corresponding cumulative distribution function is the regularized incomplete beta function $I_x(a,b) = \frac{1}{B(a,b)} \int_0^x u^{a-1}(1-u)^{b-1} du$, $x \in [0,1]$. If $(u_1, u_2, \ldots, u_m)$ is chosen uniformly at random from the unit sphere S^{m-1}, then $\sum_{i=1}^k u_i^2$ has distribution $Beta(k/2, (m-k)/2)$.

3 Lindner-Peikert's NearestPlanes Algorithm Revisited

In this section, we revisit the Lindner-Peikert NearestPlanes algorithm [10] (NP), which is a simple variant of Babai's algorithm [2] to solve BDD, and which turns out to be similar to Schnorr's random sampling [19]. We establish a connection with Gama-Nguyen-Regev's pruned enumeration [6], which allows us to randomize and generalize the NP algorithm.

3.1 Babai's Nearest Plane Algorithm

Since LWE $(\mathbf{A}, \mathbf{t} = \mathbf{sA} + \mathbf{e})$ is a BDD instance for the lattice $\Lambda_q(A)$, the simplest method is Babai's (deterministic polynomial-time) Nearest Plane algorithm, see Alg. 1.

Algorithm 1. Babai's Nearest Plane algorithm [2]

```
Input: A basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_m) \in \mathbb{Q}^m$ of a lattice $L$ and a target point $\mathbf{t} \in \mathbb{Q}^m$.
Output: $\mathbf{v} \in L$ such that $\mathbf{v} - \mathbf{t} \in \mathcal{P}_{1/2}(B^*)$.
1: $\mathbf{v} \leftarrow \mathbf{0}$
2: For $i \leftarrow m, ..., 1$
3:     Compute the integer $c$ closest to $\langle \mathbf{b}_i^*, \mathbf{t} \rangle / \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle$
4:     $\mathbf{t} \leftarrow \mathbf{t} - c\mathbf{b}_i$, $\mathbf{v} \leftarrow \mathbf{v} + c\mathbf{b}_i$
5: Return $\mathbf{v}$
```

Babai's algorithm outputs a lattice vector $\mathbf{v}$ relatively close to the input target vector $\mathbf{t}$. More precisely, $\mathbf{v}$ is the unique lattice vector such that $\mathbf{v} - \mathbf{t} \in \mathcal{P}_{1/2}(B^*)$. If the input is a BDD instance such that the closest lattice point to $\mathbf{t}$ is $\mathbf{v} \in L$, then Babai's algorithm solves BDD if and only if $\mathbf{v} - \mathbf{t} \in \mathcal{P}_{1/2}(B^*)$, *i.e.*

$$\forall i \in \{1, \ldots, m\} \ \ |\langle \mathbf{e}, \mathbf{b}_i^* \rangle| < \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle / 2. \quad (1)$$

In the LWE case, we can rigorously define a success probability, even though Babai's algorithm is deterministic: the probability of $\mathbf{v} - \mathbf{t} \in \mathcal{P}_{1/2}(B^*)$ is with respect to LWE parameter generation, *i.e.* the generation of $\mathbf{t}$. Unfortunately, one does not know how to compute this probability efficiently. Instead, Lindner and Peikert [10] compute the success probability in an idealized model (which we call CLWE for "continuous" LWE) where the LWE error distribution χ is replaced by a continuous Gaussian distribution with mean 0 and standard deviation $\alpha q/\sqrt{2\pi}$, and claim (without proof) that the actual probability is very close.

By definition of CLWE, the distribution of the error $\mathbf{e}$ is spherical, which implies because B^* has orthogonal rows:

$$\Pr[\mathbf{e} \in \mathcal{P}_{1/2}(B^*)] = \prod_{i=1}^{m} Pr[|\langle \mathbf{e}, \mathbf{b}_i^* \rangle| < \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle/2] = \prod_{i=1}^{m} \operatorname{erf}\left(\frac{\|\mathbf{b}_i^*\|\sqrt{\pi}}{2s}\right),$$

where $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-y^2}dy$. Typically, $\mathbf{b}_m^*$ is exponentially shorter than $\mathbf{b}_1$, which makes the probability of $\mathbf{e} \in \mathcal{P}_{1/2}(B^*)$ very small.

3.2 The NearestPlanes Algorithm

Lindner and Peikert [10] presented a simple generalization of Babai's nearest plane algorithm, by adding some exhaustive search to increase the success probability, at the expense of the running time. Instead of choosing the closest plane in every i-th level, the NearestPlanes algorithm (Alg. 2) enumerates d_i distinct planes.

Algorithm 2. NearestPlanes Algorithm [10]

```
Input: A lattice basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_m)$, a vector $\mathbf{d} = (d_1, d_2, \ldots, d_m) \in \mathbb{Z}^m$, a target
    point $\mathbf{t} \in \mathbb{Q}^m$.
Output: A set of $\prod_{i=1}^m d_i$ distinct lattice vectors in $\mathcal{L}(\mathbf{B})$ close to $\mathbf{t}$.
1: if $m = 0$ then
2:     Return $\mathbf{0}$
3: else
4:     Compute the $d_m$ integers $c_1, c_2, \ldots, c_{d_m} \in \mathbb{Z}$ closest to $\langle \mathbf{b_m^*}, \mathbf{t} \rangle / \langle \mathbf{b_m^*}, \mathbf{b_m^*} \rangle$
5:     Return $\bigcup_{i \in [d_m]} (c_i \mathbf{b}_m + NearestPlanes(\{\mathbf{b}_1, \ldots, \mathbf{b_{m-1}}, (d_1, \ldots, d_{m-1}), \mathbf{t} - c_i \mathbf{b}_m\})$
6: end if
```

Compared to Babai's nearest plane algorithm, the NearestPlanes algorithm is also deterministic, its running time is essentially multiplied by $\prod d_i$, and its CLWE success probability (*i.e.* under the assumption that the LWE distribution is continuous) increases to: $\prod_{i=1}^m \operatorname{erf}\left(\frac{d_i\|\mathbf{b}_i^*\|\sqrt{\pi}}{2s}\right)$. In fact, the algorithm succeeds if and only if $\mathbf{e} \in \mathcal{P}_{1/2}(\operatorname{diag}(\mathbf{d}) \cdot B^*)$, where $\operatorname{diag}(\mathbf{d})$ is the $m \times m$ diagonal matrix formed by the d_i's.

Lindner and Peikert [10] briefly compared their algorithm to the distinguisher of Micciancio and Regev [11]. Their data suggests that their algorithm is better for most parameters and success probability, with larger improvements in the high-advantage regime.

3.3 Connection with Schnorr's Random Sampling

We note that the NP algorithm is very similar to Schnorr's random sampling [19] from STACS '03. Schnorr's method aims at finding short vectors, but it can easily be adapted to BDD: in the BDD setting, there is an integer parameter $u \in \{1, \ldots, m\}$, and one computes all lattice vectors $\mathbf{v} \in L$ such that $\mathbf{v} - \mathbf{t} \in \mathcal{P}(\Delta B^*))$ where Δ is the $m \times m$ diagonal matrix formed by $m - u$ coefficients equal to 1, followed by u coefficients equal to 2. In other words, Schnorr's method corresponds to the particular case of NP where $(d_1, \ldots, d_m) = (1, 1, \ldots, 1, 2, 2, \ldots, 2)$, the number of 2's being exactly u. However, Schnorr's analysis is very different from [10] and uses a more debatable model: it assumes that the 2^u vectors $\mathbf{v} - \mathbf{t}$ are uniformly distributed over $\mathcal{P}(\Delta B^*))$, which cannot actually hold.

3.4 Connection with Lattice Enumeration

We note that both Babai's algorithm and the NP algorithm can be viewed as a pruned enumeration but with a different kind of pruning rule than Gama *et al.* [6]. Let us recall what is lattice enumeration. Given a target $\mathbf{t} \in \mathbb{Q}^m$, a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_m)$ of a lattice L and a BDD radius R, an enumeration algorithm enumerates all lattice vectors $\mathbf{v} \in L$ such that $\|\mathbf{v} - \mathbf{t}\| \leq R$, and selects the closest one to solve BDD. This is done by searching over a huge tree defined as follows: for each level $k \in \{0, \ldots, m\}$ where $k = 0$ corresponds to the root and $k = m$ corresponds to the leaves, the tree nodes of level k are all vectors $\mathbf{v} \in L$ such that $\|\pi_{m+1-k}(\mathbf{t} - \mathbf{v})\| \leq R$. This means that the number of nodes at level k is exactly the number of points in the projected lattice $\pi_{m+1-k}(L)$ which are within distance R of $\pi_{m+1-k}(\mathbf{t})$, which can be heuristically estimated as $H_k = V_k(R)/\mathrm{vol}(\pi_{m+1-k}(L))$ by the Gaussian Heuristic: such estimates seem to be fairly accurate in practice [6].

Pruned enumeration was first proposed by Schnorr and Euchner [21] to cut down some branches in the enumeration tree to decrease the time complexity, at the cost of potentially missing the solution vector. It was hoped that the overall cost (taking into account failure probability) would decrease. An algorithmic description of pruned enumeration for BDD is given in Appendix A. The first rigorous analysis of pruned enumeration was only recently given by Gama, Nguyen and Regev [6], where a framework generalizing [21] was proposed. For every tree level k, they used a variable enumeration radius $\|\pi_{m+1-k}(\mathbf{v} - \mathbf{t})\| \leq R_k$ such that $R_1 \leq R_2 \leq \cdots \leq R_m$. This means that the pruned tree is a subset of the enumeration tree.

Babai's algorithm and the NP algorithm also only consider a subset of the enumeration tree, but it is a different subset than [6]. More precisely, Babai's algorithm looks at a single branch of the enumeration tree with radius $R = \frac{1}{2}\sqrt{\sum_{i=1}^{m} \|\mathbf{b}_i^*\|^2}$. Since the NP algorithm enumerates all lattice points inside the (orthogonal) parallelepiped $\mathcal{P}_{1/2}(\mathrm{diag}(\mathbf{d}) \cdot B^*)$ centered at $\mathbf{t}$, it actually considers the radius $R = \frac{1}{2}\sqrt{\sum_{i=1}^{m} d_i^2 \|\mathbf{b}_i^*\|^2}$. Then, among all the nodes $\mathbf{v} \in L$ of the enumeration tree at level k, it only considers those such that $|\xi_i(\mathbf{v}-\mathbf{t})| \leq d_i\|\mathbf{b}_i^*\|/2$ for all $i \geq m+1-k$, where $\xi_i(\mathbf{x})$ denotes the i-th coordinate of $\mathbf{x}$ in the normalized Gram-Schmidt basis $(\mathbf{b}_1^*/\|\mathbf{b}_1^*\|, \cdots, \mathbf{b}_m^*/\|\mathbf{b}_m^*\|)$, *i.e.* $\xi_i(\mathbf{x}) = \langle \mathbf{x}, \mathbf{b}_i^* \rangle / \|\mathbf{b}_i^*\|$.

In other words, GNR pruning [6] only keeps the nodes with bounded projections $\|\pi_{m+1-k}(\mathbf{v}-\mathbf{t})\| \leq R_k$, whereas NP [10] only keeps the nodes with bounded coordinates $|\xi_i(\mathbf{v}-\mathbf{t})|$. In some sense, NP can be viewed as a secondary pruning of GNR: if the coordinates are all bounded, then so are the projections, which means that the NP tree is a subset of some GNR tree.

3.5 Randomizing the NearestPlanes Algorithm

This connection of NP with enumeration allows us to revisit and improve NP. First, NP can be generalized by selecting arbitrary bounds on coordinates, namely $|\xi_i(\mathbf{v}-\mathbf{t})| \leq R_i$ instead of $|\xi_i(\mathbf{v}-\mathbf{t})| \leq d_i\|\mathbf{b}_i^*\|/2$, where the bounds $R_1, \ldots, R_m$ are parameters which are not necessarily multiples of the $\|\mathbf{b}_i^*\|/2$'s.

Second, we perform a randomization similar to the one of extreme pruning [6] compared to basic pruning. More precisely, we randomize the algorithm by repeating several times the basic algorithm with different randomized reduced bases. If we use ℓ different bases, then the running time is roughly multiplied by ℓ (depending on the exact reduction time compared to the NP exhaustive search), but the success probability is also heuristically multiplied by ℓ. This gives rise to Alg. 3, and allows more optimization. In particular, the numerical data from [10, Table 3] is far from optimal, since the running times of basis reduction and enumeration are not totally balanced. When the enumeration time is longer than the reduction time, we can decrease the total cost by decreasing the number of enumerations. We can obtain better trade-offs because the randomized algorithm has more freedom than the original one.

This can be proved by the simple analysis below. Since $\mathrm{erf}(a) < \frac{\mathrm{erf}(c \cdot a)}{c} (c < 1)$, and we use cd_i instead of d_i, the algorithm is $1/c$ times faster than the original one, while the success probability is higher than a c-fraction of the original one. Thus, by choosing different bases to repeat the algorithm several times, the total cost will actually decrease. In [10], no information was given on how to choose the d_i's. We performed an exhaustive search in the proper range to find an optimal $(d_1, d_2, \ldots, d_m)$ for the randomized algorithm, which yielded a much more efficient attack.

Algorithm 3. Our Randomized NearestPlanes Algorithm

```
Input: A lattice basis $B$, a vector $\mathbf{d} = (d_1, d_2, \ldots, d_m)$, a target point $\mathbf{t} \in \mathbb{Q}^m$, and a
    number $\ell$ of iterations.
Output: A set of candidate lattice vectors in $\mathcal{L}(\mathbf{B})$ close to $\mathbf{t}$.
1: Repeat $\ell$ times
2: Randomize the input basis, and apply lattice reduction to obtain a (random) re-
   duced basis $B'$
3: Run the NearestPlanes algorithm on $(B', \mathbf{d}, \mathbf{t})$
```

A numerical comparison between the NP algorithm [10] and our randomized variant is given in Table 1 below. This shows that randomization can provide significant speedups, as high as 2^{32} *e.g.* $n = 320$. Basis reduction time estimates are taken from [10], where it is estimated that $lg(T_{recd}) = \frac{1.8}{\lg(\delta)} - 110$. Columns marked with "NearestPlanes" are from [10], while columns with "Randomized-NP" correspond to our randomized variant. When we search for the best root Hermite constant, we increment by 0.0001 each time as in [10]. Because the estimates of [10] for lattice reduction are debatable, we use the estimates of Chen and Nguyen [4] to update the cost estimates of the attack in Table 2.

Table 1. NP vs. randomized NP. δ is the root Hermite factor. `Adv` is the target success probability. `log(Nb of bases)` is the number of bases needed to reach the target success probability in base-2 logarithm. `Red` and `Cost` indicate the lattice reduction time (using [10]) and total time in base-2 logarithm seconds respectively. `Enum` and `Pr` are respectively the number $\prod_{i=1}^{n} d_i$ of enumerations and success probability, in base-2 logarithm.

			Adv	NearestPlanes [10]				Randomized-NP						Log
n	q	s	(log)	δ	Red [10]	Enum	Cost	δ	Red [10]	Enum	Pr	log(Nb of bases)	Cost	speedup
			0	1.0089	30.8	47	32	1.0104	10.6	27	-9.6	9.6	21.2	10.8
128	2053	6.7	-32	1.0116	< 0	13	< 0	1.0114	< 0	17	-25.5	0	< 0	
			-64	1.013	< 0	1	< 0							
			0	1.0067	76.8	87	78	1.0077	52.6	68.9	-12	12	65.6	12.4
192	4093	8.9	-32	1.0083	40.9	54	42							
			-64	1.0091	27.7	44	29							
			0	1.0052	130.5	131	132	1.006	98.6	115	-11.8	11.8	111.3	20.7
256	4093	8.3	-32	1.0063	88.7	87	90	1.0065	82.6	99	-34	2	85.6	4.4
			-64	1.0068	74.1	73	75	1.007	68.9	85	-67	3	72.9	2.1
			0	1.0042	187.7	163	189	1.005	140.2	156.9	-15.7	15.7	156.9	32.1
320	4093	8	-32	1.0052	130.6	138	132	1.0053	126	143	-34.7	2.7	129.7	2.3
			-64	1.0055	117.5	117	119	1.0056	113.4	127	-61.6	0	114.4	4.6

4 Solving BDD by (GNR) Pruned Enumeration

In this section, we study how BDD can be solved in practice using the pruned enumeration algorithm of Gama, Nguyen and Regev [6] (GNR). Timings given are for a single 3-Ghz Intel-Core2 core. First, we consider a theoretical application to LWE, to compare GNR pruning with our randomized NP algorithm. Then, we report improved experimental results on two well-known cryptanalytical applications of BDD: attacks on DSA with partially known nonces [15] and attacks on the GGH encryption challenges [13]. In these settings, the best method known in practice was to use the heuristic embedding method that transforms BDD into a Unique-SVP instance. Our experiments show that the embedding method is now outperformed by pruned enumeration, even without taking into account past improvements in lattice reduction algorithms [4]; and we did not find any application where Randomized-NP was better than pruned enumeration.

4.1 Further Background on GNR Pruned Enumeration

We recall additional information on GNR pruned enumeration, to complete the description of Sect. 3.4. Given a target $\mathbf{t} \in \mathbb{Q}^m$, a basis $B = (\mathbf{b}_1, \ldots, \mathbf{b}_m)$ of a lattice L and a BDD radius R, GNR aims at finding the supposedly unique lattice vector $\mathbf{u} \in L$ such that $\|\mathbf{u} - \mathbf{t}\| \leq R$. To this aim, GNR [6] selects a bounding function determined by radius $R_1 \leq R_2 \leq \cdots \leq R_m = R$, and performs a depth-first search of the pruned-enumeration tree defined as follows: for each level $k \in \{0, \ldots, m\}$ where $k = 0$ corresponds to the root and $k = m$ corresponds to the leaves, the tree nodes of level k are all vectors $\mathbf{u} \in L$ such that $\|\pi_{m+1-i}(\mathbf{t} - \mathbf{u})\| \leq R_i$ for all $0 \leq i \leq k$. By assumption, there is at most a single leaf in the tree.

The complexity of the pruned enumeration is equal to the total number of nodes in the pruned-enumeration tree, up to some polynomial-time factor. To estimate this number of nodes, GNR [6] introduced the k-dimensional cylinder intersection defined by $R_1 \leq R_2 \leq \cdots \leq R_m$ as follows:

$$C_{R_1,R_2,\ldots,R_k} = \{(x_1, x_2, ..., x_k) \in \mathbb{R}^k, \forall j \leq k, \sum_{l=1}^{j} x_l^2 \leq R_j^2\}.$$

GNR apply the Gaussian heuristic to (heuristically) estimate the number of nodes at depth k by $H_k = \frac{\mathrm{vol}(C_{R_1,R_2,\ldots,R_k})}{\prod_{i=m+1-k}^{m} \|\mathbf{b}_i^*\|}$. GNR [6] provide efficient algorithms to estimate $\mathrm{vol}(C_{R_1,R_2,\ldots,R_k})$, and therefore H_k. This can be used to find good bounding functions by numerical optimization. In practice, one considers either the *linear bounding function* defined by $R_k = \sqrt{k/m}R_m$, or numerical functions found by numerical optimization.

GNR [6] rigorously defined a success probability, by assuming that the input basis is a random reduced basis. This probability p_{succ} is the probability that the closest lattice vector $\mathbf{u}$ to $\mathbf{t}$ remains in the enumeration tree. They explained how to compute this probability in practice, if one assumes that the BDD error vector $\mathbf{t} - \mathbf{u}$ is a vector chosen uniformly at random in the sphere (or ball) of radius R. And they also give theoretical estimates for certain bounding functions: for instance, $p_{succ} = 1/m$ for linear bounding. The total cost of pruned-enumeration is roughly equal to:

$$\frac{T_{redu} + T_{node} \cdot N(R_1, \ldots, R_m, \mathbf{b}_1^*, \ldots, \mathbf{b}_m^*)}{p_{succ}(R_1, \ldots, R_m)},$$

where T_{redu} is the basis reduction time, T_{node} is the time for enumerating one node, and $N(R_1, \ldots, R_m, \mathbf{b}_1^*, \ldots, \mathbf{b}_m^*) = \sum_{k=1}^{m} H_k$ is the (approximate) number of nodes.

Though GNR [6] presented the first rigorous framework to analyzed pruned-enumeration algorithms, it must be stressed that the GNR analysis only holds for certain settings. For instance, when one wants to solve BDD in practice, it is often the case that we do not know the exact value of $\|\mathbf{t} - \mathbf{u}\|$, which makes a theoretical analysis difficult. In this case, we may take for R the expected value of $\|\mathbf{t}-\mathbf{u}\|$, or an upper bound satisfied with high probability; then pretending that $\mathbf{t}-\mathbf{u}$ is a vector chosen uniformly at random in the sphere (or ball) of radius R will only provide a rough estimate of p_{succ}. However, the distribution of $\mathbf{t}-\mathbf{u}$ is often known, which allows to compute experimentally (by sampling) the probability that $\mathbf{t}-\mathbf{u}$ satisfies a given bounding function, provided that the probability is not too small. This is the method we used in our experiments, and this is what we mean by success probability.

4.2 Application to LWE

In this subsection, we provide numerical evidence that pruned enumeration is better than NP for solving LWE. As mentioned previously, the analysis of [6] must be adapted, because the LWE noise distribution does not fit the GNR model: in particular, the exact norm of the noise is not known.

If we assume the CLWE model, the error vector $\mathbf{e}$ has continuous Gaussian distribution, then all the coordinates of $\mathbf{e}$ with respect to the normalized Gram-Schmidt basis $(\mathbf{b}_m^*/\|\mathbf{b}_m^*\|, \cdots, \mathbf{b}_1^*/\|\mathbf{b}_1^*\|)$, have Gaussian distribution. It follows that the success probability is:

$$p_{succ} = p_{succ}(R_1, \ldots, R_m) = Pr_{\mathbf{e}\sim\chi}(\forall j \in [1,m], \sum_{i=1}^{j} u_i^2 \le R_j^2),$$

where χ is the (continuous) noise distribution of CLWE. If p_{succ} is high, we can use Monte Carlo sampling to compute it numerically. But for extreme pruning, where probabilities are voluntarily chosen very small, this is impractical. However, we notice that $\mathbf{e}/\|\mathbf{e}\|$ is uniformly distributed in the unit sphere, by definition. Hence, if we use a bound R_m such that $Pr(\|\mathbf{e}\| > R_m)$ is negligible, the probability $Pr_{\mathbf{u}\sim S^{m-1}}(\forall j \in [1,m], \sum_{i=1}^{j} u_i^2 \le \frac{R_j^2}{R_m^2})$ can be used as a lower bound of the actual success probability.

Table 2 provides numerical comparisons between Randomized-NP and linear pruning (where the success probability is lower-bounded, with optimized enumeration radius around the expected length of error). As opposed to Table 1,

Table 2. Randomized-NP vs. Linear Pruning. δ is the root Hermite factor. `Red` indicates approximate lattice reduction times from Chen-Nguyen [4] in base-2 logarithm (in seconds). The rest of the data is organized as in Table 1, except that the enumeration radius chosen for linear pruning is the expected error length times he square root of `Radius factor`.

						Randomized-NP				Linear Pruning				Log
n	q	s	δ	Red [4] (log)	Adv (log)	Enum (log) (log)	Pr (log)	Nb. of bases	Cost	Enum	Pr (log)	Radius factor	Cost	Speedup
128	2053	6.7	1.01	18.4	0	35.3	−7.1	7.1	26.6	35.4	−3.9	1.01	23.6	3
			1.012	8.2	−32	25.1	−34.5	2.5	11.7					
			1.013	< 0	−64	10.10	−63.1	0	< 0					
192	4093	8.9	1.007	61.8	0	78.7	−3.7	3.7	66.5	74.1	−0.9	1.5	62.8	3.7
			1.008	42	−32	48.6	−32	0	42					
			1.009	28	−64	44.9	−66.1	2.1	31					
256	4093	8.3	1.006	95.3	0	112.2	−15.1	15.1	111.4	113	−8.5	0.9	105.5	5.8
			1.006	95.3	−32	71.9	−31.6	0	95.3					
			1.007	62.2	−64	79.2	−77.7	13.7	76.9					

Table 2 uses Chen-Nguyen estimates [4] for lattice reduction times. Independently of reduction time estimates, linear pruning is better in practice than NearestPlanes (even randomized).

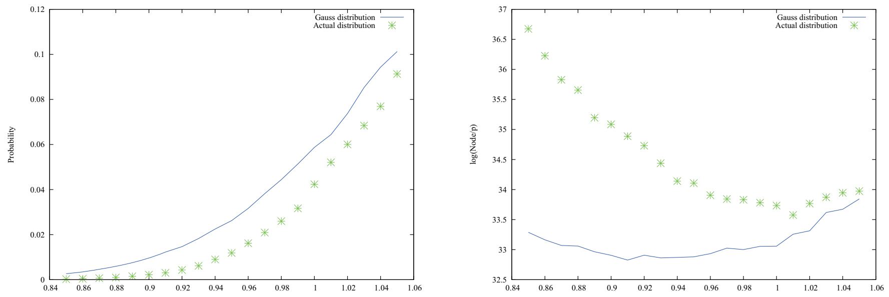
4.3 Application to GGH

In this subsection, we apply pruned-enumeration on the encryption challenges for the Goldreich-Goldwasser-Halevi cryptosystem [9] (GGH). The lattice dimension is either 200, 250, 300, 350 and 400. There are two types of challenges: key-recovery and message-recovery. Key-recovery can be viewed as solving m BDD-instances with error vector chosen uniformly at random from $[-4,\dots,+3]^m$. To the best of our knowledge, none of these key-recovery challenges was ever solved. Each message-recovery challenge is a BDD instance where the error vector is chosen uniformly at random from $\{-3,+3\}^m$: Nguyen [13] solved the message-recovery challenges in dimensions 200,250,300 and 350 by showing how to reduce each such BDD instance to a small number of BDD instances with error in $\{-1/2,+1/2\}^m$, and solving these easier BDD instances using the embedding strategy and lattice reduction: BKZ-20 was enough for dimensions 200-300, but dimension 350 required pruned-BKZ with higher blocksize 60.

Table 3. Key-recovery for GGH Challenges

Dimension	200	250	300	300
BKZ blocksize	60	60	90	90
Bounding function	linear	linear	linear	optimized
Estimated Nb of Nodes	800	5.84×10^8	7.58×10^{10}	1.33×10^9
Average Nb of Nodes	666	5.93×10^8	-	1.35×10^9
Success probability	0.0418	0.0409	0.0371	0.0185
Nb of Success	4	11	-	2

First, we re-solved the GGH-350 message-recovery challenge by using only a BKZ-20 reduced basis and high-probability pruning, which shows that pruning is better than the well-known embedding method. In this case, the BDD radius is exactly $R_m = \frac{\sqrt{m}}{2}$, after applying Nguyen's trick [13], and the BDD factor γ is ≈ 0.125. We used a high-probability bounding function defined as follows: $R_k^2 = \min\{E(X_k)+3\sigma(X_k),1\}R_m^2$ where E and σ denote respectively the expectation and standard deviation of the distribution $Beta(k/2,(m-k)/2)$, namely $\frac{k}{m}$ and $\sqrt{\frac{k(m-k)}{m^2(\frac{m}{2}+1)}}$. The pruned tree contained 1.76×10^{11} nodes, very close to the Gaussian heuristic estimate 1.73×10^{11}. The error behaved as if it was uniformly distributed in the sphere of radius R_m: indeed, in both our experiments and in the uniform model, the success probability was ≈ 0.92. Note that the largest GNR experiments [6] used a lattice dimension of 110. Because enumeration (pruned or not) has a super-exponential running time, one may have thought that much larger dimensions would be unreachable. However, our experiments show that



(a) Success probability for GGH-250 key-recovery

(b) Total cost for GGH-250 key-recovery

Fig. 1. Here, the x-coordinate is the ratio between the (squared) enumeration radius with the (squared) expected length of the error vector. In (b), the y-coordinate is $\log_2$(Number of enumeration nodes/Success probability).

dimensions as high as 350 are reachable, provided that the BDD enumeration radius is sufficiently small.

Next, we recovered several secret-key vectors in dimensions 200, 250 and 300, using pruned-enumeration and BKZ 2.0 [4] as the reduction algorithm: the experiments are summarized in Table 3, where the Gaussian heuristic is used to estimate the number of nodes.

In these BDD instances, the exact BDD radius is unknown, but the factor γ is approximately $\frac{\sqrt{\frac{11}{2}m}}{4\sqrt{m}} \approx 0.59$, which is more difficult than for the message-recovery challenges. For each dimension, we computed only one BKZ-reduced basis, and tried to recover as many secret-key vectors with the same basis: because the success probability is much lower than 1 for one pruned enumeration, we actually only recovered a fraction of all secret-key vectors, but of course, our experiments show that one could recover all secret-key vectors simply by repeating our experiments a small number of times. For dimension 300, linear bounding was not sufficient, so we tried another bounding function by optimization, using the GNR method [6]: we start with the linear bounding function, then randomly modify it by small perturbation successively. Using this bounding function, one pruned-enumeration only takes several minutes. In these experiments, the enumeration radius was chosen as the expected error length. However, there is clearly a trade-off if one wants to optimize the total running time: by selecting a smaller radius, one can decrease the running time of a single enumeration, at the expense of the success probability. Fig 1 shows the impact of varying the enumeration radius around the expected length, on the success probability and the total cost: here, one obtains slightly better results by increasing the radius. Fig 1 also compares experimental probabilities and costs with that of a Gaussian modelization where the error vector is chosen with Gaussian distribution of expectation $\sqrt{\frac{11}{2}m}$: we see that it is better to compute experimental probabilities (by sampling the error distribution).

We also tried to solve GGH challenges using Randomized-NP but the performances were worse than pruned enumeration.

4.4 Application to DSA

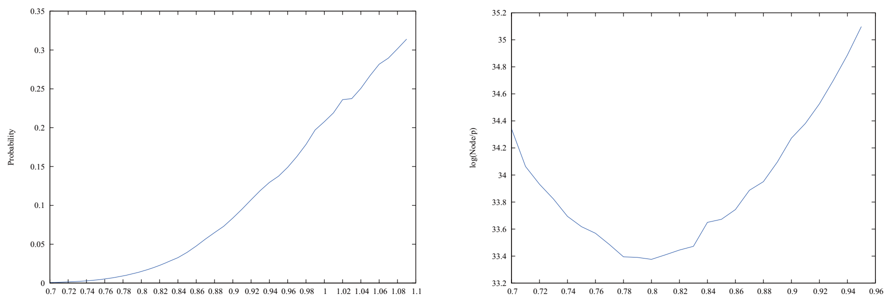
In this subsection, we apply pruned-enumeration to attack the Digital Signature Algorithm [12] (DSA) with partially known nonces. Each DSA signature generation require the use of a one-time key k modulo q, where q is usually a 160-bit prime number. It is well-known (and obvious) that disclosing the full one-time key k of a single (message,signature) pair allows to recover the DSA secret key in polynomial time. It is also well-known (but not obvious) that disclosing ℓ bits of each one-time key k for several (message,signature) pair allows to recover the DSA secret key, see *e.g.* [15]. More precisely, this cryptanalytical problem can be reduced to the so-called hidden number problem (HNP), which can itself be reduced to BDD. For any real z, let the symbol $|\cdot|_q$ be $|z|_q = \min_{b\in\mathbb{Z}} |z - bq|$. $APP_{\ell,q}(n)$ denotes any rational number r satisfying $|n - r|_q \le \frac{q}{2^{\ell+1}}$. The HNP asks to recover $\alpha \in \mathbb{Z}_q$, given many approximations $u_i = APP_{\ell,q}(\alpha t_i)$ where each t_i is known and chosen uniformly at random, for $1 \le i \le d$. The reduction to BDD works as follows. One constructs the $(d+1)$-dimensional lattice spanned by the following row matrix:

$$\begin{pmatrix} q & 0 & \cdots & 0 & 0 \\ 0 & q & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & q & 0 \\ t_1 & \cdots & \cdots & t_d & \frac{1}{2^{l+1}} \end{pmatrix} \tag{2}$$

The target vector is $\mathbf{u} = (u_1, u_2, \ldots, u_d, 0)$. There exists a lattice vector $\mathbf{h} = (\alpha t_1 + qh_1, \ldots, \alpha t_d + qh_d, \frac{\alpha}{2^{\ell+1}})$, such that $\| \mathbf{h} - \mathbf{u} \| \le \sqrt{d+1}\frac{q}{2^{\ell+1}}$. And finding $\mathbf{h}$ discloses α.

Nguyen and Shparlinski [15] used this attack to recover the DSA secret key in a few hours, given the $\ell = 3$ least significant bits of each one-time key for about 100 signatures, but the attack failed for $\ell = 2$. By using BKZ-90 reduction [4] and linear pruning, we were able to attack the $\ell = 2$ case given about 100 signatures, within a few hours, but the lattice needs to be slightly changed: indeed, for the GNR analysis to hold, one needs that the error vector looks like a random vector in the Gram-Schmidt basis; because the shape of reduced bases is special for these HNP lattices, one needs to modify the right-bottom coefficient of the row matrix by some scaling factor.

We constructed 100 instances to check the cost and success probability. The average number of nodes was 1.37×10^{10}, slightly smaller than the estimated number of nodes 1.5×10^{10}. The average actual running time is about 4185 seconds per enumeration. Our experiments solved 23 out of 100 instances, which means that the running time of a single enumeration needs to be multiplied by roughly 4, which is a few hours at most. Because the exact length of the



(a) Success probability for DSA **(b)** Total cost for DSA

Fig. 2. Here, the x-coordinate is the ratio between the (squared) enumeration radius and the (squared) expected length the error vector. In (b), the y-coordinate is $\log_2$(Number of enumeration nodes/Success probability)

error vector is not known, like in the GGH case, there is a trade-off for choosing the enumeration radius: Fig. 2 shows the impact when varying the enumeration radius around the expected length, like Fig. 1. Here, the optimal enumeration radius is about $\sqrt{0.8} \approx 0.89$ smaller than the expected length.

Acknowledgements. Part of this work is supported by the Commission of the European Communities through the ICT program under contract ICT-2007-216676 ECRYPT II, the European Research Council, and by China's 973 Program (Grants 2013CB834201 and 2013CB834205).

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: STOC, pp. 99–108 (1996)
2. Babai, L.: On Lovász' Lattice Reduction and the Nearest Lattice Point Problem (Shortened Version). In: Mehlhorn, K. (ed.) STACS 1985. LNCS, vol. 182, pp. 13–20. Springer, Heidelberg (1984)
3. Brakerski, Z., Vaikuntanathan, V.: Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
4. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better Lattice Security Estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011)
5. Gama, N., Nguyen, P.Q.: Predicting Lattice Reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)
6. Gama, N., Nguyen, P.Q., Regev, O.: Lattice Enumeration Using Extreme Pruning. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 257–278. Springer, Heidelberg (2010)
7. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proc. STOC 2009, pp. 169–178. ACM (2009)

8. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proc.STOC 2008, pp. 197–206. ACM (2008)
9. Goldreich, O., Goldwasser, S., Halevi, S.: Public-Key Cryptosystems from Lattice Reduction Problems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (1997)
10. Lindner, R., Peikert, C.: Better Key Sizes (and Attacks) for LWE-Based Encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
11. Regev, O.: Lattice-Based Cryptography. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 131–141. Springer, Heidelberg (2006)
12. National Institute of Standards and Technology (NIST). Fips publication 186:digital signature standard (1994)
13. Nguyên, P.Q.: Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto'97. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 288–304. Springer, Heidelberg (1999)
14. Nguyen, P.Q.: Public-key cryptanalysis. In: Luengo, I. (ed.) Recent Trends in Cryptography. Contemporary Mathematics, vol. 477, AMS–RSME (2009)
15. Nguyen, P.Q., Shparlinski, I.: The insecurity of the digital signature algorithm with partially known nonces. J. Cryptology 15(3), 151–176 (2002)
16. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Proc. STOC 2009, pp. 333–342. ACM (2009)
17. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proc. STOC 2005, pp. 84–93. ACM (2005)
18. Regev, O.: The learning with errors problem (invited survey). In: Proc. IEEE Conference on Computational Complexity, pp. 191–204 (2010)
19. Schnorr, C.-P.: Lattice Reduction by Random Sampling and Birthday Methods. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 145–156. Springer, Heidelberg (2003)
20. Schnorr, C.-P.: Lattice Reduction by Random Sampling and Birthday Methods. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 145–156. Springer, Heidelberg (2003)
21. Schnorr, C.-P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Programming 66, 181–199 (1994)

A Description of Pruned Enumeration for BDD

Algorithm 4. Pruned Enumeration for BDD (BDD version of [6])

```
Input: A basis $B = (\mathbf{b}_1, \dots, \mathbf{b}_m)$, a target vector $\mathbf{t} = \sum_{i=1}^{m} t_i \mathbf{b}_i$, a bounding function $R_1^2 \le \cdots \le R_m^2$, the Gram-Schmidt matrix $\mu$ and the (squared) norms $\|\mathbf{b}_1^*\|^2, \dots, \|\mathbf{b}_m^*\|^2$.
Output: Nothing or the coefficients of a lattice vector $\mathbf{v}$ such that the projections of $\mathbf{v} - \mathbf{t}$ have norms less than the $R_i$'s, i.e. $\|\pi_{m+1-k}(\mathbf{v} - \mathbf{t})\| \le R_k$ for all $1 \le k \le m$.
1: $\sigma \leftarrow (0)_{(m+1)\times m}$; $r_0 = 0; r_1 = 1; \cdots; r_m = m; \rho_{m+1} = 0$
2: for $k = m$ downto 1
3:     for $i = m$ downto $k+1$ do $\sigma_{i,k} \leftarrow \sigma_{i+1,k} + (t_i - v_i)\mu_{i,k}$ endfor
4:     $c_k \leftarrow t_k + \sigma_{k+1,k}$ // $c_k \leftarrow t_k + \sum_{i=k+1}^{m}(t_i - v_i)\mu_{i,k}$, centers
5:     $v_k \leftarrow \lfloor c_k \rceil$ // current combination;
6:     $w_k = 1$ // jumps;
7:     $\rho_k = \rho_{k+1} + (c_k - v_k)^2 \cdot \|\mathbf{b}_k^*\|^2$
8: endfor
9: $k = 1$;
10: while true do
11:     $\rho_k = \rho_{k+1} + (c_k - v_k)^2 \cdot \|\mathbf{b}_k^*\|^2$ // compute norm squared of current node
12:     if $\rho_k \le R_{m+1-k}^2$ (we are below the bound) then
13:         if $k = 1$ then
14:             return $(v_1, \dots, v_m)$; (solution found; program ends)
15:         else
16:             $k \leftarrow k - 1$ // going down the tree
17:             $r_{k-1} \leftarrow \max(r_{k-1}, r_k)$ // to maintain the invariant for $j < k$
18:             for $i = r_k$ downto $k+1$ do $\sigma_{i,k} \leftarrow \sigma_{i+1,k} + (t_i - v_i)\mu_{i,k}$ endfor
19:             $c_k \leftarrow t_k + \sigma_{k+1,k}$ // $c_k \leftarrow t_k + \sum_{i=k+1}^{m}(t_i - v_i)\mu_{i,k}$
20:             $v_k \leftarrow \lfloor c_k \rceil$; $w_k = 1$
21:         end if
22:     else
23:         $k \leftarrow k + 1$ // going up the tree
24:         if $k = m + 1$ then
25:             return $\emptyset$ (there is no solution)
26:         end if
27:         $r_{k-1} \leftarrow k$ // since $v_k$ is about to change, indicate that $(i, j)$ for $j < k$ and $i \le k$ are not synchronized
28:         // update $v_k$
29:         if $v_k > c_k$ then $v_k \leftarrow v_k - w_k$ else $v_k \leftarrow v_k + w_k$
30:         $w_k \leftarrow w_k + 1$
31:     end if
32: end while
```

The *k*-BDH Assumption Family: Bilinear Map Cryptography from Progressively Weaker Assumptions

Karyn Benson[1], Hovav Shacham[1,⋆], and Brent Waters[2,⋆⋆]

[1] University of California, San Diego
{kbenson,hovav}@cs.ucsd.edu
[2] University of Texas at Austin
bwaters@cs.utexas.edu

Abstract. Over the past decade bilinear maps have been used to build a large variety of cryptosystems. In addition to new functionality, we have concurrently seen the emergence of many strong assumptions. In this work, we explore how to build bilinear map cryptosystems under progressively weaker assumptions.

We propose k-BDH, a new family of progressively weaker assumptions that generalizes the decisional bilinear Diffie-Hellman (DBDH) assumption. We give evidence in the generic group model that each assumption in our family is strictly weaker than the assumptions before it. DBDH has been used for proving many schemes secure, notably identity-based and functional encryption schemes; we expect that our k-BDH will lead to generalizations of many such schemes.

To illustrate the usefulness of our k-BDH family, we construct a family of selectively secure Identity-Based Encryption (IBE) systems based on it. Our system can be viewed as a generalization of the Boneh-Boyen IBE, however, the construction and proof require new ideas to fit the family. Our methods can be extended to produce hierarchical IBEs and CCA security; and give a fully secure variant. In addition, we discuss the opportunities and challenges of building new systems under our weaker assumption family.

1 Introduction

Since the introduction of the Boneh-Franklin [1] Identity-Based Encryption (IBE) system a decade ago, we have seen an explosion of new cryptosystems based on bilinear maps. These systems have provided a wide range of functionality including: new signature systems, functional encryption, e-cash, "slightly" homomorphic encryption, broadcast encryption and oblivious transfer to name just

⋆ Supported by the MURI program under AFOSR Grant No. FA9550-08-1-0352.
⋆⋆ Supported by NSF CNS-0915361 and CNS-0952692, AFOSR Grant No: FA9550-08-1-0352, DARPA PROCEED, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, and Microsoft Faculty Fellowship.

a few. The focus of many of this work was to develop new (and often not realized before) functionality. While Boneh-Franklin and many first IBE systems used "core" assumptions such as the Bilinear Diffie-Hellman or decisional variants, over time there has been a trend in bilinear map based work to employ stronger assumptions in order to obtain these functionalities. Examples of these assumptions range from "q-type" [2] assumptions, assumptions in composite order groups [3], interactive assumptions [4] and proofs that appealed directly on the generic group model [5,6]

Interestingly, even some work that focused on tightening security (versus achieving new functionality) have had to employ relatively strong assumptions. For example, Gentry and Halevi [7] and Waters [8] proposed two different approaches for solving the problem of achieving adaptive security for Hierarchical Identity-Based encryption. To achieve this the former used a q-type assumption where the strength of the assumption depends on the number of attacker private key queries. The latter used the decisional-Linear assumption, where the target of the assumption is in the source element of the bilinear group versus the target element. Both of these assumptions are potentially stronger than the classic decisional-BDH prior IBE and related systems were built upon.

Our Goals. In this work, we move in the opposite direction of this trend. We will build bilinear map systems that depend on *weaker* assumptions than the decisional-BDH assumption. In particular, we want to create a suitable family of assumptions that becomes progressively weaker as some parameter k is increased. Therefore one can increase k as a hedge against potential future attacks such as an n-linear map for $n > 2$.

A natural starting point for our investigation is the k-Linear family of assumptions [9,10], which generalizes the decisional Diffie-Hellman assumption (DDH) and the decisional Linear assumption of Boneh, Boyen, and Shacham [11]. For $k \geq 1$, a k-Linear problem instance is a tuple $(g, g_1, \ldots, g_k, g_1^{r_1}, \ldots, g_k^{r_k}, T)$, where the generators are random in the group $\mathbb{G}$, the exponents in its support $\mathbb{Z}_p$, and the goal is to determine whether T is equal to $g^{r_1+\cdots+r_k}$ or random. DDH is 1-Linear, and the Linear assumption is 2-Linear.

The k-Linear assumption family has been successfully used to build chosen ciphertext secure encryption [9,10]; to construct pseudorandom functions [12,13]; to construct public-key encryption secure in the presence of encryption cycles [14,15] and public-key encryption resilient to key leakage [16,17]; to construct lossy trapdoor functions [18]; to construct leakage-resilient signatures [19].

While the k-Linear family has been successful in the above contexts, we desire an assumption that can be used in bilinear map cryptosystems in place of where DBDH has typically been applied. Here using the k-Linear family does not appear well suited for two reasons. First, since the assumption of the family operates solely in the source group, the assumption is not even "aware" of bilinear groups. Therefore it is not clear how it might be applied in certain systems (e.g. an variant of Boneh-Boyen IBE) where we are hiding a message in the target group. Second, the Linear assumption family has an inconsistent interaction with the DBDH assumption: the $1, 2$-Linear assumptions are actually stronger than DBDH, but the the k-Linear assumptions for $k > 2$ are generically incomparable to DBDH. One reason that the (2-)Linear assumption has proved so useful is that it gives DBDH "for free," but this is lost as soon as one increases k beyond 2. If a new IBE system were based on k-Linear *and* DBDH, it is not clear that this would provide an improvement in security.[1]

Our goals, then, are to find an assumption family that meets the following criteria:

– As we increase the assumption family parameter k, we should become more confident in the security of our assumption. In particular, we would argue that our k parameterized assumption is in some sense more secure than both existing decisional assumptions in bilinear groups and more secure than the $k - 1$ instance.
– Our family of assumptions should be amenable to building cryptographic systems. Ideally, for any system built using the DBDH assumption, one could find a variant built using our family.

The k-BDH Family of Assumptions. Our main contribution is a new family of assumptions that can serve as a weaker generalization of DBDH.

We propose a family of progressively weaker assumptions, the k-BDH assumptions, that generalizes the DBDH assumption. The 1-BDH assumption is equivalent to DBDH. More generally, the k-BDH assumption is as follows:

$$\text{given } g,\, g^x, g^y,\, v_1, \ldots, v_k,\, v_1^{r_1}, \ldots, v_k^{r_k} \text{ in } \mathbb{G},$$
$$\text{decide whether } T = e(g,g)^{(xy)(r_1+\cdots+r_k)} \text{ or random in } \mathbb{G}_T.$$

Here g and $\{v_i\}$ are random generators of $\mathbb{G}$ and x, y, and $\{r_i\}$ are random elements of its support $\mathbb{Z}_p$. We consider only the decisional versions of these problems; as with k-Linear, the computational versions are all equivalent to each other. (This is also why we refer to our assumption family as k-BDH and not k-DBDH; there is no interesting family of computational assumptions from which our decisional assumptions must be distinguished.)

We remark that discovering and choosing such a family turned out to be challenging. Initially, we considered the assumption family in which the adversary, given the same input values in $\mathbb{G}$, must distinguish $\prod_i e\bigl(g, v_i\bigr)^{xyr_i}$ from random in $\mathbb{G}_T$. This assumption family is easier to use than our k-BDH because the values v_i and $v_i^{r_i}$ are available to pair with g^x or g^y, the way that in DBDH we can use the pairing to compute any of $e(g,g)^{xy}$, $e(g,g)^{xz}$, $e(g,g)^{yz}$. However, it turns out that every member of this alternative assumption family is equivalent to DBDH.[2] The fact that the values $\{g^{r_i}\}$ are not supplied in the k-BDH challenge make constructing an IBE from k-BDH more challenging.

[1] Similarly, for attribute-based encryption, if attribute-hiding were established based on k-Linear, but payload-hiding were established based on DBDH, then one the assumption for one property would be weakened while the assumption for the other property would remain strong.

[2] The reduction makes use of the DBDH tuple $(g, \prod_i v_i^{r_i}, g^x, g^y, C \overset{?}{=} \prod_i e\bigl(g, v_i\bigr)^{xyr_i})$.

We justify our choice by arguing both that the k-BDH assumptions are no stronger than existing (decisional) assumptions in bilinear groups and that it is plausible that they are strictly weaker. The former follows in a relatively straightforward by finding appropriate reductions. We can show that in a given group the k-BDH assumption is no stronger than DBDH and for a given k the k-BDH assumption is no stronger than the k-Linear assumption, for all values of k.

Arguing that the assumptions are weaker is more nuanced. Whether certain assumptions hold or do not hold might vary with the choice of a group and clearly if $\mathcal{P} = \mathcal{NP}$ all assumptions are equally false. We give evidence that, for each k, the $(k+1)$-BDH assumption is strictly weaker than the k-BDH assumption (i.e., the $(k+1)$-BDH problem is strictly harder to solve than the k-BDH problem). As in previous proofs of this sort for Linear [11] and k-Linear [9], we rely on an argument in the generic group model [20,21]. We show that the $k+1$-BDH problem is generically hard even in the presence of an oracle that solves k-BDH.

We demonstrate the utility of our assumption family, by constructing a family of IBEs secure under k-BDH. The size of the public parameters, secret keys, and ciphertexts are all linear in the parameter k. One can view our family as a generalization of the Boneh-Boyen selectively secure IBE system [22].

In the full version of the paper [23], we extend our construction family to a family of hierarchical IBEs. These yield CCA-secure schemes via standard transformations [24,25]. In addition, we show how to produce a Waters-IBE–style variant [26] that is fully secure in the standard model.

Looking Ahead. In the future, we expect that one will be able to build cryptosystems from our k-BDH assumption where DBDH was previously used. However, as our experience with IBE has taught us, this might require new insights or techniques.

One interesting challenge is whether one can build more complex systems using the k-BDH assumption where the performance overhead is *additive* in k versus a multiplicative factor (which seems more natural). For instance, in existing (Key-Policy) Attribute-Based Encryption [27,28] systems, the size of a private key is proportional to a policy expressed as a boolean formula. If, the cost of using the k-BDH assumption only required adding $\approx k$ more group elements, this could be a relatively small key size overhead for reasonably chosen k. This is in contrast to blowing up the entire key size by a factor of k. A similar argument holds for other parameters such as ciphertext size and decryption time. In one datapoint suggesting that this might be possible, Freeman et. al. [18] recently built Lossy Trapdoor Functions in a novel way from the k-linear assumption which were rather efficient relative to the "natural" extension of the Peikert and Waters [29] DDH construction.

There also exist currently exist several functionalities where there are no known systems that reduce to DBDH. These include systems that appear to inherently on assumption related to source group elements such as Decision Linear. Examples of these include Groth-Sahai NIZKs [30], dual system encryption proofs [8], and the Boneh-Goh-Nissim [3] slightly 2-homomorphic encryption system.

Finally, an interesting question is which k values one might use in practice. For very large k, it might turn out that bit by bit encryption systems built from using hard core bits [31] and Computational Diffie-Hellman or Computational Bilinear Diffie-Hellman have comparable efficiency. When proposing systems, it is important to keep in mind where these lines cross. However, we believe for most practical choices of k the k-BDH assumption will yield more efficient systems.

2 The *k*-BDH Assumption and Relationships

Throughout this paper we work in a cyclic group $\mathbb{G}$ of order p where p is a large prime. g is a generator of $\mathbb{G}$. $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ denotes an admissible bilinear map where $\mathbb{G}_T$ is another cyclic group of order p. The standard definitions of bilinear maps and well known complexity assumptions BDH, DBDH, Linear [11] and k-Linear [10] are used.

Definition 1. *The k-BDH problem in $\langle \mathbb{G}, \mathbb{G}_T, e\rangle$ asks given (g, g^x, g^y, v_1, ..., v_k, $v_1^{r_1}$, ..., $v_k^{r_k}$, T) for $x, y, r_1, \ldots, r_k, c \in \mathbb{Z}_p^*$, $g, v_1, \ldots, v_k \in \mathbb{G}$ and $T \in \mathbb{G}_T$ does $T = e(g,g)^{xy(r_1+\cdots+r_k)}$ or is it the case that $T = e(g,g)^c$. An adversary, $\mathcal{B}$ outputs 1 if $T = e(g,g)^{xy(r_1+\cdots+r_k)}$ and 0 otherwise. $\mathcal{B}$ has advantage ϵ in solving k-BDH if*

$$|Pr[\mathcal{B}(g, g^x, g^y, v_1, \ldots, v_k, v_1^{r_1}, \ldots, v_k^{r_k}, e(g,g)^{xy(r_1+\cdots+r_k)}) = 1] - Pr[\mathcal{B}((g, g^x, g^y, v_1, \ldots, v_k, v_1^{r_1}, \ldots, v_k^{r_k}, e(g,g)^c) = 1]| \geq 2\epsilon.$$

Where the probability is taken over the random choice of $x, y, r_1, \ldots, r_k, c \in \mathbb{Z}_p^*$, $g, v_1, \ldots, v_k \in \mathbb{G}$ and the random bits consumed by $\mathcal{B}$.

The k-BDH Assumption is that if no t-time algorithm can achieve advantage at least ϵ in deciding the k-BDH problem in $\mathbb{G}$ and $\mathbb{G}_T$.

This is only a decisional problem. We show that, as a corollary of Theorem 4, the computational version is equivalent to the computational BDH problem.

2.1 *k*-BDH's Relationship to Standard Assumptions

In this subsection we state k-BDH's relationship to standard cryptographic assumptions; the proofs are straightforward and given in the full version [23].We also note that k-BDH is a member of the (R,S,T,f)-Diffie Hellman uber- assumption family [6]. Namely: $R = S = \{1, x, y, a_1, \ldots, a_k, a_1r_1, \ldots, a_kr_k\}$, $T = \{1\}$ and $f = xy(r_1 + \cdots + r_k)$ where $v_i = g^{a_i}$ for $1 \leq i \leq k$. Being part of this family tells us that it is generically secure, however, the focus on our work is to understand the relative strengths of assumptions (discussed in Section 4).

***k*-BDH's Relationship to *k*-Linear.** We will use the notation L_k to denote the k-Linear problem. If we wish to specify the k-Linear assumption in a specific group $\mathbb{G}$ we write $L_k^{\mathbb{G}}$, and similarly for $\mathbb{G}_T$.

Theorem 1. *If the $L_k^{\mathbb{G}}$ assumption holds, then so does the k-BDH assumption.*

Theorem 2. *If the k-BDH assumption holds, then so does the $L_k^{\mathbb{G}_T}$ assumption.*

Evidence that k-BDH is not equivalent to either $L_k^{\mathbb{G}}$ or $L_k^{\mathbb{G}_T}$. From the above theorems, the natural question arises: Is k-BDH equivalent to the linear assumption in either $\mathbb{G}$ or $\mathbb{G}_T$? Such an equivalence would imply that k-BDH assumption is neither a new assumption nor a new tool to construct a family of IBEs. Fortunately, separation of the assumptions appears to be related to the hard problem of inverting a bilinear map [32,33]. We show separation results for these assumptions in the full version of this paper [23] in the generic group model.

***k*-BDH's Relationship to BDH**

Theorem 3. *If the DBDH assumption holds, then so does the k-BDH assumption.*

Theorem 4. *If the Computational k-BDH assumption holds, then so does the Computational BDH assumption.*

Corollary 1. *The Computational k-BDH assumption is equivalent to the BDH assumption.*

Corollary 2. *The DBDH assumption is equivalent to the 1-BDH assumption.*

3 A Selectively Secure IBE System from the *k*-BDH Assumption

The standard definitions of IBE [1] and the selective-ID model [34] are used.

Using the k-BDH assumption in to create an IBE system is not straightforward. The main technical difficulty arises because the target in the k-BDH assumption, $(e(g,g)^{xy\sum_{i=1}^{k} r_i})$, is naturally an embedding of k Computational BDH problems: Given (g, g^x, g^y, g^{r_i}) find $e(g,g)^{xyr_i}$. However, we do not have the value g^{r_i} for each i. Instead, we have the pair $(v_i, v_i^{r_i})$, where v_i is a generator not used elsewhere.

We use a cancellation trick to effectively switch the base of the $v_i^{r_i}$. The setup algorithm will provide the values $e(g^x, v_i^{r_i})$ and v_i which are both taken to the same power in the encryption algorithm, namely y_i. The challenge needs to be crafted so that it takes $e(g^x, g^{r_i})$ to the power y instead of taking $e(g^x, v_i^{r_i})$ to the power y_i. To do this, we provide g^y in place of $v_i^{y_i}$. Since $v_i = g^{s_i}$ for some value of s_i we implicitly set $y_i = y/s_i$. Using the bilinear property of e, this effectively changes the value of the other term to $e(g^x, v_i^{r_i})^{y_i} = e(g,g)^{xr_i s_i \frac{y}{s_i}} = e(g^x, g^{r_i})^y$. The product of these values is exactly the target of the k-BDH assumption.

To ensure the challenge has the proper distribution in the view of the adversary it is required to randomize g^y for every value of k.

Our IBE construction is related to the Boneh-Boyen scheme in the selective-ID model [22], which is proven secure under the DBDH assumption. To prove our scheme is secure under the k-BDH assumption requires an alteration to the "Boneh-Boyen trick" for generating the private key for identities other than the target identity.

The "Boneh-Boyen trick" raises elements of the DBDH instance to cleverly selected random values to obtain a valid private key. However, constructing the same private key with the $\mathsf{KeyGen(ID)}$ algorithm is impossible as the random selections are unknown. Our construction uses the same idea but using multiple bases $(g,\, v_i)$ requires *three* components instead of two for the first term of the private key.

Specifically, we use $(v_i^{\hat{r_i}})^{-t_i/d} v_i^{t_i m_i} (g^x)^{d m_i}$ for the first term. $v_i^{\hat{r_i}}$ is the randomization of $v_i^{r_i}$ that permits the challenge have the proper distribution. The value d is a function of the target identity and the identity associated with the private key; t_i is used to randomize a public parameter; and m_i randomizes the private key. The first term is dependent on both g^x and $v_i^{r_i}$ from the k-BDH assumption.

The IBE system works as follows:
Setup : The public parameters are $(g, u = g^x, v_1 = g^{s_1}, \ldots, v_k = g^{s_k}, v_1^{\hat{r_1}}, \ldots, v_k^{\hat{r_k}}, w_1, \ldots, w_k)$. The values $s_1, \ldots, s_k, \hat{r_1}, \ldots, \hat{r_k}, x$ (chosen uniformly and independently at random) are kept as the master-key.

$\mathsf{KeyGen(ID)}$: Select random $n_1, \ldots, n_k \in \mathbb{Z}_p^*$. For each $1 \le i \le k$ output $(K_{A,i}, K_{B,i}) = ((g^{x\hat{r_i}}(w_i u^{\mathsf{ID}})^{n_i},\, v_i^{n_i})$.

$\mathsf{Encrypt}(m, \mathsf{ID})$: Select random $y_1, \ldots, y_k \in \mathbb{Z}_p^*$. Output $C_0 = m \prod_{1 \le i \le k} e(g^x, v_i^{\hat{r_i}})^{y_i}$ and for each $1 \le i \le k$ output $(C_{A,i}, C_{B,i}) = (v_i^{y_i}, (w_i u^{\mathsf{ID}})^{y_i})$ for a total of $2k+1$ values.

$\mathsf{Decrypt}(c)$: Output

$$\frac{C_0 \cdot \prod_{1 \le i \le k} e(K_{B,i}, C_{B,i})}{\prod_{1 \le i \le k} e(K_{A,i}, C_{A,i})} = \frac{m \prod_{1 \le i \le k} e(g^x, v_i^{\hat{r_i}})^{y_i} \cdot \prod_{1 \le i \le k} e(v_i^{n_i}, (w_i u^{\mathsf{ID}})^{y_i})}{\prod_{1 \le i \le k} e(g^{x\hat{r_i}}(w_i u^{\mathsf{ID}})^{n_i}, v_i^{y_i})} = m$$

3.1 Proof of Security

Theorem 5. *Suppose the k-BDH assumption holds in $\mathbb{G}$ and $\mathbb{G}_T$ (precisely, no t-time algorithm has advantage at least ϵ in solving the k-BDH problem in $\mathbb{G}$ and $\mathbb{G}_T$). Then the previously defined IBE system is $(t - \Theta(\tau k q), q, \epsilon)$-Selective-ID IND-CPA secure where τ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the IBE system. We build algorithm $\mathcal{B}$ to solve a decisional k-BDH instance $(g, g^x, g^y, v_1, \ldots, v_k, v_1^{r_1}, \ldots, v_k^{r_k}, T \overset{?}{=} e(g,g)^{xy(r_1+\cdots+r_k)})$. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a selective identity game as follows: Init: The selective identity games begins with $\mathcal{A}$ outputting an identity to attacked ID^*.

Setup: Algorithm $\mathcal{B}$ first selects random a_i, t_i for $1 \le i \le k$. It then sets the public parameters to: $(g, u = g^x, v_1, \ldots, v_k, v_1^{\hat{r_1}} = (v_1^{r_1})^{1/a_1}, \ldots, v_k^{\hat{r_k}} = (v_k^{r_k})^{1/a_k}, w_1 = v_1^{t_1}(g^x)^{-\mathsf{ID}^*}, \ldots, w_k = v_k^{t_k}(g^x)^{-\mathsf{ID}^*})$. These parameters are are all independent of ID^* in the view of $\mathcal{A}$. The a_i terms will serve as the way to randomize the challenge.

$\mathsf{Phase\ 1}$: $\mathcal{A}$ issues queries for the private key of an identity, ID. It must be the case that $\mathsf{ID} \neq \mathsf{ID}^*$. $\mathcal{B}$'s response is generated as follows for each value of $1 \le i \le k$:

Select random m_i. Let $d = \mathsf{ID} - \mathsf{ID}^*$. Output $(K_{A,i}, = (v_i^{\hat{r_i}})^{-t_i/d} v_i^{t_i m_i} (g^x)^{d m_i}$, $K_{B,i} = (v_i^{\hat{r_i}})^{(-1/d)} v_i^{m_i}$. For $n_i = -\hat{r_i}/d + m_i$, which implies $m_i = \hat{r_i}/d + n_i$, this is the expected value:

$$\begin{aligned}
&((v_i^{\hat{r_i}})^{-t_i/d} v_i^{t_i m_i} (g^x)^{d m_i}, (v_i^{\hat{r_i}})^{(-1/d)} v_i^{m_i}) \\
&= ((v_i^{\hat{r_i}})^{-t_i/d} v_i^{t_i(\hat{r_i}/d + n_i)} (g^x)^{d(\hat{r_i}/d + n_i)}, (v_i^{\hat{r_i}})^{(-1/d)} v_i^{(\hat{r_i}/d) + n_i)}) \\
&= (v_i^{t_i n_i} (g^x)^{\hat{r_i} + d n_i}, v_i^{n_i}) = (g^{x\hat{r_i}} (v_i^{t_i} g^{xd})^{n_i}, v_i^{n_i}) \\
&= (g^{x\hat{r_i}} (v_i^{t_i} g^{x(\mathsf{ID}-\mathsf{ID}^*)})^{n_i}, v_i^{n_i}) = (g^{x\hat{r_i}} (w_i g^{x\mathsf{ID}})^{n_i}, v_i^{n_i})
\end{aligned}$$

The second term is uniformly distributed among all elements in $\mathbb{Z}_p^*$ due to the selection of m_i. Private keys can be generated for all identities except ID^*.

$\mathsf{Challenge}(m_0, m_1)$: $\mathcal{B}$ picks random bit $b \in \{0,1\}$. The response is: $(C_0$, $(C_{A,1}$, $C_{B,1})$, ... ,$(C_{A,k}$, $C_{B,k}))$. $\mathcal{B}$ sets $C_0 = m_b T$ and for each i from 1 to k it sets:

$$C_{A,i} = (g^y)^{a_i}, \quad C_{B,i} = (g^y)^{a_i \cdot t_i}.$$

We observe that $(g^y)^{a_i} = v_i^{y_i}$ and that $(g^y)^{a_i t_i} = v_i^{t_i y_i} = (w_1 u^{\mathsf{ID}^*})^{y_i}$ from which correctness follows. The simulator's ability to construct the second term in this manner follows directly from the fact that the encrypted identity is ID^* and no g^x term appears in $w_1 u^{\mathsf{ID}^*}$.

For each value of i, this implicitly sets $y a_i = s_i y_i$ or $y_i = y a_i / s_i$. If the input is a valid k-BDH tuple then the response is drawn from a uniform distribution and $m_b T$ is the expected value:

$$\begin{aligned}
m_b T = m_b \prod_{1 \le i \le k} e(g,g)^{x y r_i} &= m_b \prod_{1 \le i \le k} e(g^x, g^{s_i r_i / a_i})^{y a_i / s_i} \\
= m_b \prod_{1 \le i \le k} e(g^x, v_i^{r_i / a_i})^{y_i} &= m_b \prod_{1 \le i \le k} e(g^x, v_i^{\hat{r_i}})^{y_i}
\end{aligned}$$

If T is not a valid k-BDH tuple then the distribution is uniform and independent of b.

Phase 2: $\mathcal{A}$ issues more private key queries. It is exactly the same as Phase 1.

Guess: $\mathcal{A}$ outputs a guess of $b' \in \{0, 1\}$. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning T is a valid k-BDH tuple. Otherwise, it is not a valid k-BDH tuple and the output is 0.

When the input is a valid k-BDH instance, $\mathcal{A}$ must satisfy $|\Pr[b = b'] - \frac{1}{2}| \geq \epsilon$. When the input is not a valid k-BDH instance, the input is uniform and independent and $\Pr[b = b'] = \frac{1}{2}$. Therefore, we have, as required:

$$|\Pr[\mathcal{B}(\text{valid } k\text{-BDH}) = 1] - \Pr[\mathcal{B}(\text{not valid } k\text{-BDH}) = 1]| \geq |(\frac{1}{2} + \epsilon) - \frac{1}{2}| \geq \epsilon.$$

3.2 Efficiency

Assume that the value $e(g^x, v_i^{r_i})$ is precomputed for all values $1 \leq i \leq k$. Each encryption takes k exponentiations and k group operations in $\mathbb{G}_T$, $2k + 1$ exponentiations and k group operation in $\mathbb{G}$. Decryption requires $2k$ bilinear map computations, one inversion and $2k + 2$ group operations in $\mathbb{G}_T$.

3.3 Extensions

This construction fits in Boneh-Boyen framework. We give the natural extension to a hierarchical IBE and to a fully secure IND-CPA scheme in the style of [26] in the full version of the paper [23].

4 Relationship between k-BDH Problems

In this section we prove that the k-BDH family of problems becomes progressively weaker. Informally, this means that an oracle for k-BDH does not help in solving a $(k + 1)$-BDH instance.

The proof uses the generic group model [35,20,21]. This idealized version of a group retains the important properties of the group while facilitating reasoning about its minimal possible assumptions. If a statement cannot be proven in the generic group model then it is impossible to find a group for which the statement holds. The generic group model has been used to reason about complexity assumptions both with bilinear maps [5,6] and without bilinear maps [21].

The closely related proof for the separation of k-Linear family of assumptions [10] could not be used directly. This stems from the fact that a standard multilinear map [36] cannot be used to solve k-BDH. We create a modified k-multilinear map that takes as input k elements in $\mathbb{G}$ and 1 element in $\mathbb{G}_T$ (which is the result of a bilinear map on two elements in $\mathbb{G}$) and produces an output in a third group $\mathbb{G}_M$ (the target group of the k-multilinear map). The modified k-multilinear map acts as an oracle for k-BDH. The main technical difficulty is showing that all inputs to the k-multilinear map fail to produce a multiple of the target element in the $(k + 1)$-BDH instance.

Theorem 6. *If the k-BDH assumption holds, then so does the* $(k+1)$*-BDH assumption.*

Proof. Informally, this means that if $(k+1)$-BDH is easy, then k-BDH is also easy. Suppose we have an oracle $\mathcal{A}$ for $(k+1)$-BDH. $\mathcal{A}$ can be used to solve an k-BDH instance $(g, g^x, g^y, v_1, \ldots, v_k, v_1^{r_1}, \ldots, v_k^{r_k}, T)$. Select random $v_{k+1} \in \mathbb{G}$ and $r_{k+1} \in \mathbb{Z}_p^*$ and run $\mathcal{A}$ on input $(g, g^x, g^y, v_1, \ldots, v_k, v_{k+1}, v_1^{r_1}, \ldots, v_k^{r_k}, v_{k+1}^{r_{k+1}}, T \cdot e(g^x, g^y)^{r_{k+1}})$. By returning the same value as $\mathcal{A}$, the simulation is perfect.

As in the k-Linear generic group separation proof [10], we prove a stronger result by means of a multilinear map [36,37] in Theorem 7. A k multilinear map is an efficiently computable map $e_k : \mathbb{G}^k \to \mathbb{G}_M$ such that $e_k(g_1^{a_1}, \ldots, g_k^{a_k}) = e_k(g_1, \ldots, g_k)^{\prod_{i=1}^k a_i}$ for all $g_1, \ldots, g_k \in \mathbb{G}$ and $a_1, \ldots, a_k \in \mathbb{Z}_p$; and $e_k(g, \ldots, g) \neq 1$. Here, we consider a modified k-multilinear map: $\hat{e_k} : \mathbb{G}_T \times \mathbb{G}^k \to \mathbb{G}_M$ where $\mathbb{G}_T$ is the group resulting from a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We define $\hat{e_k} : (e(g_x, g_y)^{a_T}, g_1^{a_1}, \ldots, g_k^{a_k}) = \hat{e_k}(e(g_x, g_y), g_1, \ldots, g_k)^{a_T \prod_{i=1}^k a_i}$.

Lemma 1. *Given a modified k-multilinear map there is an efficient algorithm to solve k-BDH.*

Proof. On input a k-BDH instance $(g, g^x, g^y, v_1, \ldots, v_k, v_1^{r_1}, \ldots, v_k^{r_k}, T)$ output "yes" if $\hat{e_k}(T, v_1, \ldots, v_k) \overset{?}{=} \prod_{i=1}^{k} \hat{e_k}(e(g^x, g^y), v_1, \ldots, v_{i-1}, v_i^{r_i}, v_{i+1}, v_k)$ and "no" otherwise. This is correct because

$$\prod_{i=1}^{k} \hat{e_k}(e(g^x, g^y), v_1, \ldots, v_{i-1}, v_i^{r_i}, v_{i+1}, v_k) = \prod_{i=1}^{k} \hat{e_k}(e(g, g), v_1, \ldots, v_k)^{xyr_i}$$
$$= \hat{e_k}(e(g, g), v_1, \ldots, v_k)^{xy \sum_{i=1}^k r_i}$$

and when $T = e(g,g)^{xy \sum_{i=1}^k r_i}$ equality holds as required.

In the generic group model, elements of $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{G}_M$ are encoded as opaque strings such that only equality can be tested by the adversary. To perform operations in the group the adversary queries oracles. The oracles map the opaque string representations to elements of $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{G}_M$ using ξ_G,ξ_T and ξ_M respectively. In our case, we provide the adversary with oracles to perform Group Action in each group, Inversion in each group, Bilinear Map for $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and Modified k-Multilinear Map for $\mathbb{G}_T \times \mathbb{G}^k$.

Theorem 7. *Let* $\mathcal{A}$ *be an algorithm that solves* $(k+1)$*-BDH in the generic group model making a total of q queries to the oracles computing the group action in* $\mathbb{G}$, $\mathbb{G}_T$ *and* $\mathbb{G}_M$*, the oracles computing inversion in* $\mathbb{G}$, $\mathbb{G}_T$ *and* $\mathbb{G}_M$*, the bilinear map oracle and an oracle for modified k-multilinear map. Then* $\mathcal{A}$*'s probability of success is bounded by*

$$\epsilon \leq \frac{(k+5)(q+2k+5)^2}{p}.$$

Proof. Consider an algorithm $\mathcal{B}$ that interacts with $\mathcal{A}$ as follows.

Let g be a randomly selected generator of $\mathbb{G}$. Select random x, y, v_1, …, v_{k+1}, $r_1, \ldots, r_{k+1}$, $c \in \mathbb{Z}_p$ as well as random bit $d \in \{0, 1\}$. Set $T_d = e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$ and $T_{1-d} = e(g, g)^c$. $\mathcal{A}$ is given ($\xi_G(g)$, $\xi_G(g^x)$, $\xi_G(g^y)$, $\xi_G(g^{v_1})$, …, $\xi_G(g^{v_{k+1}})$, $\xi_G(g^{v_1 r_1})$, …, $\xi_G(g^{v_{k+1} r_{k+1}})$, $\xi_T(T_0)$, $\xi_T(T_1)$) with the goal of guessing d.

$\mathcal{B}$ keeps track of the elements known to $\mathcal{A}$ as three lists: $L_G = \{(F_{G,i}, \xi_{G,i})\}$, $L_T = \{(F_{T,i}, \xi_{T,i})\}$ and $L_M = \{(F_{M,i}, \xi_{M,i})\}$. The first element of each list is the internal representation kept by $\mathcal{B}$- represented as a polynomial in the ring $\mathbb{Z}_p[1, x, y, v_1, \ldots, v_{k+1}, r_1, \ldots, r_{k+1}, c]$. The set of all elements in these rings are denoted F_G, F_T and F_M. The second element is the opaque representation known to $\mathcal{A}$. $\mathcal{B}$ handles oracle queries from $\mathcal{A}$ by calculating the correct value and checking to see if a the corresponding external representation already exists. If so, the corresponding known representation is returned; otherwise $\mathcal{B}$ generates a distinct random string to serve as the external representation and adds it to the respective list. We assume that the domains of ξ_G, ξ_T and ξ_M are sufficiently large so that the probability that algorithm $\mathcal{A}$ makes queries for an element other than one obtained through $\mathcal{B}$ is negligible.

Oracle queries from $\mathcal{A}$ are handled by $\mathcal{B}$ as follows:

Group Action: Given elements in $\mathbb{G}$ with internal representations $F_{G,i}$ and $F_{G,j}$ compute $F' = F_{G,i} + F_{G,j}$. If there does not already exist an external representation of the value F' then generate $\xi_G(F')$ and add $(F', \xi_G(F'))$ to L_G . Return $\xi_G(F')$. Group Action for $\mathbb{G}_T$ and $\mathbb{G}_M$ are handled analogously. Denote the number of Group Action queries made in $\mathbb{G}$ as q_{G_g}, the number of Group Action queries made in $\mathbb{G}_T$ as q_{T_g} and the number of Group Action queries made in $\mathbb{G}_M$ as q_{M_g}.

Inversion: Given an element in $\mathbb{G}$ with internal representation $F_{G,i}$ set $F' = -F_{G,i}$. If there does not already exist an external representation of the value F' generate $\xi_G(F')$ and add $(F', \xi_G(F'))$ to L_G . Return $\xi_G(F')$. Inversion for $\mathbb{G}_T$ and $\mathbb{G}_M$ are handled analogously. Denote the number of Group Action queries made in $\mathbb{G}$ as q_{G_i}, the number of Group Action queries made in $\mathbb{G}_T$ as q_{T_i} and the number of Group Action queries made in $\mathbb{G}_M$ as q_{M_i}.

Bilinear Map (e): Given elements in $\mathbb{G}$ with internal representations $F_{G,i}$ and $F_{G,j}$ calculate $F' = F_{G,i} \cdot F_{G,j}$. If there does not already exist an external representation of the value F' generate $\xi_T(F')$ and add $(F', \xi_T(F'))$ to L_T. Return $\xi_T(F')$. Let q_B denote the number of bilinear map queries made.

Modified k-Multilinear Map ($\hat{e_k}$): Given elements in $\mathbb{G}$ with internal representations $F_{G,v1}$, …, $F_{G,vk}$, and an element in $\mathbb{G}_T$ with internal representation $F_{T,j}$. Compute $F' = F_{T,j} \prod_{i=1}^{k} F_{G,vi}$. If there does not already exist an external representation of the value F' generate $\xi_M(F')$ and add $(F', \xi_M(F'))$ to L_M. Return $\xi_M(F')$.

Elements in F_G have at most degree 2; elements of F_T have at most degree 4; elements in F_M have degree at most $2k + 4$. The input elements that are in $\mathbb{G}$ have corresponding elements in F_G with degree at most 2 and the elements in $\mathbb{G}_T$ have corresponding elements in F_T with degree at most 3. The group action and inversion operations cannot increase the degree of the polynomials in F_G,F_T

or F_M. The Bilinear Map operation uses elements in $\mathbb{G}$ to produce elements of at most degree $2+2=4$ in F_T. The Modified Multilinear Map produces elements of at most degree $4+k(2)$ in F_M.

Finally, $\mathcal{A}$ halts and outputs a guess of d' for d. $\mathcal{B}$ now selects random $g^* \in \mathbb{G}$ and x^*, y^*, v_1^*, …, v_{k+1}^*, r_1^*, …, r_{k+1}^*, $c^* \in \mathbb{Z}_p$. T_b is set to $e(g^*, g^*)^{x^* y^* \sum_{i=1}^{k+1} r_i^*}$ and $T_{1-b} = e(g^*, g^*)^{c^*}$. All elements besides T_b are independent of each other. Therefore the simulation engineered by $\mathcal{B}$ is consistent with these values unless one of the following events occur:

- Two values in F_G have the same representation in $\mathbb{G}$
- Two values in F_T have the same representation in $\mathbb{G}_T$
- Two values in F_M have the same representation in $\mathbb{G}_M$
- Using a bilinear map on values in F_G, it is possible to find a multiple of $e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$ in F_T.
- Using a modified k-multilinear map on values in F_G, it is possible to find a multiple of $e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$ in F_M.

The input elements are all chosen independently. Since $\mathcal{A}$ makes $q_{G_g} + q_{G_i}$ group actions or inversion queries for group $\mathbb{G}$ the corresponding elements in F_G are at most degree 2 and the probability of a collision is $\binom{q_{G_g}+q_{G_i}+2(k+1)+3}{2}\frac{2}{p}$. For the elements in $\mathbb{G}_T$ there $q_{T_g} + q_{T_i} + q_B$ group actions or inversion or bilinear map queries are made resulting in elements in $\mathbb{G}_T$. Since elements in $\mathbb{G}_T$ have corresponding polynomials in F_T with degree at most 4 the probability of a collision is $\binom{q_{T_g}+q_{T_i}+q_B+2}{2}\frac{4}{p}$. For each of the group actions in $\mathbb{G}_M$, inversion in $\mathbb{G}_M$ and k-Modified Multilinear Map queries the probability of a collision is $\binom{q_{M_g}+q_{M_i}+q_K}{2}\frac{2k+4}{p}$.

Next, we show the probability of finding a multiple of $e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$ from the terms in F_G is zero. If a multiple exists, it must be formed using at least one bilinear map operation. Since x, y, r_i all appear in T_b then the product of at least two of these values must appear in the same element in F_G for each value of i, $1 \le i \le k+1$. This is impossible by the following claim:

Claim: It is impossible for any two of x, y, r_i to appear in the same monomial in F_G:

Proof. We show that each way to choose two of the three values to appear in the same term is impossible:

- x and y appear in the same term. This requires creating a multiple of the polynomial xy. We are initially given the polynomials x and y each of degree 1 (and polynomials that are independent of x and y). This means from polynomial of degree 1 we must create a polynomial of degree 2 also in F_G. Only the group action and inversion oracles result in new elements in F_G. However, the output of these oracles cannot increase the degree of a monomial. Thus we cannot create monomials of degree greater than 1 from x and y. In particular, xy cannot be created by the adversary.

– x and r_i appear in the same term. This requires creating a multiple of xr_i namely axr_i. Since the term r_i never appears without v_i it follows that $v_i \mid a$ and we can rewrite axr_i as $a'xr_iv_i$. This is a polynomial of degree 3. It is impossible to create a polynomial of degree greater than 2 in F_G. So x and r_i cannot appear in the same term.
– y and r_i appear in the same term. This follows from a symmetric argument that x and r_i cannot appear in the same term.

Finally, we claim that it is impossible to find a multiple of $e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$ in F_M. In order to use the modified k-multilinear map to find a multiple of a target value, $T_d \overset{?}{=} e(g^x, g^y)^{\sum_{i=1}^{k+1} r_i}$, at least one $\hat{e_k}$ operation involving a multiple of T_d is required. The only option is to use T_d as the input element in $\mathbb{G}_T$. The modified k-multilinear map produces a multiple of $xy\sum_{i=1}^{k+1} r_i$, namely $Axy\sum_{i=1}^{k+1} r_i$. $\mathcal{A}$ must then use combination of oracle calls, denoted $\mathcal{F}$, using only values in F_G to form $Axy\sum_{i=1}^{k+1} r_i$ so that it can test equality.[3]

All inputs in F_G containing r_i also contain v_i. As a result, any monomial divisible by r_i is also divisible by v_i. Every type of oracle call preserves this property. In particular, consider the polynomial $Axy\sum_{i=1}^{k+1} r_i$ constructed by the adversary in $\mathcal{F}$. It is required that each monomial in the expansion of $Axyr_i$ must be divisible by v_i. It follows that for each of the $k+1$ values of v_i it is the case that $v_i|A$.[4] Specifically, A is divisible by $\prod_{i=1}^{k+1} v_i$.

For a given value i, the value $Axyr_i$ is divisible by $k+4$ values: x, y, v_1, …, v_{k+1}, and r_i. Producing such a term requires taking the product of at least $k+3$ terms available to the adversary (x and y only appear on their own and it is impossible to produce a multiple of v_iv_j in F_G using the group action and inversion oracles). However, the bilinear map can only take the product of 2 values and the modified k-multilinear map can only take the product from a bilinear map and k additional values for a total of $k+2$. Consequently, we deduce that the adversary cannot synthesize a multiple of $xy\sum_{i=1}^{k+1} r_i$ in F_M to cause a collision.

The probability of finding a collision is bounded by

$$\begin{aligned}\epsilon \le & \binom{q_{G_g} + q_{G_i} + 2(k+1) + 3}{2}\frac{2}{p} + \binom{q_{T_g} + q_{T_i} + q_B + 2}{2}\frac{4}{p} \\ & + \binom{q_{T_m} + q_{T_i} + q_k}{2}\frac{(2k+4)}{p} \\ & < \frac{(q+2k+5)^2 + 2(q+2)^2 + (k+2)q^2}{p} < \frac{(k+5)(q+2k+5)^2}{p}\end{aligned}$$

The combination of these two theorems implies: DBDH=1-BDH $\lnsim$ 2-BDH $\lnsim \cdots \lnsim$ k-BDH $\lnsim$ $(k+1)$-BDH $\lnsim \cdots$.

[3] $\mathcal{A}$ could first perform $\hat{e_k}(CT_d + D, n_1, \dots, n_k)$ for constant C and a polynomial D that does not contain T_d. It would then perform some combination of oracle calls,$\mathcal{F}$, to produce a value equal to $\hat{e_k}(CT_d + D, n_1, \dots, n_k)$. However, an equivalent test is to first perform $\hat{e_k}(CT_d, n_1, \dots, n_k)$ and then test equality with $\mathcal{F}/\hat{e_k}(D, n_1, \dots, n_k)$.

[4] For a more detailed argument see [10].

4.1 Relationship between k and the Group Size

From Theorem 7, we know that increasing k increases security. The generic attack on k-BDH appears to require $O(k)$ discrete logarithm calculations, and that solving t discrete logarithm problems on a given curve appears to require $O(t)$ times solving one problem; assuming that, the generic attack scales linearly with k.

Another means of increasing security is to increase the group size. An interesting question is, "what is the equivalent increase in group size if we increase k to $k+1$." We assume finding the discrete log is a function, f, of the order of the group. Then in the generic attack on k-BDH where $\mathbb{G}$ has prime order p increasing k to $k+1$ is approximately equivalent to increasing the group size from p to $f^{-1}(\frac{(k+1)f(p)}{k})$.

5 Conclusions and Future Work

We have proposed k-BDH, a family of assumptions generalizing the DBDH assumption. We have given evidence, using the generic group model, that assumptions in the k-BDH family become strictly weaker with increasing values of the parameter k. Unlike the k-Linear family of assumptions, k-BDH makes a natural tool for constructing pairing-based cryptosystems, including IBEs. We have demonstrated this by constructing a family of IBEs in which the kth member is secure based on k-BDH. Our IBE family fits in the Boneh-Boyen framework. Our k-BDH family allows IBEs to be instantiated with an assumption safety buffer for the first time.

We hope that, like k-Linear, our k-BDH assumption family will see widespread use. We believe that it will be especially well suited for constructing attribute-based encryption and other forms of functional encryption. In addition, we believe that dual system encryption techniques could be applied to k-BDH, yielding more efficient cryptosystems with tighter security reductions.

An important open problem arises from the fact that the k-BDH assumptions are all no weaker than computational BDH (just as the k-Linear assumptions are all no weaker than CDH). Because the components of our IBE grow with k, there may be a crossover point beyond which an IBE based on hard-core bits of the computational BDH problem is more efficient than one based on k-BDH.

References

1. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. SIAM J. Computing 32(3), 586–615 (2003); Extended abstract in Proceedings of Crypto 2001
2. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)

3. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
4. Abdalla, M., Pointcheval, D.: Interactive Diffie-Hellman Assumptions with Applications to Password-Based Authentication. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 341–356. Springer, Heidelberg (2005)
5. 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)
6. Boyen, X.: The Uber-Assumption Family – A Unified Complexity Framework for Bilinear Groups. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 39–56. Springer, Heidelberg (2008)
7. Gentry, C., Halevi, S.: Hierarchical Identity Based Encryption with Polynomially Many Levels. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 437–456. Springer, Heidelberg (2009)
8. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
9. Hofheinz, D., Kiltz, E.: Secure Hybrid Encryption from Weakened Key Encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
10. Shacham, H.: A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074 (2007), http://eprint.iacr.org/
11. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
12. Lewko, A.B., Waters, B.: Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In: ACM Conference on Computer and Communications Security, pp. 112–120. ACM (November 2009)
13. Boneh, D., Montgomery, H., Raghunathan, A.: Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In: Keromytis, A., Shmatikov, V. (eds.) Proceedings of CCS 2010, pp. 131–140. ACM Press (October 2010)
14. 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)
15. Camenisch, J., Chandran, N., Shoup, V.: A Public Key Encryption Scheme Secure against Key Dependent Chosen Plaintext and Adaptive Chosen Ciphertext Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009)
16. Naor, M., Segev, G.: Public-Key Cryptosystems Resilient to Key Leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
17. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: Trevisan, L. (ed.) Proceedings of FOCS 2010, pp. 511–520. IEEE Computer Society (October 2010)
18. Freeman, D.M., Goldreich, O., Kiltz, E., Rosen, A., Segev, G.: More Constructions of Lossy and Correlation-Secure Trapdoor Functions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 279–295. Springer, Heidelberg (2010)

19. Boyle, E., Segev, G., Wichs, D.: Fully Leakage-Resilient Signatures. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 89–108. Springer, Heidelberg (2011)
20. Nechaev, V.I.: Complexity of a determinate algorithm for the discrete logarithm. Mathematical Notes 55(2), 165–172 (1994)
21. Shoup, V.: Lower Bounds for Discrete Logarithms and Related Problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
22. 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)
23. Benson, K., Shacham, H., Waters, B.: The k-bdh assumption family: Bilinear map cryptography from progressively weaker assumptions. Cryptology ePrint Archive, Report 2012 (2012), http://eprint.iacr.org/
24. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. 36(5), 1301–1328 (2007)
25. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: Atluri, V., Meadows, C., Juels, A. (eds.) Proceedings of CCS 2005, pp. 320–329. ACM Press (November 2005)
26. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
27. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of ACM Conference on Computer and Communications Security 2006, pp. 89–98. ACM (November 2006)
28. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
29. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Dwork, C. (ed.) Proceedings of STOC 2008, pp. 187–196. ACM (May 2008)
30. Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
31. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: Proceedings of STOC 1989, pp. 25–32. ACM (1989)
32. Verheul, E.R.: Evidence that XTR Is More Secure than Supersingular Elliptic Curve Cryptosystems. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 195–210. Springer, Heidelberg (2001)
33. Moody, D.: The diffie hellman problem and generalization of verheuls theorem. Designs, Codes and Cryptography 52, 381–390 (2009)
34. Canetti, R., Halevi, S., Katz, J.: A Forward-secure Public-key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
35. Babai, L., Szemerédi, E.: On the complexity of matrix group problems I. In: Proceedings of FOCS 1984, pp. 229–240. IEEE Computer Society (October 1984)
36. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. In: Topics in Algebraic and Noncommutative Geometry: Proceedings in Memory of Ruth Michler. Contemporary Mathematics, vol. 324, pp. 71–90. American Mathematical Society (2003)
37. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices and applications. Cryptology ePrint Archive, Report 2012/610 (2012)

Accountable Authority Identity-Based Encryption with Public Traceability

Junzuo Lai[1], Robert H. Deng[1], Yunlei Zhao[2,3], and Jian Weng[4,5]

[1] School of Information Systems, Singapore Management University, Singapore
{junzuolai,robertdeng}@smu.edu.sg
[2] Software School, Fudan University, China
[3] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, China
yunleizhao@gmail.com
[4] Department of Computer Science, and Emergency Technology Research Center of Risk Evaluation and Prewarning on Public Network Security, Jinan University, China
[5] Shanghai Key Laboratory of Integrate Administration Technologies for Information Security, China
cryptjweng@gmail.com

Abstract. At Crypto'07, Goyal introduced the notion of accountable authority identity-based encryption (A-IBE) in order to mitigate the inherent key escrow problem in identity-based encryption, and proposed two concrete constructions. In an A-IBE system, if the private key generator (PKG) distributes a decryption key or produces an unauthorized decryption box for a user maliciously, it runs the risk of being caught and sued in the court of law with the help of a tracing algorithm. Subsequent efforts focused on constructions of A-IBE schemes with enhanced security. In these A-IBE constructions, the tracing algorithm needs to take a user's decryption key as input. If the user lost his key or is deliberately uncooperative in court, then we cannot implicate the PKG or the user. An interesting open problem left by Goyal et al. at CCS'08 is to consider the possibility of tracing a decryption box using only a public tracing key, or with the assistance of a tracing authority. In this paper, we address this problem positively. We first extend the original model of A-IBE to accommodate public traceability, and then propose an A-IBE scheme in the new model. To the best of our knowledge, the proposed scheme is the first A-IBE with public traceability.

Keywords: Accountable Authority, Identity-Based Encryption, Public Traceability.

1 Introduction

Identity-based encryption (IBE), first introduced by Shamir [30], simplifies public key and certificate management in a traditional public key infrastructure (PKI). In an IBE system, the public key of a user may be an arbitrary string such as an e-mail address or an IP address belonging to a network host; the corresponding private key is generated by a trusted authority called a private key generator (PKG). In this way, a certificate for a public key is implicitly provided and it is no longer necessary to explicitly authenticate the public key. Although the notion of IBE was introduced in 1984 [30], it took nearly two decades for the problem of designing an IBE scheme to be solved by Boneh-Franklin [9] and Cocks [11]. Since then, various extensions have been proposed, such as hierarchical IBE [22,14], anonymous IBE [8,1], fuzzy IBE [28] and attribute-based encryption (ABE) [20,6].

In spite of its appealing advantages, IBE suffers from the key escrow problem. The PKG is able to generate and distribute decryption keys for any identity without any risk of being caught and sued in a court of law; thus the security of an IBE system relies heavily on trusting the PKG. One approach to reduce this trust is to employ multiple PKGs [9]. In this approach, multiple PKGs share the master secret key for the IBE system; and the private key for an identity is generated in a threshold manner. However, this approach inevitably entails extra communication and infrastructure cost, and the problem of collusion among collaborating PKGs remains.

Goyal [18] introduced the notion of accountable authority IBE (A-IBE) as a new approach to mitigate the key escrow problem in IBE. In an A-IBE system, a user gets his decryption key from the PKG using a *key generation protocol* which allows the user to obtain a *single* decryption key while concealing to the PKG which key he obtained. Now, if the PKG generates a decryption key for a user for malicious usage, it runs the risk of being caught and sued in the court of law with the help of a tracing algorithm. Goyal [18] formalized the security requirements for A-IBE using three games: IND-ID-CPA, DishonestPKG and DishonestUser. The first one is the standard security requirement for IBE schemes. The latter two model the usefulness of the tracing algorithm. DishonestPKG requires that if a dishonest PKG generates a decryption key for a user maliciously, the tracing algorithm could implicate the PKG. On the other hand, if a decryption key is generated maliciously by colluding users, DishonestUser could implicate the users.

Goyal [18] presented two constructions for A-IBE. The first construction, which we denote as Goyal-1, builds on Gentry's IBE scheme [13] and depends on seeing the actual maliciously issued decryption key, and as such it is referred to having *white-box* traceability. Specifically, white-box traceability only provides the guarantee that given a *well-formed* decryption key for a user, the tracing algorithm could implicate its source. As noted by Goyal, this kind of white-box traceability is insufficient in practice. For example, instead of distributing a decryption key itself in the clear, the PKG could release a decryption box which allows decryption of ciphertexts encrypted for a user and yet does not contain a decryption key in any canonical form. Since the exact information used to create this decryption box is assumed hidden, A-IBE schemes that can deal with this type of decryption boxes are referred to as having *black-box* traceability. A natural question is how to construct A-IBE with black-box traceability in which, seeing a decryption box for a user, the tracing algorithm could implicate its source. The second construction of [18] (denoted as Goyal-2), which is based on the IBE scheme proposed by Waters [31] and the fuzzy IBE scheme proposed by Sahai and Waters [28], achieves security guarantees according to a *weak* black-box traceability model in which the malicious PKG has no access to a decryption oracle in the attack game.

Subsequently, Goyal et al. [19] improved Goyal-2 to obtain full black-box traceability in the DishonestPKG security game. An A-IBE scheme has full black-box, or simply black-box, traceability if the malicious PKG is given access to decryption queries and no assumptions are made regarding how the decryption box works. However, DishonestUser security of the A-IBE scheme in [19] is only satisfied in the selective-ID setting, where the adversary must choose an identity as its target before seeing the public parameters. Since a real-world adversary has the ability to pick target adaptively, selective-ID is not a reasonable requirement for security. Libert and Vergnaud [25] proposed an adaptive-ID DishonestUser secure A-IBE scheme at the cost of only obtaining weak black-box traceability. Recently, Sahai and Seyalioglu [27] presented the first A-IBE scheme which has adaptive-ID DishonestUser security and full black-box traceability.

In all existing A-IBE schemes, in order to trace the source of a decryption box for a user, the tracing algorithm needs to take the user's decryption key as input. If the user lost his key or is deliberately uncooperative, then the tracing algorithm cannot implicate the PKG or the user. One interesting open problem left by Goyal et al. at CCS'08 is to consider the possibility of tracing a decryption box using only a public tracing key, or with the assistance of a tracing authority. This paper is motivated by the task of resolving the problem.

1.1 Our Contributions

In this paper, we first extend the original definition of A-IBE [18] to account for public traceability. In our definition, at the end of the key generation protocol, a user obtains a *single* decryption key without letting the PKG know which key he obtained, and the PKG obtains a public tracing key corresponding to the user's decryption key. Thus, the responsibility of the PKG includes maintaining a public tracing key list $\mathcal{TK}$ in addition to issuing decryption keys to users. The PKG is required to ensure the integrity of $\mathcal{TK}$. In case a user's public tracing key in $\mathcal{TK}$ is modified by the PKG intentionally or otherwise, with the help of an augmented algorithm Judge, the user can prove to a judge that his public tracing key has indeed been tampered with. We require that this proving process do not leak additional information about the user's decryption key except the tracing key itself. We also extend the original security model of A-IBE [18] due to the introduction of the public tracing key which is derived from a user's decryption key.

Then, we present the first A-IBE scheme with public traceability that combines Goyal-1 with a public weak black-box tracing mechanism. We prove that our scheme is IND-ID-CPA secure, weak black-box DishonestPKG secure and adaptive-ID DishonestUser secure. An overview comparing our A-ABE scheme with other A-IBE schemes is given in Table 1.

Table 1. Comparison of A-IBE schemes

Scheme	IND-ID-CPA	DishonestPKG	DishonestUser	Public Traceability
Goyal-1 [18]	IND-ID-CPA	white-box	white-box	no
Goyal-2 [18]	IND-ID-CPA	weak black-box	selective-ID	no
[19]	IND-ID-CPA	black-box	selective-ID	no
[25]	IND-ID-CPA	weak black-box	adaptive-ID	no
[27]	IND-ID-CPA	black-box	adaptive-ID	no
Ours	IND-ID-CPA	weak black-box	adaptive-ID	yes

1.2 Related Work

Au et al. [3] extended the work of Goyal [18] by introducing a retrieval algorithm while causes the PKG's master secret key to be revealed if more than one decryption key for a user are released. Their security proofs require the PKG to release a well-formed decryption key; thus traceability and retrievability remain only in the white-box model. Certificateless public key encryption (CL-PKE) [2] and certificate-based encryption (CBE) [12] were introduced in order to remove the key escrow problem in IBE. In these paradigms, every user has a traditional public key (though not explicitly certified); thus these solutions come at the cost of losing the benefit of human-memorizable public keys (such as an e-mail address).

1.3 Organization

The rest of the paper is organized as follows. In Section 2, we review some standard notations and cryptographic definitions. In Section 3, we describe the formal definition and security model of A-IBE with public traceability. The proposed A-IBE scheme with public traceability and its security analysis are presented in Section 4. Finally, we state our conclusion in Section 5.

2 Preliminaries

If S is a set, then $s \overset{\$}{\leftarrow} S$ denotes the operation of picking an element s uniformly at random from S. Let $\mathbb{N}$ denote the set of natural numbers. If $\lambda \in \mathbb{N}$ then 1^λ denotes the string of λ ones. Let $z \leftarrow \mathsf{A}(x, y, \ldots)$ denote the operation of running an algorithm A with inputs $(x, y, \ldots)$ and output z. A function $f(\lambda)$ is *negligible* if for every $c > 0$ there exists a λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$.

2.1 Bilinear Groups

Let $\mathcal{G}$ be an algorithm that takes as input a security parameter 1^λ and outputs a tuple $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of prime order p, g is a generator of $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map such that

1. (Bilinearity:) $\forall g_1, g_2 \in \mathbb{G}, a, b \in \mathbb{Z}_p, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
2. (Non-degeneracy:) $e(g, g) \neq 1$.

We say that $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ is a bilinear group system if multiplication in $\mathbb{G}$ and $\mathbb{G}_T$, as well as the bilinear map e, are computable in time polynomial in λ.

2.2 Complexity Assumptions

We state the complexity assumptions we use below.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption Let $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ be a bilinear group system and $a, b, c, z \in \mathbb{Z}_p$ be chosen at random. The DBDH assumption [7] in the bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ is that no probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$ can distinguish the tuple $(g^a, g^b, g^c, T = e(g, g)^{abc})$ from the tuple $(g^a, g^b, g^c, T = e(g, g)^z)$ with non-negligible advantage. The advantage of $\mathcal{A}$ is defined as

$$\left|\Pr[\mathcal{A}(g^a, g^b, g^c, T = e(g, g)^{abc}) = 1] - \Pr[\mathcal{A}(g^a, g^b, g^c, T = e(g, g)^z) = 1]\right|$$

where the probability is over the randomly chosen a, b, c, z and the random bits consumed by $\mathcal{A}$.

Decisional Truncated q-ABDHE Assumption Let $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ be a bilinear group system and $g' \in \mathbb{G}, \alpha, z \in \mathbb{Z}_p$ be chosen at random. The decisional truncated q-ABDHE assumption [13] in the bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ is that no PPT algorithm $\mathcal{A}$ can distinguish the tuple $(g', g'^{(\alpha^{q+2})}, g, g^\alpha, g^{(\alpha^2)}, \ldots, g^{(\alpha^q)}, Z = e(g, g')^{(\alpha^{q+1})})$ from the tuple $(g', g'^{(\alpha^{q+2})}, g, g^\alpha, g^{(\alpha^2)}, \ldots, g^{(\alpha^q)}, Z = e(g, g')^z)$ with non-negligible advantage. The advantage of $\mathcal{A}$ is defined as

$$\Big|\Pr[\mathcal{A}(g', g'^{(\alpha^{q+2})}, g, g^\alpha, g^{(\alpha^2)}, \ldots, g^{(\alpha^q)}, Z = e(g, g')^{(\alpha^{q+1})}) = 1] - \Pr[\mathcal{A}(g', g'^{(\alpha^{q+2})}, g, g^\alpha, g^{(\alpha^2)}, \ldots, g^{(\alpha^q)}, Z = e(g, g')^z) = 1]\Big|$$

where the probability is taken over the random bits consumed by $\mathcal{A}$, the random choices of $g' \in \mathbb{G}$ and $\alpha, z \in \mathbb{Z}_p$.

Modified DDH-1 Assumption. Let $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ be a bilinear group system and $a, b, z \in \mathbb{Z}_p$ be chosen at random. The modified DDH-1 assumption in the bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ is that no PPT algorithm $\mathcal{A}$ can distinguish the tuple $(g^a, e(g, g)^b, T = e(g, g)^{ab})$ from the tuple $(g^a, e(g, g)^b, T = e(g, g)^z)$ with non-negligible advantage. The advantage of $\mathcal{A}$ is defined as

$$\left|\Pr[\mathcal{A}(g^a, e(g, g)^b, T = e(g, g)^{ab}) = 1] - \Pr[\mathcal{A}(g^a, e(g, g)^b, T = e(g, g)^z) = 1]\right|$$

where the probability is over the randomly chosen a, b, z and the random bits consumed by $\mathcal{A}$.

Modified DDH-2 Assumption. Let $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ be a bilinear group system and $a, b, z \in \mathbb{Z}_p$ be chosen at random. The modified DDH-2 assumption in the bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ is that no PPT algorithm $\mathcal{A}$ can distinguish the tuple $(g^a, g^{1/a}, e(g,g)^b, T = e(g,g)^{b/a})$ from the tuple $(g^a, g^{1/a}, e(g,g)^b, T = e(g,g)^z)$ with non-negligible advantage. The advantage of $\mathcal{A}$ is defined as

$$\Big|\Pr[\mathcal{A}(g^a, g^{1/a}, e(g,g)^b, T = e(g,g)^{b/a}) = 1] - \Pr[\mathcal{A}(g^a, g^{1/a}, e(g,g)^b, T = e(g,g)^z) = 1]\Big|$$

where the probability is over the randomly chosen a, b, z and the random bits consumed by $\mathcal{A}$.

Modified DDH-1 and DDH-2 assumptions can be viewed as two variants of the traditional DDH assumption in $\mathbb{G}_T$. It is easy to show that these assumptions hold in the generic group model by applying the theorems of Katz et al. [23].

2.3 Zero-Knowledge Protocol and Proof of Knowledge Protocol

Informal, a zero-knowledge (ZK) protocol (or ZK proof) enables a prover to prove to a verifier that a statement is true, without revealing anything other than the veracity of the statement [17,16]. Let $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ be a bilinear group system. We use the term

$$\mathsf{ZK}\{(a, h_2) : A = e(g,g)^a \ \wedge \ e(h_1, h_2) = A \cdot B\}$$

to denote the ZK protocol, on common input $((p, \mathbb{G}, \mathbb{G}_T, e, g), h_1, A, B)$ for the statement that there exist a and h_2 such that $A = e(g,g)^a$ and $e(h_1, h_2) = A \cdot B$. Efficient ZK protocols for bilinear groups are known (see [21]). A protocol is *zero-knowledge* if there exists a simulator which is able to simulate the view of any (possibly malicious) PPT verifier in the protocol from scratch (i.e., without being given the witness as input).

We also require a zero-knowledge proof of knowledge (ZK-PoK) of discrete log protocol that enables a prover to prove to a verifier that it possesses the discrete log of a given group element in question. Efficient ZK-PoK of discrete log protocols can be found in [29]. A ZK-PoK protocol has the *proof of knowledge* property besides the zero-knowledge property. The proof of knowledge property implies the existence of a knowledge-extractor which interacts with the prover and extracts the witness using rewinding techniques [4,5].

3 Formal Definition and Security Model

We extend the original definition of A-IBE [18] to account for public traceability. An A-IBE with public traceability consists of the following six algorithms:

- Setup. This randomized algorithm takes as input the security parameter λ and outputs the public parameters PK and a master secret key MSK.

KeyGen. This is an interactive protocol between the public parameter generator PKG and a user U. The common input to PKG and U are the public parameters PK and the identity ID of U. The private input to PKG is the master secret key MSK. Additionally, PKG and U may use a sequence of random coin tosses as private input. At the end of the protocol, U receives a decryption key d_{ID} as its private output; PKG gets a tracing key t_{ID} for ID and adds it to a public tracing key list $\mathcal{TK}$.

Encrypt. This randomized algorithm takes as inputs the public parameters PK, an identity ID and a message M and outputs a ciphertext C.

Decrypt. This algorithm takes as inputs the public parameters PK, the ciphertext C that was encrypted under the identity ID and the decryption key d_{ID} for ID. It outputs a message M.

Trace. This is a randomized algorithm that takes as inputs the public parameters PK, an identity ID, the public tracing key list $\mathcal{TK}$, and has black-box access to an ϵ-useful decryption box $\mathbb{D}_{\mathsf{ID}}$[1] for the identity ID. It outputs PKG or User.

Judge. This is an interactive protocol between a user U and a judge. The common input to U and the judge are the public parameters PK, the identity ID of U and the tracing key t_{ID} for ID in $\mathcal{TK}$. The private input to U is his/her decryption key d_{ID}. Additionally, U and the judge may use a sequence of random coin tosses as private input. At the end of the protocol, the judge is able to decide whether t_{ID} is the tracing key for ID or not.

In our definition, the duty of the PKG includes maintaining a public tracing key list $\mathcal{TK}$ besides helping users get their decryption keys. Now, the tracing algorithm uses the public tracing key list $\mathcal{TK}$ (not the user's decryption key) to trace the source of a decryption box for a user.

Correctness. Correctness requires that, firstly, for any outputs (PK, MSK) of Setup, any message M and any identity ID, whenever d_{ID} is the user ID's private output of KeyGen, we have $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{Encrypt}(\mathsf{PK}, \mathsf{ID}, M), d_{\mathsf{ID}}) = M$. Secondly, if the public tracing key t_{ID} for ID in $\mathcal{TK}$ is modified by the PKG intentionally or otherwise, the user U with ID can use the algorithm Judge to convince a judge that his public tracing key has indeed been tampered with.

Zero-Knowledge of Judge Algorithm. One can design a trivial Judge algorithm in which, the user sends his/her decryption key to the judge directly and the judge can use the decryption key to decide whether the user's public tracing key has indeed been tampered with or not. However, since the goal of A-IBE with public traceability is to avoid exposing the user's decryption key to a third party, this trivial design is clearly unacceptable. Zero-knowledge of Judge algorithm requires that, at the end of the Judge algorithm, the judge cannot obtain any additional information about the user's decryption key except that whether the t_{ID} in $\mathcal{TK}$ is the tracing key for the user or not.

[1] For non-negligible ϵ, a PPT algorithm $\mathbb{D}_{\mathsf{ID}}$ is an ϵ-useful decoder box for the identity ID if for any message M, $\Pr[\mathbb{D}_{\mathsf{ID}}(\mathsf{Encrypt}(\mathsf{PK}, \mathsf{ID}, M)) = M] \geq \epsilon$.

Security. An A-IBE scheme is deemed secure if it satisfies the following three requirements. First, it must satisfy the standard security notion for IBE schemes: IND-ID-CPA. Second, it is intractable for the PKG to create a decryption box $\mathbb{D}_{\mathsf{ID}}$ such that the tracing algorithm outputs User. Finally, it is infeasible for users to create a decryption box such that the tracing algorithm implicates the PKG. Goyal et al. [19] captured these security requirements by three games: the IND-ID-CPA game, the DishonestPKG game and the DishonestUser game. Since in our A-IBE definition, the tracing key for a user could contain partial information of the user's decryption key and is publicly available, we make a critical enhancement of the three games to account for public traceability.

The IND-ID-CPA game for A-IBE with public traceability is defined as follows.

Setup. The challenger runs the Setup algorithm of A-IBE to obtain the public parameters PK and a master secret key MSK. It gives the public parameters PK to the adversary $\mathcal{A}$ and keeps MSK to itself.

Phase 1. The adversary $\mathcal{A}$ runs the KeyGen protocol with the challenger for adaptively chosen distinct identities $\mathsf{ID}_1, \ldots, \mathsf{ID}_{q_{\mathsf{ID}}}$, and receives the decryption keys $d_{\mathsf{ID}_1}, \ldots, d_{\mathsf{ID}_{q_{\mathsf{ID}}}}$. The challenger gets the tracing keys $t_{\mathsf{ID}_1}, \ldots, t_{\mathsf{ID}_{q_{\mathsf{ID}}}}$ and adds them to a public tracing key list $\mathcal{TK}$.

Output a challenge identity. The adversary $\mathcal{A}$ outputs an identity ID^* as his challenge identity which it did not query during Phase 1. The challenger sends the public tracing key t_{ID^*} for the identity ID^* to $\mathcal{A}$.

Phase 2. This is identical to Phase 1 except that the adversary is not allowed to ask for a decryption key for ID^*.

Challenge. The adversary $\mathcal{A}$ submits two (equal length) messages M_0, M_1. The challenger flips a random coin β, sets $C^* = \mathsf{Encrypt}(\mathsf{PK}, \mathsf{ID}^*, M_\beta)$ and sends C^* to the adversary as its challenge ciphertext.

Phase 3. This is identical to Phase 1 except that the adversary is not allowed to ask for a decryption key for ID^*.

Guess. The adversary $\mathcal{A}$ outputs its guess β' for β and wins the game if $\beta = \beta'$.

The advantage of the adversary in this game is defined as $|\mathsf{Pr}[\beta = \beta'] - \frac{1}{2}|$ where the probability is taken over the random bits used by the challenger and the adversary.

Definition 1. *An A-IBE scheme with public traceability is IND-ID-CPA secure if all probabilistic polynomial time adversaries have at most a negligible advantage in the IND-ID-CPA game.*

The DishonestPKG game for A-IBE with public traceability is defined as follows. The intuition behind this game is that an adversarial PKG attempts to create a decryption box which will frame a user.

Setup. The adversary (acting as a malicious PKG) generates public parameters PK. Then, it sends the public parameters PK and an identity ID^* to the challenger. The challenger checks that PK and ID^* are well-formed and aborts if the check fails.

Key Generation. The challenger and the adversary engage in the key generation protocol KeyGen to generate a decryption key for the identity ID^*. If neither party aborts, the challenger receives the decryption keys d_{ID^*}; and the adversary gets the tracing keys t_{ID^*} for ID^*.

Output. The adversary $\mathcal{A}$ outputs an ϵ-useful decryption box $\mathbb{D}_{\mathsf{ID}^*}$ for the identity ID^*, and succeeds if $\mathsf{Trace}^{\mathbb{D}_{\mathsf{ID}^*}}(\mathsf{PK}, \mathsf{ID}^*, \mathcal{TK} = \{t_{\mathsf{ID}^*}\}) = \mathsf{User}$.

The advantage of the adversary in this game is defined as $|\mathsf{Pr}[\mathcal{A}\ succeeds]|$ where the probability is taken over the random coins of Trace, the adversary and the challenger.

Definition 2. *An A-IBE scheme with public traceability is weak black-box DishonestPKG secure if all probabilistic polynomial time adversaries have at most a negligible advantage in the above game.*

One can define a full black-box $\mathsf{DishonestPKG}$ game where the adversary adaptively queries the challenger with a sequence of ciphertexts $C_1, \ldots, C_{q_C}$. The challenger decrypts the ciphertexts with its key d_{ID^*} and sends the results to the adversary.

The $\mathsf{DishonestUser}$ game for A-IBE with public traceability is defined as follows. The intuition behind this game is that some colluding set of users $\mathsf{ID}_1, \ldots, \mathsf{ID}_{q_{\mathsf{ID}}}$ attempts to create a decryption box which will frame the PKG.

Setup. The challenger runs the Setup algorithm of A-IBE to obtain the public parameters PK and a master secret key MSK. It gives the public parameters PK to the adversary $\mathcal{A}$ and keeps MSK to itself.

Phase 1. The adversary $\mathcal{A}$ runs the KeyGen protocol with the challenger for adaptively chosen distinct identities $\mathsf{ID}_1, \ldots, \mathsf{ID}_{q_{\mathsf{ID}}}$, and receives the decryption keys $d_{\mathsf{ID}_1}, \ldots, d_{\mathsf{ID}_{q_{\mathsf{ID}}}}$. The challenger gets the tracing keys $t_{\mathsf{ID}_1}, \ldots, t_{\mathsf{ID}_{q_{\mathsf{ID}}}}$ and adds them to a public tracing key list $\mathcal{TK}$.

Output a decryption box. The adversary $\mathcal{A}$ outputs an ϵ-useful decryption box $\mathbb{D}_{\mathsf{ID}^*}$ for an identity ID^* queried during Phase 1, and succeeds if $\mathsf{Trace}^{\mathbb{D}_{\mathsf{ID}^*}}$ $(\mathsf{PK}, \mathsf{ID}^*, \mathcal{TK}) = \mathsf{PKG}$.

The requirement that ID^* was queried during Phase 1 does not weaken security, since outputting a key for an identity which has not been queried would contradict the $\mathsf{IND\text{-}ID\text{-}CPA}$ security requirement. The advantage of the adversary in this game is defined as $|\mathsf{Pr}[\mathcal{A}\ succeeds]|$ where the probability is taken over the random coins of Trace, the adversary and the challenger.

Definition 3. *An A-IBE scheme with public traceability is adaptive-ID DishonestUser secure if all probabilistic polynomial time adversaries have at most a negligible advantage in the above game.*

We note that one can also define a selective-ID $\mathsf{DishonestUser}$ game where the adversary has to declare ID^* in advance (i.e., before the setup phase).

4 Our Construction

In [25], Libert and Vergnaud proposed a weak black-box tracing mechanism in their A-IBE scheme which is close to the one used by Kiayias and Yung [24] in

2-user traitor tracing schemes. In this tracing algorithm, the decryption box $\mathbb{D}$ for a user with identity ID is fed with n ill-formed ciphertexts for n random messages and ID. Given an ill-formed ciphertext C for a random message M and ID, $\mathbb{D}$ outputs M' such that $M' \neq M$ with overwhelming probability if $\mathbb{D}$ is generated by the PKG; otherwise, $M' = M$ with overwhelming probability. However, computing the ill-formed ciphertext for a message and ID needs to use the user's decryption key d_{ID}; thus their A-IBE scheme does not have the property of public traceability. We observe that the ill-formed ciphertext in [25] can be computed only using partial information of d_{ID}, which can be published as a tracing key for the user ID. Thus, we obtain a public weak black-box tracing mechanism. Based on Goyal-1 and the public weak black-box tracing mechanism, we propose a new A-IBE scheme. In our scheme, we also make a important modification in the key generation protocol of Goyal-1 in order to achieve DishonestPKG security.

Our proposed A-IBE scheme with public traceability consists of the following algorithms:

Setup. Given a security parameter λ, the PKG runs $\mathcal{G}(1^\lambda)$ to obtain a bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. It chooses $\alpha, \beta \in \mathbb{Z}_p$ uniformly at random. The public parameters are published as $\mathsf{PK} = (g, g_1 = g^\alpha, h = g^\beta)$. The master secret key is $\mathsf{MSK} = (\alpha, \beta)$.

KeyGen. The PKG and a user U with identity ID interact in the following key generation protocol. (We assume that the identity ID is an element in $\mathbb{Z}_p^*$. One can easily to extend the construction to arbitrary identities in $\{0,1\}^*$ by using a collision-resistant hash $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$.)

1. U first chooses $r \in \mathbb{Z}_p$ uniformly at random. Next, U sends $R = g^r$ to the PKG and runs an interactive ZK-PoK of the discrete log of R with respect to g with the PKG.
2. PKG checks that the ZK-PoK is valid and aborts if the check fails. Otherwise, the PKG picks a random $r' \in \mathbb{Z}_p$ and sends $(R', (r', h'))$ to U, where
$$R' = R^\beta = h^r, \; h' = (R' \cdot g^{-r'})^{1/(\alpha - \mathsf{ID})}.$$
3. U checks whether the following equalities hold or not:
$$e(g, R') = e(R, h) \text{ and } e(g_1 \cdot g^{-\mathsf{ID}}, h') = e(g, R' g^{-r'}).$$
If no, U aborts. Otherwise, U sets his decryption key $d_{\mathsf{ID}} = (r_{\mathsf{ID}}, h_{\mathsf{ID}})$, where $r_{\mathsf{ID}} = r'/r$ and $h_{\mathsf{ID}} = h'^{1/r} = (h^r \cdot g^{-r'})^{1/r(\alpha - \mathsf{ID})} = (h g^{-r_{\mathsf{ID}}})^{1/(\alpha - \mathsf{ID})}$.
4. U sends $R_{\mathsf{ID}} = e(g, g)^{r_{\mathsf{ID}}}$ to the PKG and runs an interactive zero-knowledge proof $\mathsf{ZK}\{(r_{\mathsf{ID}}, h_{\mathsf{ID}}) : R_{\mathsf{ID}} = e(g, g)^{r_{\mathsf{ID}}} \;\wedge\; e(g_1 \cdot g^{-\mathsf{ID}}, h_{\mathsf{ID}}) = e(g, h) \cdot R_{\mathsf{ID}}^{-1}\}$ with the PKG.
5. PKG checks that the ZK proof is valid and aborts if the check fails. Otherwise, PKG adds the tracing key $t_{\mathsf{ID}} = (\mathsf{ID}, R_{\mathsf{ID}})$ for the identity ID to a public tracing key list $\mathcal{TK}$.

Encrypt. Given the public parameters PK, an identity ID and a message $M \in \mathbb{G}_T$, the encryption algorithm chooses random $s \in \mathbb{Z}_p$ and computes the ciphertext $C = (C_1, C_2, C_3)$, where
$$C_1 = g_1^s g^{-s \cdot \mathsf{ID}}, \; C_2 = e(g, g)^s, \; C_3 = M \cdot e(g, h)^{-s}.$$

Decrypt. Given the public parameters PK, a ciphertext $C = (C_1, C_2, C_3)$ and a decryption key $d_{\mathsf{ID}} = (r_{\mathsf{ID}}, h_{\mathsf{ID}})$ for ID, the decryption algorithm outputs $M = e(C_1, h_{\mathsf{ID}}) \cdot C_2^{r_{\mathsf{ID}}} \cdot C_3$.

Trace. Given the public parameters PK, an identity ID, the public tracing key list $\mathcal{TK}$ and an ϵ-useful decryption box $\mathbb{D}_{\mathsf{ID}}$ for the identity ID, the tracing algorithm first finds the tuple $(\mathsf{ID}, R_{\mathsf{ID}})$ in the public tracing key list $\mathcal{TK}$. If such tuple does not exist, it aborts. Otherwise, it performs the following steps.

1. Set a counter $ctr = 0$ and repeat the following experiment $n = 8\lambda/\epsilon$ times:
   - Pick two *distinct* elements $s, s' \in \mathbb{Z}_p$ at random.
   - Choose a random message $M \in \mathbb{G}_T$ and compute $C_1 = g_1^s g^{-s \cdot \mathsf{ID}}$, $C_2 = e(g, g)^{s'}$, $C_3 = M \cdot e(g, h)^{-s} \cdot R_{\mathsf{ID}}^{(s-s')}$.
   - Feed the ϵ-useful decryption box $\mathbb{D}_{\mathsf{ID}}$ with the ciphertext $C = (C_1, C_2, C_3)$. The decryption box $\mathbb{D}_{\mathsf{ID}}$ outputs a message M'. If $M' = M$, increment ctr.
2. If $ctr = 0$, output PKG. Otherwise, output User.

Judge A user U with identity ID and a judge interact in the following protocol.

1. Let $d_{\mathsf{ID}} = (r_{\mathsf{ID}}, h_{\mathsf{ID}})$ be the user's decryption key. U first sets $R_{\mathsf{ID}} = e(g, g)^{r_{\mathsf{ID}}}$. Next, U sends R_{ID} to the judge and runs an interactive zero-knowledge proof $\mathsf{ZK}\{(r_{\mathsf{ID}}, h_{\mathsf{ID}}) : R_{\mathsf{ID}} = e(g, g)^{r_{\mathsf{ID}}} \ \wedge \ e(g_1 \cdot g^{-\mathsf{ID}}, h_{\mathsf{ID}}) = e(g, h) \cdot R_{\mathsf{ID}}^{-1}\}$ with the judge.
2. The judge checks that the ZK proof is valid and aborts if the check fails. Otherwise, the judge accepts $t_{\mathsf{ID}} = (\mathsf{ID}, R_{\mathsf{ID}})$ as the user's public tracing key.

Correctness. If the ciphertext $C = (C_1, C_2, C_3)$ is well-formed for ID, then

$$\begin{aligned} e(C_1, h_{\mathsf{ID}}) \cdot C_2^{r_{\mathsf{ID}}} \cdot C_3 &= e(g^{s \cdot (\alpha - \mathsf{ID})}, (hg^{-r_{\mathsf{ID}}})^{1/(\alpha - \mathsf{ID})}) \cdot e(g, g)^{s \cdot r_{\mathsf{ID}}} \cdot C_3 \\ &= e(g, h)^s \cdot C_3 = e(g, h)^s \cdot M \cdot e(g, h)^{-s} = M. \end{aligned}$$

Zero-knowledge of Judge *algorithm.* Obviously, because of the zero-knowledge property of the ZK proof, at the end of the Judge algorithm, the judge cannot obtain any additional information about the user's decryption d_{ID} except the tracing key itself.

4.1 Discussion

In the key generation protocol of Goyal-1, a user first sends h^r to the PKG; but in our scheme, the user sends g^r to the PKG, which enables us to prove that our A-IBE with public traceability is weak black-box DishonestPKG secure.

The IND-ID-CPA security and DishonestUser security of our scheme require a strong q-ABDHE assumption inherited from Gentry's IBE scheme [13]. However, combining the A-IBE scheme proposed by Libert and Vergnaud [25] with our public tracing mechanism, we can obtain a new A-IBE scheme with public traceability, in which the IND-ID-CPA security and DishonestUser security require the DBDH assumption, instead of the strong q-ABDHE assumption.

CCA-security can be obtained by applying the Canetti-Halevi-Katz transformation [10] to a two receiver variant of the Gentry-Waters identity-based broadcast encryption (IBBE) scheme [15] that does not affect our tracing algorithm.

4.2 IND-ID-CPA Security

Theorem 1. *The above A-IBE scheme with public traceability is IND-ID-CPA secure under the decisional truncated q-ABDHE and modified DDH-1 assumption in the bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e, g)$.*

Proof. The proofs will be given in the full version of the paper due to the space limitation.

4.3 DishonestPKG Security

Theorem 2. *The above A-IBE scheme with public traceability is weak black-box DishonestPKG secure under the modified DDH-2 assumption in the bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e, g)$.*

Proof. To prove the weak black-box DishonestPKG security of our A-IBE scheme, we consider the following two games.

Game$_0$. The original weak black-box DishonestPKG game. Note that, in the end of the original weak black-box DishonestPKG game, the adversary outputs an ϵ-useful decryption box $\mathbb{D}_{\mathsf{ID}^*}$ for an identity ID^*; and in the tracing algorithm, $\mathbb{D}_{\mathsf{ID}^*}$ is fed with n ill-formed ciphertexts for n random messages and the identity ID^*. The ill-formed ciphertext $C = (C_1, C_2, C_3)$ for a random message M and the identity ID^* is set as $C_1 = g_1^s g^{-s \cdot \mathsf{ID}^*}$, $C_2 = e(g, g)^{s'}$, $C_3 = M \cdot e(g, h)^{-s} \cdot R_{\mathsf{ID}^*}^{(s-s')}$, where $t_{\mathsf{ID}^*} = (\mathsf{ID}^*, R_{\mathsf{ID}^*})$ is the tracing key for ID^*, $s, s' \xleftarrow{\$} \mathbb{Z}_p$ and $s \neq s'$.

Game$_1$. The same as Game$_0$ except that in the tracing algorithm, the ill-formed ciphertext $C = (C_1, C_2, C_3)$ for a random message M and the identity ID^* is set as $C_1 = g_1^s g^{-s \cdot \mathsf{ID}^*}$, $C_2 = e(g, g)^{s'}$, $C_3 = M \cdot e(g, h)^{-s} \cdot e(g, g)^r$, where $s, s', r \xleftarrow{\$} \mathbb{Z}_p$ and $s \neq s'$.

We prove this theorem by the following two lemmas. Lemma 1 states that Game$_0$ and Game$_1$ are indistinguishable; and Lemma 2 states that the advantage of the adversary in Game$_1$ is negligible. Therefore, we conclude that the advantage of the adversary in Game$_0$ (i.e., the original weak black-box DishonestPKG game) is negligible. This completes the proof of Theorem 2.

Lemma 1. *Game$_0$ and Game$_1$ are computationally indistinguishable under the modified DDH-2 assumption in the bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e, g)$.*

Proof. Suppose there exists an adversary $\mathcal{A}$ who can distinguish between Game_0 and Game_1 with non-negligible probability. Then we can build an algorithm $\mathcal{B}$ that makes use of $\mathcal{A}$ to break the modified DDH-2 assumption in the bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e, g)$.

$\mathcal{B}$ receives $(g^a, g^{1/a}, e(g,g)^b, Z)$ and tries to decide if $Z = e(g,g)^{b/a}$. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine and proceeds as follows.

Setup. The adversary $\mathcal{A}$ sends the public parameters $\mathsf{PK} = (g, g_1, h)$ and an identity ID^* to $\mathcal{B}$. $\mathcal{B}$ checks that g, g_1, h are elements in $\mathbb{G}$ and ID^* is a element in $\mathbb{Z}_p^*$, and aborts if the check fails.

Key Generation. $\mathcal{B}$ engages in the interactive key generation protocol with $\mathcal{A}$ to get a decryption key for the challenge identity ID^* as follows. It sends $R = g^a$ and now has to give a ZK-PoK of the discrete log of R with respect to g to $\mathcal{A}$. Note that, the zero-knowledge property of the proof system implies the existence of a simulator which is able to successfully simulate the view of $\mathcal{A}$ in the protocol (by rewinding $\mathcal{A}$) with overwhelming probability; and $\mathcal{B}$ can use the simulator to simulate the required proof even without knowledge of a. Then, $\mathcal{B}$ receives $(R', (r', h'))$ from $\mathcal{A}$. $\mathcal{B}$ checks if $e(R, h) \overset{?}{=} e(R', g)$ and $e(g_1 g^{-\mathsf{ID}^*}, h') \overset{?}{=} e(g, R' g^{-r'})$. If the test fails, $\mathcal{B}$ aborts. Otherwise, the decryption key d_{ID^*} for ID^* is defined (but unknown to $\mathcal{B}$) as $(r_{\mathsf{ID}^*} = r'/a, h_{\mathsf{ID}^*} = h'/a)$. Next, $\mathcal{B}$ sends $R_{\mathsf{ID}^*} = e(g,g)^{r_{\mathsf{ID}^*}} = e(g,g)^{r'/a} = e(g^{1/a}, g^{r'})$ to $\mathcal{A}$ and now has to give a zero-knowledge proof $\mathsf{ZK}\{(r_{\mathsf{ID}^*}, h_{\mathsf{ID}^*}) : R_{\mathsf{ID}^*} = e(g,g)^{r_{\mathsf{ID}^*}} \ \wedge \ e(g_1 \cdot g^{-\mathsf{ID}^*}, h_{\mathsf{ID}^*}) = e(g,h) \cdot R_{\mathsf{ID}^*}^{-1}\}$. Similarly, $\mathcal{B}$ can use the simulator of the zero-knowledge proof to simulate the required proof even without knowledge of $(r_{\mathsf{ID}^*}, h_{\mathsf{ID}^*})$. Finally, $\mathcal{A}$ adds the tracing key $t_{\mathsf{ID}^*} = (\mathsf{ID}^*, R_{\mathsf{ID}^*})$ for the identity ID^* to a public tracing key list $\mathcal{TK}$.

Output. At this point, the adversary $\mathcal{A}$ outputs an ϵ-useful decryption box $\mathbb{D}_{\mathsf{ID}^*}$ for the identity ID^*. In the tracing stage, $\mathcal{B}$ proceeds as follows.

1. Choose n messages $M_1, \ldots, M_n \in \mathbb{G}_T$ uniformly at random. For $i = 1$ to n, compute the ciphertext $C^{(i)} = (C_1^{(i)}, C_2^{(i)}, C_3^{(i)})$ as
$$C_1^{(i)} = g^{s_i \cdot (\alpha - \mathsf{ID}^*)}, \ C_2^{(i)} = \left(e(g,g)^b\right)^{\delta_i} \cdot e(g,g)^{a_i},$$
$$C_3^{(i)} = M_i \cdot e(g,h)^{-s_i} \cdot R_{\mathsf{ID}^*}^{s_i - a_i} \cdot Z^{-r' \cdot \delta_i},$$
where $s_i, \delta_i, a_i \overset{\$}{\leftarrow} \mathbb{Z}_p$.
2. $\mathbb{D}_{\mathsf{ID}^*}$ is fed with n ciphertexts $C^{(1)}, \ldots, C^{(n)}$.

Observe that, if $Z = e(g,g)^{b/a}$, then $Z^{-r' \cdot \delta_i} = e(g,g)^{-(b/a) \cdot r' \cdot \delta_i} = \left(e(g,g)^{r'/a}\right)^{-b \cdot \delta_i} = R_{\mathsf{ID}^*}^{-b \cdot \delta_i}$,
$$C_1^{(i)} = g^{s_i \cdot (\alpha - \mathsf{ID}^*)}, \ C_2^{(i)} = e(g,g)^{s_i'}, \ C_3^{(i)} = M_i \cdot e(g,h)^{-s_i} \cdot R_{\mathsf{ID}^*}^{s_i - s_i'},$$
where $s_i' = b \cdot \delta_i + a_i$ is uniformly random; $\mathcal{B}$ has properly simulated Game_0. If Z is a random element in $\mathbb{G}_T$, $\mathcal{B}$ has properly simulated Game_1. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ (i.e., the ϵ-useful decryption box $\mathbb{D}_{\mathsf{ID}^*}$) to distinguish between these possibilities for Z.

Lemma 2. *In Game_1, the advantage of the adversary is negligible.*

Proof. In the end of Game_1, the adversary $\mathcal{A}$ outputs an ϵ-useful decryption box $\mathbb{D}_{\mathsf{ID}^*}$ for an identity ID^*. The advantage of the adversary is the probability that the tracing algorithm outputs User.

The tracing algorithm points to the User if it ends up with $ctr \neq 0$. In an iteration of the tracing stage of Game_1, $\mathbb{D}_{\mathsf{ID}^*}$ is given a ciphertext $C = (C_1, C_2, C_3)$ for a random message M and the identity ID^*, where

$$C_1 = g_1^s g^{-s \cdot \mathsf{ID}^*},\ C_2 = e(g,g)^{s'},\ C_3 = M \cdot e(g,h)^{-s} \cdot e(g,g)^r,$$

and $s, s', r \overset{\$}{\leftarrow} \mathbb{Z}_p$ and $s \neq s'$; $\mathbb{D}_{\mathsf{ID}^*}$ returns M' such that $M' = M$ with the probability $1/p$ since s, s', r are uniformly random and independent from the adversary's view. Therefore, we conclude that the advantage of the adversary in Game_1 is negligible since $\Pr[ctr \neq 0] = \Pr[ctr \geq 1] \leq n/p = 8\lambda/(\epsilon p) \leq 8\lambda/(2^\lambda \epsilon)$.

As noted in [26], a decoder box $\mathbb{D}_{\mathsf{ID}^*}$ for ID^* generated by the malicious PKG is able to recognize invalid ciphertexts in the tracing stage (as it may contain the master secret key MSK). However, as long as $\mathbb{D}_{\mathsf{ID}^*}$ is assumed stateless, it cannot shutdown or self-destruct when detecting a tracing attempt. Moreover, with overwhelming probability, $\mathbb{D}_{\mathsf{ID}^*}$ will never be able to decrypt such invalid ciphertexts in the same way as the owner of d_{ID^*} would.

In the proof of Lemma 1, the decryption key d_{ID^*} for ID^* is unknown to the challenger $\mathcal{B}$; thus it is not able to provide the decryption oracle to the adversary (i.e., the malicious PKG). Hence, we can only prove that our scheme is weak black-box DishonestPKG secure. One future direction is to construct A-IBE schemes with public and full black-box traceability.

4.4 DishonestUser Security

Theorem 3. *The above A-IBE scheme with public traceability is adaptive-ID DishonestUser secure under the decisional truncated q-ABDHE assumption in the bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e, g)$.*

Proof. The proofs will be given in the full version of the paper due to the space limitation.

5 Conclusions

In this paper, we studied the public traceability problem of A-IBE. We first extended the original A-IBE model [18] to account for public traceability. Then, we presented the first A-IBE scheme with public traceability that combines the first A-IBE construction proposed by Goyal in [18] with a public tracing mechanism. A limitation of our scheme is that it only allows for weak black-box traceability. It remains an open problem to construct an A-IBE scheme with public traceability while has the properties of IND-ID-CPA security, black-box DishonestPKG security and adaptive-ID DishonestUser security.

Acknowledgement. The authors thank the anonymous reviewers for their helpful comments. The first and second authors are supported by the Office of Research, Singapore Management University. The third author is supported by National Science Foundation of China (No. 61070248, 61272012) and Innovation Project (No.12ZZ013) of Shanghai Municipal Education Commission. The fourth author is supported by the National Science Foundation of China under Grant Nos. 60903178, 61005049, 61133014, 61272413 and 61272415, the Fok Ying Tung Education Foundation under Grant No. 131066, the Fundamental Research Funds for the Central Universities under Grant No. 21610204, the Guangdong Provincial Science and Technology Project under Grand No. 2010A032000002, the Opening Project of Shanghai Key Laboratory of Integrate Administration Technologies for Information Security under Grand No. AGK201100,and the R&D Foundation of Shenzhen Basic Research Project under Grand No. JC201105170617A.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
2. Al-Riyami, S.S., Paterson, K.G.: Certificateless Public Key Cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
3. Au, M.H., Huang, Q., Liu, J.K., Susilo, W., Wong, D.S., Yang, G.: Traceable and Retrievable Identity-Based Encryption. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 94–110. Springer, Heidelberg (2008)
4. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
5. Bellare, M., Goldreich, O.: On Probabilistic versus Deterministic Provers in the Definition of Proofs of Knowledge. In: Goldreich, O. (ed.) Studies in Complexity and Cryptography. LNCS, vol. 6650, pp. 114–123. Springer, Heidelberg (2011)
6. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
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)
8. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
9. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
10. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)

11. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: IMA Int. Conf., pp. 360–363 (2001)
12. Gentry, C.: Certificate-based Encryption and the Certificate Revocation Problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
13. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
14. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
15. Gentry, C., Waters, B.: Adaptive Security in Broadcast Encryption Systems (with Short Ciphertexts). In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 171–188. Springer, Heidelberg (2009)
16. Goldreich, O.: The Foundations of Cryptography. Basic Techniques, vol. 1. Cambridge University Press (2001)
17. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity for all languages in np have zero-knowledge proof systems. J. ACM 38(3), 691–729 (1991)
18. Goyal, V.: Reducing Trust in the PKG in Identity Based Cryptosystems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 430–447. Springer, Heidelberg (2007)
19. Goyal, V., Lu, S., Sahai, A., Waters, B.: Black-box accountable authority identity-based encryption. In: ACM Conference on Computer and Communications Security, pp. 427–436 (2008)
20. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
21. Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
22. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
23. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. Cryptology ePrint Archive, Report 2007/404 (2007), http://eprint.iacr.org/
24. Kiayias, A., Yung, M.: Traitor Tracing with Constant Transmission Rate. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 450–465. Springer, Heidelberg (2002)
25. Libert, B., Vergnaud, D.: Towards black-box accountable authority IBE with short ciphertexts and private keys. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 235–255. Springer, Heidelberg (2009)
26. Libert, B., Vergnaud, D.: Towards practical black-box accountable authority ibe: Weak black-box traceability with short ciphertexts and private keys. IEEE Transactions on Information Theory 57(10), 7189–7204 (2011)
27. Sahai, A., Seyalioglu, H.: Fully Secure Accountable-Authority Identity-Based Encryption. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 296–316. Springer, Heidelberg (2011)

28. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
29. Schnorr, C.-P.: Efficient Identification and Signatures for Smart Cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
30. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
31. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Efficient Delegation of Key Generation and Revocation Functionalities in Identity-Based Encryption

Jae Hong Seo and Keita Emura

National Institute of Information and Communications Technology, 4-2-1, Nukui-kitamachi, Koganei, Tokyo, 184-8795, Japan
{jaehong,k-emura}@nict.go.jp

Abstract. In the public key cryptosystems, revocation functionality is required when a secret key is corrupted by hacking or the period of a contract expires. In the public key infrastructure setting, numerous solutions have been proposed, and in the Identity Based Encryption (IBE) setting, a recent series of papers proposed revocable IBE schemes. Delegation of key generation is also an important functionality in cryptography from a practical standpoint since it allows reduction of excessive workload for a single key generation authority. Although efficient solutions for either revocation or delegation of key generation in IBE systems have been proposed, an important open problem is *efficiently* delegating both the key generation and revocation functionalities in IBE systems. Libert and Vergnaud, for instance, left this as an open problem in their CT-RSA 2009 paper. In this paper, we propose the first solution for this problem. We prove the selective-ID security of our proposal under the Decisional Bilinear Diffie-Hellman assumption in the standard model.

1 Introduction

Shamir introduced the concept of the Identity-based Encryption (IBE) scheme, a public key encryption scheme allowing any bit-string (e.g., e-mail address) to be a public key of a user that chooses such a bit-string [23]. Since Boneh and Franklin's first realization of IBE using bilinear pairings over elliptic curves, IBE systems have been applied in numerous applications. Several variations of IBE systems have also been proposed for adding other functionalities. In particular, the hierarchical identity-based encryption (HIBE) scheme allows the key generation center (KGC) to delegate the key generation functionality to users [11] and the revocable IBE (RIBE) scheme allows the KGC to efficiently revoke users for each time period [2].

Revocation Functionality in IBE. In public key cryptosystems, we need revocation functionality when a secret key is corrupted by hacking or the period of a contract expires. In the public key infrastructure setting, numerous solutions have been proposed, and in the IBE setting, a series of recent papers has proposed *scalable* RIBE schemes since Boldyreva et al. [2]. In fact, Boneh and Franklin [5] already proposed a trivial solution for revocation functionality, wherein new decryption keys are issued for each time period. However, their solution introduces huge overheads for the KGC that are linearly increased in the number of users. Boldyreva et al. and all subsequent works were aimed at constructing a scalable RIBE scheme, that is, the KGC's overhead increases logarithmically in the number of users. All proposed scalable RIBE schemes used the same methodology for revocation by using a binary tree structure. Each user ID is assigned to a leaf node ζ_{ID} of the binary tree structure and has keys corresponding to the nodes on the path between the assigned leaf node and the root node. By using the technique called the Complete Subtree (CS) method [20], which is widely accepted for broadcast encryption, the KGC broadcasts the key update for each time period (i.e., no secure channel is required in this phase) such that only non-revoked users can generate the decryption key for that time period from their secret key and the key update. For a non-revoked user, there is at least one subkey among the $\log N$ size key, where N is the maximum number of users. Since the CS method is secure against colluding and allows short key updates, the resulting RIBE scheme is well scalable and secure[1].

Delegation Functionality in IBE. For a large network, a single KGC has an excessive workload for performing computationally expensive key generation and establishing secure channels to transmit each user's secret key. To mitigate this problem, Horwitz and Lynn [14] introduced the concept of HIBE such that the responsibility for key generation is distributed to the lower-level KGC by delegating key generation functionality. Numerous constructions for HIBE schemes and variants with additional properties have subsequently been proposed [11,3,4,6,10,22,24,16].

Delegation of Both Key Generation and Revocation Functionalities in IBE. Although IBE schemes with either efficient revocation or efficient delegation for key generation functionality have been proposed, it is non-trivial to achieve both functionalities at the same time, and in fact Libert and Vergnaud left this as an open problem at CT-RSA 2009 [18]. We simply call such a scheme having both functionalities a Revocable HIBE (RHIBE) scheme. There are some difficulties in achieving RHIBE.

1. Trivial approaches will lead to exponentially large secret keys in the corresponding hierarchical level.
2. Key generations and key updates are recursively defined: this leads some difficulty in the security proof.

All existing scalable RIBE schemes utilize binary tree structures, that is the CS method, for revocation. In the scalable RIBE scheme using the CS method, a secret key of each user consists of $\log N$ subkeys, where N is the number of all

[1] The security model for the RIBE scheme is almost equal to that of the conventional IBE scheme. The only difference is that the adversary of the RIBE scheme is allowed to query for the challenge identity ID^*, but in this case the challenge identity should be revoked on the challenge time T^*.

users and at least one subkey of a non-revoked user ID can be used to generate a decryption key $\mathsf{dk}_{\mathsf{ID},\mathsf{T}}$ from the key update ku_{T} on a time period T. If we extend the RIBE scheme for the RHIBE scheme in a natural way, the second-level user has to have $(\log N)^2$ subkeys since one of the subkeys of the paraent's key can be used in each time period so that a child should have a $\log N$ subkey for each parent's subkey. In general, ℓ-level users have $(\log N)^\ell$ subkeys, so the size of the secret key exponentially grows in the corresponding hierarchical depth.

For constructing RHIBE, if we follow the same strategy used by all scalable RIBE schemes, KGC may not be able to directly generate secret keys of descendants (except for the first-level user). Each intermediate-level user's secret key is generated according to the shape of the binary tree structure, which is managed by its parent. However, the KGC does not know such a binary tree, so the KGC cannot create secret keys of intermediate-level users. (Note that the KGC can generate decryption keys for all descendants.) Therefore, the secret key and key updates have to be *recursively* defined. This makes the situation more complicated. In particular, in the security model the adversary can query secret keys of descendants of the challenge identity, but it is non-trivial to recursively generate such secret keys without knowing the challenge identity's secret key. We explain the detailed difficulty in Section 4.

Our Contribution. In this paper, we study a way to *efficiently* delegate both key generation and revocation functionalities in the IBE system. In particular, we propose the first realization of RHIBE. Our RHIBE construction is based on the Boneh-Boyen HIBE (BB-HIBE) scheme [3]. We carefully deal with the difficulties mentioned above. The ciphertext size of our RHIBE is almost the same as that of the BB-HIBE (only one additional group element is required) and the revocation cost of each user is the same as that of the Boldyreva et al. one. Nevertheless, our scheme enables hierarchical structures of identities. Moreover, a user (of a level ℓ) needs to manage $O(\ell^2 \log N)$-size secret key (polynomial in the hierarchical depth). Our RHIBE is selective-ID secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption in the standard model.

Related Work. Boldyreva et al. [2] proposed the first RIBE by applying the fuzzy IBE scheme [21]. Thanks to the collusion resistance of the underlying fuzzy IBE, no revoked users can compute a decryption key from their own secret key and publicly available key update information. Moreover, by applying the CS method [20], the Boldyreva et al. scheme is scalable in the sense that the costs of the KGC logarithmically depend on the number of users. Although their scheme is secure under a relatively weaker notion, called selective-ID security, Libert and Vergnaud [18] proposed adaptive-ID secure RIBE by applying a variant of the Waters IBE scheme [19]. In the proof of the Libert-Vergnaud RIBE scheme, the simulator can construct all the secret keys (as in the Gentry IBE [9]) to answer the secret key query for the challenge user. Recently, a RIBE scheme from lattices [7] was proposed.

In R(H)IBE, each decryption key is computed from the long-term secret key and key update information. Dodis et al. considered a similar functionality which we call key-insulated PKE [8,1,17] and IBE [13,12,25]. A user computes its decryption key from the long-term secret key and a helper key served in physically insulated storage. A difference between key-insulated PKE and RIBE is that the former requires a secure channel between a user and the storage for every key update, but the latter needs a secure channel only once for transmitting each user's secret key from its ancestor. Moreover, since each helper key is generated for each user, the total size of the helper key linearly depends on the number of total users, whereas it logarithmically depends on the number of total (i.e., non-revoked) users in RIBE.

Outline. This paper is organized as follows. The next section gives preliminaries. In Section 3, we define a syntax and a security model for the RHIBE scheme and we propose our RHIBE construction in Section 4. Lastly, we analyze the security of the proposed scheme and give a conclusion with interesting open problems.

2 Preliminaries

This section gives the definition of the bilinear groups, the DBDH assumption, and the KUNode algorithm [2].

Definition 1 (Bilinear Groups). *The bilinear group generator $\mathcal{G}(\cdot)$ is an algorithm that takes as input a security parameter λ and outputs a bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$, where p is a prime of size 2λ, $\mathbb{G}$ and $\mathbb{G}_t$ are cyclic groups of order p, and e is an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with*

- *Bilinearity : for all $u, u', v, v' \in \mathbb{G}$, $e(uu', v) = e(u, v)e(u', v)$ and $e(u, vv') = e(u, v)e(u, v')$,*
- *Non-degeneracy : for a generator g of $\mathbb{G}$, $e(g, g) \neq 1_{\mathbb{G}_t}$, where $1_{\mathbb{G}_t}$ is identity element in $\mathbb{G}_t$.*

Definition 2 (Decision Bilinear Diffie-Hellman (DBDH) Assumption). *Given a bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$ generated by $\mathcal{G}(\lambda)$, define two distributions $\mathcal{D}_0(\lambda) = (g, g^a, g^b, g^c, e(g, g)^{abc})$ and $\mathcal{D}_1(\lambda) = (g, g^a, g^b, g^c, e(g, g)^z)$, where $g \overset{\$}{\leftarrow} \mathbb{G}$ and $a, b, c, z \overset{\$}{\leftarrow} \mathbb{Z}_p$. The DBDH problem in the bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$ is to decide a bit b from given $\mathcal{D}_b$, where $b \overset{\$}{\leftarrow} \{0, 1\}$. The advantage of $\mathcal{A}$ in solving the DBDH problem in the bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$ is defined by*

$$Adv^{DBDH}_{\mathcal{G},\mathcal{A}}(\lambda) = \Big|\Pr[\mathcal{A}(\mathcal{D}_0(\lambda)) \to 1] - \Pr[\mathcal{A}(\mathcal{D}_1(\lambda)) \to 1]\Big|.$$

We say that the DBDH assumption holds in the bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$ if no Probabilistic Polynomial Time (PPT) algorithm has a non-negligible advantage in solving the DBDH problem in the bilinear group $(p, \mathbb{G}, \mathbb{G}_t, e)$.

Our RHIBE scheme is based on the BB-HIBE scheme [3], which is IND-sID-CPA secure under the DBDH assumption. The formal definition of IND-sID-CPA security and description of the BB-HIBE scheme are provided in the full

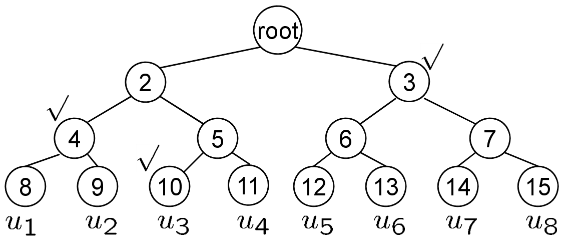


Fig. 1. Example of KUNode(BT, RL, T): We revoke the user u_4 (that is assigned to x_{11}). Then, all users, except u_4, have a node $x \in Y = \{x_3, x_4, x_{10}\}$ (we checked these nodes) that is contained in the set of nodes on the path from each user's assigned node to `root`.

version of this paper. The revocation method in each level of our construction follows Boldyreva et al.'s way using binary tree structure. The following KUNode algorithm is essentially used for revoking users in our construction.

Definition 3 (The KUNode Algorithm [2]). *This algorithm takes as input a binary tree* BT*, revocation list RL, and time T, and outputs a set of nodes. A formal description of this algorithm is as follows: If x is a non-leaf node, then x_{left} and x_{right} denote the left and right child of x, respectively. Each user is assigned to a leaf node. If a user (that is assigned to ζ) is revoked on time T, then $(\zeta, T) \in RL$.* Path(ζ) *denotes the set of nodes on the path from ζ to* `root`. *The description of* KUNode *is given below.*

```
KUNode(BT, $RL, T$) :
  X, Y $\leftarrow \emptyset$;
  $\forall(\zeta, T_i) \in RL$
      If $T_i \leq T$ then add Path($\zeta_i$) to X
  $\forall x \in$ X
      If $x_{left} \notin$ X then add $x_{left}$ to Y
      If $x_{right} \notin$ X then add $x_{right}$ to Y
  If Y $= \emptyset$ then add root to Y
  Return Y
```

We give a simple example of KUNode in the fugure 1.

3 Syntax and Security Model of RHIBE

We give formal definitions of the *hierarchical identity-based encryption with efficient revocation* scheme, which is simply called the Revocable Hierarchical Identity-Based Encryption (RHIBE) scheme, and its security by extending those of the *revocable IBE* in [2]. An RHIBE scheme consists of seven algorithms: Setup, KeyGen, KeyUp, DKG, Enc, Dec, and Revk. Roughly speaking, the Setup algorithm is run by the trusted authority called the Key Generation Center (KGC)

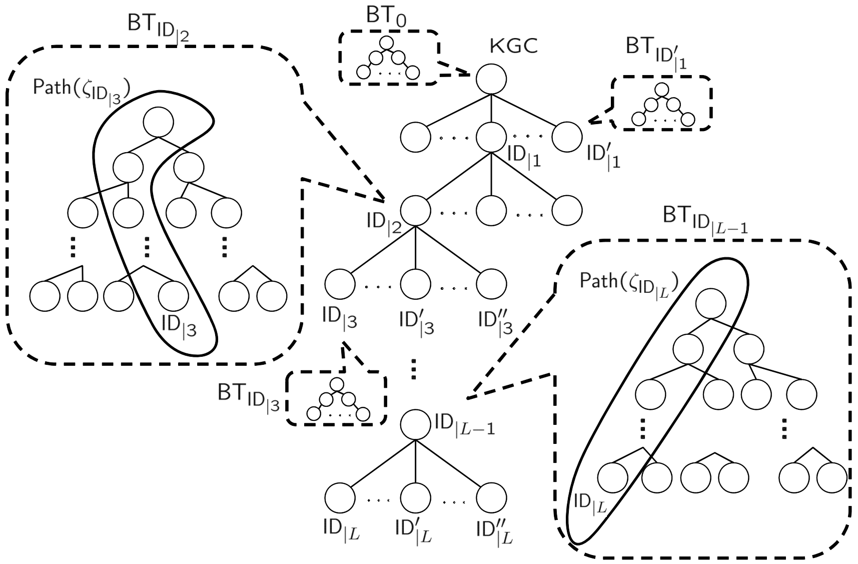


Fig. 2. The Hierarchical Structures: Each user (except the last level user) in the hierarchy of IBE system has his own binary tree for revocation functionality

for system parameters and a corresponding master secret key. When a user (possibly the KGC) generates a child's key, it can run the KeyGen algorithm with its secret key (the master secret key for the KGC) and a child's identity. If some users are revoked, the identities of revoked users should be updated along with each user's revocation time period. A $(\ell-1)$-th level user $\mathsf{ID}_{|\ell-1}$ issues key update information $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ at every time period T by running the KeyUp algorithm. This information is managed by the binary tree $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$ which is served in the state $\mathsf{st}_{\mathsf{ID}_{|\ell-1}}$. While performing revocation, senders do not need to be updated and can encrypt a message using the receiver's identity, a time period that the receiver can decrypt, and the Enc algorithm. Every user $\mathsf{ID}_{|\ell}$ that is not revoked on time T can create a decryption key $\mathsf{dk}_{\mathsf{ID}_{\ell},\mathsf{T}}$ from $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ by running the DKG algorithm, so that only a non-revoked user $\mathsf{ID}_{|\ell}$ (on time T) can decrypt an encrypted message for the identity $\mathsf{ID}_{|\ell}$ and the time period T by using the decryption key $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}$ and the Dec algorithm. Figure 2 explains the hierarchical structure with binary tree structures and we also provide the formal syntax of the RHIBE scheme below.

Definition 4. *An RHIBE scheme consists of seven algorithms:* Setup*,* KeyGen*,* KeyUp*,* DKG*,* Enc*,* Dec*, and* Revk*. The specification of each algorithm is as follows:*

Setup(λ, N, L)*: It takes a security parameter λ, maximum number of users in each level N, and maximum length of the hierarchy of identity L as input. It then outputs a master public key* mpk*, a master secret key* msk*, initial state*

st_0, *and an empty revocation list* RL. *We assume that the message space* $\mathcal{M}$ *and the identity space* $\mathcal{I}$ *(which is a union of all levels of identities), time space* $\mathcal{T}$, *and ciphertext space* $\mathcal{CT}$ *are contained in* mpk.

$\mathsf{KeyGen}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{st}_{\mathsf{ID}_{|\ell}}, \mathsf{ID}_{|\ell+1}, \mathsf{mpk})$: *On input of a private key* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$ *and state* $\mathsf{st}_{\mathsf{ID}_{|\ell}}$, *children's identity* $\mathsf{ID}_{|\ell+1}$ *and the master public key* mpk, *it outputs a private key* $\mathsf{sk}_{\mathsf{ID}_{|\ell+1}}$ *and updates state* $\mathsf{st}_{\mathsf{ID}_{|\ell}}$.

$\mathsf{KeyUp}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{T}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{st}_{\mathsf{ID}_{|\ell}}, \mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}, \mathsf{mpk})$: *It takes a private key* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$ *of* $\mathsf{ID}_{|\ell} \in \mathcal{I}$, *key update time* $\mathsf{T} \in \mathcal{T}$, *a revocation list* $\mathsf{RL}_{\mathsf{ID}_{|\ell}}$, *state* $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ *of* $\mathsf{ID}_{|\ell}$, *key update* $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ *published by* $\mathsf{ID}_{|\ell-1}$, *and the master public key* mpk *as input, and outputs a key update* $\mathsf{ku}_{\mathsf{ID}_{|\ell},\mathsf{T}}$ *or* $\perp$ *(when* $\mathsf{ID}_{|\ell}$ *is revoked).*

$\mathsf{DKG}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}, \mathsf{T}, \mathsf{mpk})$: *Given private key* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$, *key update* $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$, *and the master public key* mpk, *it outputs a decryption key* $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}$ *that can be used during a time period* T *or* $\perp$ *that means the identity* $\mathsf{ID}_{|\ell}$ *is revoked for some time period* $\mathsf{T}' \leq \mathsf{T}$.

$\mathsf{Enc}(\mathsf{ID}_{|\ell}, \mathsf{T}, \mathsf{M}, \mathsf{mpk})$: *This algorithm takes an identity* $\mathsf{ID}_{|\ell} \in \mathcal{I}$, *time* $\mathsf{T} \in \mathcal{T}$, *a message* $\mathsf{M} \in \mathcal{M}$, *and the master public key* mpk *as input, and outputs a ciphertext* $\mathsf{CT} \in \mathcal{CT}$.

$\mathsf{Dec}(\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}, \mathsf{CT}, \mathsf{mpk})$: *Given a decryption key* $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}$, *a ciphertext* $\mathsf{CT} \in \mathcal{CT}$, *and the master public key* mpk, *it outputs a message* $\mathsf{M} \in \mathcal{M}$ *or* $\perp$ *(invalid ciphertext).*

$\mathsf{Revk}(\mathsf{ID}_{|\ell}, \mathsf{T}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{st}_{\mathsf{ID}_{|\ell-1}})$: *It takes an identity* $\mathsf{ID}_{|\ell} \in \mathcal{I}$, *time* $\mathsf{T} \in \mathcal{T}$, *the revocation list* $\mathsf{RL}_{\mathsf{ID}_{|\ell-1}}$ *managed by* $\mathsf{ID}_{|\ell-1}$ *and state* $\mathsf{st}_{\mathsf{ID}_{|\ell-1}}$, *and updates the revocation list* $\mathsf{RL}_{\mathsf{ID}_{|\ell-1}}$ *by adding* $\mathsf{ID}_{|\ell}$ *as a revoked user at time* T.

Note that two algorithms KeyGen *and* Revk *are stateful, and if* $\mathsf{ID}_{|\ell}$ *is revoked at time period* T, *then all descendants should be revoked at the time* T. *For rigorous definition, if* $\ell = 0$, *then let* $\mathsf{sk}_{\mathsf{ID}_{|0}} = \mathsf{msk}$, $\mathsf{st}_{\mathsf{ID}_{|0}} = \mathsf{st}_0$, $\mathsf{RL}_{\mathsf{ID}_{|0}} = \mathsf{RL}_0$, $\mathsf{ku}_{\mathsf{ID}_{|-1},t}$ *be* msk, *and* $\mathsf{ku}_{\mathsf{ID}_{|0},\mathsf{T}} = \mathsf{ku}_{0,\mathsf{T}}$.

We require the following correctness condition: For any output (mpk,msk*) of* Setup, *any message* $\mathsf{M} \in \mathcal{M}$, *any identity* $\mathsf{ID}_{|\ell}$ *(of length* $\ell \leq L$*), any time* $\mathsf{T} \in \mathcal{T}$, *all possible states* $\{\mathsf{st}_{\mathsf{ID}_{|i}}\}_{i\in[0,\ell-1]}$ *and revocation lists* $\{\mathsf{RL}_{\mathsf{ID}_{|i}}\}_{i\in[0,\ell-1]}$, *if* $\mathsf{ID}_{|\ell}$ *is not revoked (and so all ancestors of* $\mathsf{ID}_{|\ell}$ *are not revoked) by time* T, *the following probability should be* 1.

$$\Pr\Big[\mathsf{Dec}(\mathsf{dk}_{\mathsf{ID}_{|\ell}}, \mathsf{Enc}(\mathsf{ID}_{|\ell}, \mathsf{T}, \mathsf{M}, \mathsf{mpk}), \mathsf{mpk}) = \mathsf{M}\Big| \\ \textit{for } i \in [1,\ell],\ \mathsf{KeyGen}(\mathsf{sk}_{\mathsf{ID}_{|i-1}}, \mathsf{st}_{\mathsf{ID}_{|i-1}}, \mathsf{ID}_{|i}, \mathsf{mpk}) \to \mathsf{sk}_{\mathsf{ID}_{|i}}, \\ \mathsf{KeyUp}(\mathsf{sk}_{\mathsf{ID}_{|i-1}}, \mathsf{T}, \mathsf{RL}_{\mathsf{ID}_{|i-1}}, \mathsf{st}_{\mathsf{ID}_{|i-1}}, \mathsf{ku}_{\mathsf{ID}_{|i-2},\mathsf{T}}, \mathsf{mpk}) \to \mathsf{ku}_{\mathsf{ID}_{|i-1},\mathsf{T}}; \\ \mathsf{DKG}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}, \mathsf{mpk}) \to \mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}.\Big]$$

We provide the security definition for a revocable HIBE scheme by extending Boldyreva et al.'s security definition for a revocable IBE scheme [2]. Our security definition allows the adversary to access several oracles: KeyGen, KeyUp, and Revk. We provide the precise definition of the oracles and new security notion for RHIBE using such oracles.

Definition 5 (IND-sRID-CPA). *Let* $\mathcal{RHIBE}$=*(*Setup, KeyGen, KeyUp, DKG, Enc, Dec, Revk*) be an RHIBE scheme. First, we define three oracles.*

$\mathsf{KeyGen}(\cdot)$ *is the private key generation oracle. It takes an identity* $\mathsf{ID}_{|\ell}$ *of length* ℓ *as input, runs the* KeyGen *algorithm to get the private key* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$*, and returns* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$.

$\mathsf{KeyUp}(\cdot,\cdot)$ *is the key update oracle that takes time* T *and an identity* $\mathsf{ID}_{|\ell}$ *of length* ℓ *as input, runs the* KeyUp *algorithm to obtain the key update* $\mathsf{ku}_{\mathsf{ID}_{|\ell},\mathsf{T}}$*, and returns it.*

$\mathsf{Revk}(\cdot,\cdot,\cdot)$ *is the revocation oracle. It takes an identity* $\mathsf{ID}_{|\ell}$ *of length* ℓ*, its child identity* $\mathsf{ID}_{|\ell+1}$*, and time* T *as input, runs the the* Revk *algorithm to revoke* $\mathsf{ID}_{|\ell+1}$*, and updates* $\mathsf{RL}_{\mathsf{ID}_{|\ell}}$.

We assume that all oracles share a state. Next, we define the security of RHIBE, called IND-sRID-CPA security.

$$\boxed{\mathbf{Exp}_{\mathcal{RHIBE},\mathcal{A}}^{\textit{IND-sRID-CPA}}(\lambda)}$$
$$(\mathsf{ID}^*_{|\ell^*}, \mathsf{T}^*, \mathit{state}) \leftarrow \mathcal{A}(\mathit{state})$$
$$(\mathsf{mpk}, \mathsf{msk}, \mathsf{RL}_0, \mathsf{st}_0) \leftarrow \mathsf{Setup}(\lambda, N, L)$$
$$(\mathsf{M}^*_0, \mathsf{M}^*_1, \mathit{state}) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\cdot),\mathsf{KeyUp}(\cdot,\cdot),\mathsf{Revk}(\cdot,\cdot,\cdot)}(\mathit{state}, \mathsf{mpk})$$
$$b \xleftarrow{\$} \{0,1\}$$
$$\mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{ID}^*_{|\ell^*}, \mathsf{T}^*, \mathsf{M}^*_b, \mathsf{mpk})$$
$$b' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\cdot),\mathsf{KeyUp}(\cdot,\cdot),\mathsf{Revk}(\cdot,\cdot,\cdot)}(\mathit{state}, \mathsf{mpk}, \mathsf{CT}^*)$$
$$\mathit{Return} \begin{cases} 1 & \mathit{if}\ b = b' \\ 0 & \mathit{otherwise} \end{cases}.$$

There are three conditions that $\mathcal{A}$ *should follow.*

1. *The challenge messages* M^*_0 *and* M^*_1 *should have the same length.*
2. $\mathsf{KeyUp}(\cdot,\cdot)$ *and* $\mathsf{Revk}(\cdot,\cdot,\cdot)$ *can be queried at a time that is greater than or equal to the time of all previous queries; i.e. the adversary is allowed to query only in non-decreasing order of time. Also,* $\mathsf{Revk}(\cdot,\cdot,\cdot)$ *cannot be queried on a time* T *if* $\mathsf{KeyUp}(\cdot,\cdot)$ *was queried on* T.
3. *If* $\mathsf{KeyGen}(\cdot)$ *is queried on* $\mathsf{ID}^*_{|\ell}$*, which is an ancestor's identity of the challenge identity (that is,* $\ell < \ell^*$*), then* $\mathsf{Revk}(\cdot,\cdot,\cdot)$ *must be queried to revoke* $\mathsf{ID}^*_{|\ell}$ *at a time* T *for some* $\mathsf{T} \leq \mathsf{T}^*$*; hence,* $\mathsf{ID}^*_{|\ell^*}$ *is also directly revoked on the time* $\mathsf{T} < \mathsf{T}^*$.

The advantage of the adversary $\mathcal{A}$ *is defined as*

$$\mathit{Adv}_{\mathcal{RHIBE},\mathcal{A}}^{\textit{IND-sRID-CPA}}(\lambda) = \left| \Pr[\mathbf{Exp}_{\mathcal{RHIBE},\mathcal{A}}^{\textit{IND-sRID-CPA}}(\lambda) = 1] - \frac{1}{2} \right|.$$

If the function $\mathit{Adv}_{\mathcal{RHIBE},\mathcal{A}}^{\textit{IND-sRID-CPA}}$ *is negligible in the security parameter* λ*, we say that the scheme* $\mathcal{RHIBE}$ *is IND-sRID-CPA secure.*

4 Our Construction

In this section, we propose our main construction for the RHIBE scheme. For revocation, we use the same methodology using binary structures as that used in

Table 1. Quick notations

Preparation	**Notation**	**Meaning**
For $g \in \mathbb{G}$ and $\overrightarrow{r} = (r_1, \dots, r_m) \in \mathbb{Z}_p^m$,	$g^{\overrightarrow{r}}$	$(g^{r_1}, \dots, g^{r_m}) \in \mathbb{G}^m$
For $(g_1, \dots, g_m) \in \mathbb{G}^m$ and $\overrightarrow{r} = (r_1, \dots, r_m) \in \mathbb{Z}_p^m$,	$(g_1, \dots, g_m)^{\overrightarrow{r}}$	$\prod_{i=1}^m g_i^{r_i} \in \mathbb{G}$
For $g^{\overrightarrow{r}} = (g^{r_1}, \dots, g^{r_m})$ and $g^{\overrightarrow{s}} = (g^{s_1}, \dots, g^{s_m}) \in \mathbb{G}^m$,	$g^{\overrightarrow{r}} \circ g^{\overrightarrow{s}}$	$(g^{r_1} \cdot g^{s_1}, \dots, g^{r_m} \cdot g^{s_m}) \in \mathbb{G}^m$
For $g^{\overrightarrow{r}_1}, \dots, g^{\overrightarrow{r}_m} \in \mathbb{G}^m$,	$\bigotimes_{i=1}^m g^{\overrightarrow{r}_i}$	$g^{\overrightarrow{r}_1} \circ \dots \circ g^{\overrightarrow{r}_m} \in \mathbb{G}^m$
For $g_1, h_0, \dots, h_L \in \mathbb{G}$ and $\forall i \in [0, L]$,	$F_i(x)$	$g_1^x h_i \in \mathbb{G}$
For $\mathsf{T} \in \mathcal{T}$ and $\mathsf{ID}_{\vert\ell} = (\mathsf{I}_1, \dots, \mathsf{I}_\ell) \in \mathcal{I}$,	$\overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{\vert\ell})$	$(F_0(\mathsf{T}), F_1(\mathsf{I}_1), \dots, F_\ell(\mathsf{I}_\ell)) \in \mathbb{G}^{\ell+1}$
For $\mathsf{ID}_{\vert\ell} = (\mathsf{I}_1, \dots, \mathsf{I}_\ell) \in \mathcal{I}$,	$\overrightarrow{F}(*, \mathsf{ID}_{\vert\ell})$	$(1_{\mathbb{G}}, F_1(\mathsf{I}_1), \dots, F_\ell(\mathsf{I}_\ell)) \in \mathbb{G}^{\ell+1}$

all prior scalable RIBE schemes. In the RHIBE scheme, each intermediate level user $\mathsf{ID}_{\vert\ell-1}$ has its own binary tree $\mathsf{BT}_{\mathsf{ID}_{\vert\ell-1}}$ for revoking capabilities and issues the key update $\mathsf{ku}_{\mathsf{ID}_{\vert\ell-1},\mathsf{T}}$ at each time period T. Then, a non-revoked child user $\mathsf{ID}_{\vert\ell}$ can generate $\mathsf{dk}_{\mathsf{ID}_{\vert\ell},\mathsf{T}}$ from the key update $\mathsf{ku}_{\mathsf{ID}_{\vert\ell-1},\mathsf{T}}$ and its secret key $\mathsf{sk}_{\mathsf{ID}_{\vert\ell}}$.

Before providing our construction, we first define several notations for simple description of the proposed construction, which is given in Table 1. Our RHIBE scheme is based on the BB-HIBE scheme [3]. As we mentioned before, a trivial approach for delegation of revocation functionality will end up with exponential secret key size in the hierarchical level. Therefore, our main contribution is to propose an efficient way of dealing with delegation of revoking capabilities and show that the proposed methodology does not harm the semantic security of the underlying BB-HIBE scheme; that is, in the security proof, we give a reduction to the IND-sID-CPA security of the BB-HIBE scheme.

For a decryption key of a user $\mathsf{ID}_{\vert\ell}$ at a time period T, we use a hierarchical extension as follows:

$$\mathsf{dk}_{\mathsf{ID}_{\vert\ell},\mathsf{T}} = \Big(g_2^\alpha \cdot (g_1^{\mathsf{T}} h_0)^{s_0} \cdot \prod_{i \in [1,\ell]} (g_1^{\mathsf{I}_i} h_i)^{s_i}, g^{s_0}, g^{s_1} \dots, g^{s_\ell}\Big) \in \mathbb{G}^{\ell+2},$$

where $g, g_1, g_2, h_0, \dots, h_\ell$ are public parameters, g_2^α is a master key, $\mathsf{ID}_{\vert\ell} = (\mathsf{I}_1, \dots, \mathsf{I}_\ell)$, and $s_0, \dots, s_\ell$ are random integers chosen from $\mathbb{Z}_p$. If we use the notation in Table 1, then

$$\mathsf{dk}_{\mathsf{ID}_{\vert\ell},\mathsf{T}} = \Big(g_2^\alpha \cdot \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{\vert\ell})^{\overrightarrow{s}}, g^{\overrightarrow{s}}\Big) \text{ where } \overrightarrow{s} = (s_0, \dots, s_\ell).$$

Note that $\overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{\vert\ell})$ is a vector in $\mathbb{G}^{\ell+1}$ so that $\overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{\vert\ell})^{\overrightarrow{s}}$ is a group element in $\mathbb{G}$. Information about decryption keys are divided into secret keys and key updates. In particular, the master key g_2^α is randomly divided into two parts R_θ and g_2^α / R_θ, where we will explain θ later. The secret key of $\mathsf{ID}_{\vert\ell}$ contains information about

$$\Big(R_\theta \cdot \prod_{i \in [1,\ell]} (g_1^{\mathsf{I}_i} h_i)^{s_i}, 1_{\mathbb{G}}, g^{s_1} \dots, g^{s_\ell}\Big) = \Big(R_\theta \overrightarrow{F}(*, \mathsf{ID}_{\vert\ell})^{\overrightarrow{s}'}, g^{\overrightarrow{s}'}\Big),$$

where $\overrightarrow{s}' = (0, s_1, \ldots, s_\ell) \in \mathbb{Z}_p^{\ell+1}$, and the key update for a time T, which is managed by $\mathsf{ID}_{|\ell-1}$, contains information about

$$\Big((g_2^\alpha / R_\theta) \cdot (g_1^{\mathsf{T}} h_0)^{s_0} \cdot \prod_{i \in [1,\ell-1]} (g_1^{\mathsf{I}_i} h_i)^{s_i}, g^{s_0}, \ldots, g^{s_{\ell-1}}, 1_{\mathbb{G}}\Big)$$

$$= \Big((g_2^\alpha / R_\theta) \cdot \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell-1})^{\overrightarrow{s}''}, g^{\overrightarrow{s}''}\Big),$$

where $\overrightarrow{s}'' = (s_0, s_1, \ldots, s_{\ell-1}, 0) \in \mathbb{Z}_p^{\ell+1}$. Let R_θ be assigned in the node θ in $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$, $\zeta_{\mathsf{ID}_{|\ell}}$ be the leaf node assigned for $\mathsf{ID}_{|\ell}$ in $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$, and $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$ be the path from $\zeta_{\mathsf{ID}_{|\ell}}$ to the root node in $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$. If $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$ contains the above form for all θ on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$ and key update on time T contains the above form for all θ in $\mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T})$, then a non-revoked user $\mathsf{ID}_{|\ell}$ can generate the corresponding decryption key by a simple product of the above two forms since there exists at least one θ in $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}}) \cap \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T})$.

If we consider revocable IBE schemes, the above method is sufficient. However, when constructing the RHIBE scheme, we should consider the fact that users can generate valid key updates for children *only* during the time period in which they are not revoked. This implies that both secret keys and key updates should contain information about all ancestor's secret keys and key updates. To this end, we recursively define children's secret keys and key updates from parents' secret key and key update. Note that if we use binary tree structures for revocation, the KGC cannot directly generate secret keys of descendants (except for the first-level user) since the master key parts of decryption keys of descendants are randomly divided into two parts (each for the secret key and key update) according to the binary tree structure managed by their parents, which are also intermediate-level users, but the KGC does not know of such a binary tree structure. Therefore, the secret key and key updates have to be recursively defined, which makes the situation more complicated.

A simple example helps to explain our construction for delegating the revocation functionality. Assume that a user $\mathsf{ID}_{|1} = \mathsf{I}_1$ is not revoked on time T. The secret key of $\mathsf{ID}_{|1}$ and the key update $\mathsf{ku}_{0,\mathsf{T}}$ generated by the KGC are

$$\{(R_\theta \cdot (g_1^{\mathsf{I}_1} h_1)^{\gamma_1}, g^{\gamma_1})\}_{\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})} \text{ and } \{((g_2^\alpha / R_\theta) \cdot (g_1^{\mathsf{T}} h_0)^{\gamma_0}, g^{\gamma_0})\}_{\theta \in \mathsf{KUNode}(\mathsf{BT}_0, \mathsf{RL}_0, \mathsf{T})},$$

respectively, where $\zeta_{\mathsf{ID}_{|1}}$ is a leaf node assigned for $\mathsf{ID}_{|1}$ by the KGC. The user $\mathsf{ID}_{|1}$ has its own binary tree structure $\mathsf{BT}_{\mathsf{ID}_{|1}}$ for revocation functionality. The revocation methodology used by $\mathsf{ID}_{|1}$ is the same as for the KGC. Whenever a child $\mathsf{ID}_{|2}$ registers in the system, it randomly assigns a leaf node $\zeta_{\mathsf{ID}_{|2}}$ of $\mathsf{BT}_{\mathsf{ID}_{|1}}$ for $\mathsf{ID}_{|2}$. For all nodes θ' in $\mathsf{Path}(\zeta_{\mathsf{ID}_{|2}})$, assign random values $R_{\theta'}$. Assume that the secret key of $\mathsf{ID}_{|1}$ is $\{(d'_\theta, d_\theta) := (R_\theta \cdot (g_1^{\mathsf{I}_1} h_1)^{\gamma_1}, g^{\gamma_1})\}_{\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})}$. The secret key $\mathsf{sk}_{\mathsf{ID}_{|2}}$ of a child $\mathsf{ID}_{|2} := (\mathsf{I}_1, \mathsf{I}_2)$ is generated as

$$\{(d'_\theta \cdot (g_1^{\mathsf{I}_2} h_2)^{\gamma_2},\ 1_{\mathbb{G}},\ d_\theta,\ g^{\gamma_2})\}_{\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})}$$
$$\text{and } \{(R_{\theta'} \cdot \textstyle\prod_{i \in [1,2]} (g_1^{\mathsf{I}_i} h_i)^{\gamma'_i},\ 1_{\mathbb{G}},\ g^{\gamma'_1},\ g^{\gamma'_2})\}_{\theta' \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|2}})}.$$

We use notation $\overrightarrow{d}^{(1,j)}_{\mathsf{ID}_{|2}}$ to denote a vector $(d'_\theta \cdot (g_1^{\mathsf{I}_2} h_2)^{s_2},\ 1_{\mathbb{G}},\ d_\theta,\ g^{s_2}) \in \mathbb{G}^4$, where θ is the j-th level node, and notation $\overrightarrow{d}^{(2,j)}_{\mathsf{ID}_{|2}}$ to denote a vector $(R_{\theta'} \cdot \prod_{i\in[1,2]}(g_1^{\mathsf{I}_i} h_i)^{\gamma_i},\ 1_{\mathbb{G}},\ g^{\gamma_1},\ g^{\gamma_2}) \in \mathbb{G}^4$, where θ' is the j-th level node.

The key update on T is computed as follows. Assume that $\mathsf{ID}_{|1}$ is not revoked on time T. Then, $\mathsf{ID}_{|1}$ can choose a node θ in $\mathsf{Path}(\zeta_{\mathsf{ID}_{|1}}) \cap \mathsf{KUNode}(\mathsf{BT}_0, \mathsf{RL}_0, \mathsf{T})$. Let Lv_1 be the level of θ in BT_0. Note that all nodes in $\mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})$ have different levels, so we can identify nodes from their corresponding levels. Then, $(f_1, f_2) := ((g_2^\alpha / R_\theta) \cdot (g_1^{\mathsf{T}} h_0)^{s_0}, g^{s_0})$ is a valid key update for the first level users including $\mathsf{ID}_{|1}$, and the key update for children of $\mathsf{ID}_{|1}$ is generated as

$$\{(\{\mathsf{Lv}_1\},\ (f_1/R_{\theta'})(g_1^{\mathsf{I}_1} h_1)^\delta,\ f_2,\ g^\delta,\ 1_{\mathbb{G}})\}_{\theta' \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|1}}, \mathsf{RL}_{\mathsf{ID}_{|1}}, \mathsf{T})}.$$

We use notation $\overrightarrow{f}_{\mathsf{ID}_{|1},\theta'}$ to denote a vector $((f_1/R_{\theta'})(g_1^{\mathsf{I}_1} h_1)^\delta,\ f_2,\ g^\delta,\ 1_{\mathbb{G}}) \in \mathbb{G}^4$.

The decryption key of $\mathsf{ID}_{|2}$ on time T is generated as follows. First, it identifies the subkey part of $\mathsf{sk}_{\mathsf{ID}_{|1}}$ that is used by its parent for delegation on time T. (It can see from Lv_1 in $\mathsf{ku}_{\mathsf{ID}_{|1},\mathsf{T}}$.) Let θ be the level Lv_1 node on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})$. Next, if $\mathsf{ID}_{|2} = (\mathsf{I}_1, \mathsf{I}_2)$ is not revoked, it can choose a node θ' on $\mathsf{Path}(\zeta_{\mathsf{ID}_{|2}}) \cap \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|1}}, \mathsf{RL}_{\mathsf{ID}_{|1}}, \mathsf{T})$. Let Lv_2 be the level of θ'. It then generates the decryption key as

$$\begin{aligned}&\textstyle\bigotimes_{i=1}^{2}(\overrightarrow{d}^{(i,\mathsf{Lv}_i)}_{\mathsf{ID}_{|2}}) \circ \overrightarrow{f}_{\mathsf{ID}_{|1},\theta'}\\ &= \Big(\Big(d'_\theta (g_1^{\mathsf{I}_2} h_2)^{\gamma_2}\Big)\Big(R_{\theta'} \textstyle\prod_{i\in[1,2]}(g_1^{\mathsf{I}_i} h_i)^{\gamma'_i}\Big)\Big(f_1/R_{\theta'}(g_1^{\mathsf{I}_1} h_1)^\delta\Big),\ f_2,\ d_\theta g^{\gamma'_1} g^\delta,\ g^{\gamma_2} g^{\gamma'_2}\Big).\end{aligned}$$

A simple calculation shows that the above decryption key has the desired form

$$\Big(g_2^\alpha \cdot (g_1^{\mathsf{T}} h_0)^{s_0} (g_1^{\mathsf{I}_i} h_1)^{\gamma_1+\gamma'_1+\delta} (g_1^{\mathsf{I}_i} h_2)^{\gamma_2+\gamma'_2},\ g^{s_0},\ g^{\gamma_1+\gamma'_1+\delta},\ g^{\gamma_2+\gamma'_2}\Big).$$

In a similar way, we can define the secret keys and key updates for users from other levels. However, the security of the above construction is not easy to prove since descendants have a great deal of information about the ancestors' secret keys. In particular, in the security model the adversary can query $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*+1}}$, but the simulator may not generate such a secret key without knowing $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*}}$ since secret key generation algorithm is recursively defined. (When the challenge identity $\mathsf{ID}^*_{|\ell^*}$ is not revoked on the challenge time T^*, the simulator cannot generate $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*}}$. If not, the simulator can generate the $\mathsf{dk}_{\mathsf{ID}^*_{|\ell^*},\mathsf{T}^*}$ by itself from $\mathsf{sk}_{\mathsf{ID}^*_{\ell^*}}$ and $\mathsf{ku}_{\mathsf{ID}_{\ell^*-1},\mathsf{T}}$ so that the simulator can solve the underlying problem without a help of the adversary.) To circumvent this obstacle, we slightly modify the above construction by adding re-randomization processes in the KeyGen and KeyUp algorithms. Hence, the simulator can generate $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*+1}}$ even when it does not know $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*}}$ since all randomness used in $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*+1}}$ is independent from $\mathsf{sk}_{\mathsf{ID}^*_{|\ell^*}}$.

Now we describe our RHIBE construction. In our construction, each user $\mathsf{ID}_{|\ell}$ keeps state information $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ including a binary tree $\mathsf{BT}_{\mathsf{ID}_{|\ell}}$. We sometimes use $\mathsf{BT}_{\mathsf{ID}_{|\ell}}$ to precisely indicate the state information associated with the binary tree. $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ contains the randomness $\{\tilde{R}_{(i,j)}\}_{(i,j)\in[1,\ell-1]\times[1,n]}$ used for the re-randomization process as well. All states are initialized as empty sets.

$\mathsf{Setup}(\lambda, N, L)$: We assume, without loss of generality, that $N = 2^n$ for some n. Randomly choose group elements $g, g_2, h_0, \ldots, h_L \xleftarrow{\$} \mathbb{G}$ and an integer $\alpha \xleftarrow{\$} \mathbb{Z}_p$. Set $\mathsf{mpk} = \{g, g_1 = g^\alpha, g_2, h_0 \ldots, h_L\}$ and $\mathsf{msk} = \{g_2^\alpha\}$.

$\mathsf{KeyGen}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{st}_{\mathsf{ID}_{|\ell}}, \mathsf{ID}_{|\ell+1}, \mathsf{mpk})$: According to the value ℓ, this algorithm is differently defined.

$\ell = 0$: Note that $\mathsf{sk}_{\mathsf{ID}_{|0}} = \mathsf{msk}$ and $\mathsf{st}_{\mathsf{ID}_{|0}} = \mathsf{st}_0$. Randomly choose an unassigned leaf ζ from $\mathtt{BT}_0$, and store $\mathsf{ID}_{|1}$ in the node ζ. We sometimes use a notation ζ_{ID} to precisely indicate that the node ζ is associated with ID. For all $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|1}}) \subset \mathtt{BT}_0$,

1. Recall R_θ from θ in $\mathtt{BT}_0$ if it is defined. Otherwise, $R_\theta \xleftarrow{\$} \mathbb{G}$ and store it in the node $\theta \in \mathtt{BT}_0$.
2. Choose $\gamma_\theta \xleftarrow{\$} \mathbb{Z}_p$ and compute $\overrightarrow{d}^{(1,j)}_{\mathsf{ID}_{|1}} := (R_\theta \cdot F_1(\mathsf{ID}_{|1})^{\gamma_\theta}, 1_{\mathbb{G}},\ g^{\gamma_\theta}) \in \mathbb{G}^3$, where j is the level of θ on the path $\mathsf{Path}(\zeta_{\mathsf{ID}_{|1}})$.

Return $\mathsf{sk}_{\mathsf{ID}_{|1}} = \{\overrightarrow{d}^{(1,j)}_{\mathsf{ID}_{|1}} \in \mathbb{G}^3\}_{j \in [1,n]}$.

$\ell > 0$: Randomly choose an unassigned leaf ζ from $\mathtt{BT}_{\mathsf{ID}_{|\ell}}$, and store $\mathsf{ID}_{|\ell+1}$ in the node $\zeta_{\mathsf{ID}_{|\ell+1}}$. Parse $\mathsf{sk}_{\mathsf{ID}_{|\ell}} = \{\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}} \in \mathbb{G}^{\ell+2}\}_{(i,j) \in [1,\ell] \times [1,n]}$.

For all $(i, j) \in [1, \ell] \times [1, n]$,

1. Recall $\tilde{R}_{(i,j)}$ from $\mathsf{st}_{\mathsf{ID}_{|\ell}}$ if it is defined. Otherwise, $\tilde{R}_{(i,j)} \xleftarrow{\$} \mathbb{G}$ and store it in $\mathsf{st}_{\mathsf{ID}_{|\ell}}$.
2. Choose $\overrightarrow{\gamma}_{(i,j)} \xleftarrow{\$} \{0\} \times \mathbb{Z}_p^{\ell+1}$ and compute $\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell+1}} := (\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}},\ 1_{\mathbb{G}}) \circ (\tilde{R}_{(i,j)} \cdot \overrightarrow{F}(*, \mathsf{ID}_{|\ell+1})^{\overrightarrow{\gamma}_{(i,j)}},\ g^{\overrightarrow{\gamma}_{(i,j)}}) \in \mathbb{G}^{\ell+3}$.

For all $\theta \in \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell+1}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell}}$,

1. Recall R_θ from the corresponding node θ in $\mathtt{BT}_{\mathsf{ID}_{|\ell}}$ if it is defined. Otherwise, $R_\theta \xleftarrow{\$} \mathbb{G}$ and store it in the node θ.
2. Choose $\overrightarrow{\gamma}_\theta \xleftarrow{\$} \{0\} \times \mathbb{Z}_p^{\ell+1}$ and compute $\overrightarrow{d}^{(\ell+1,j)}_{\mathsf{ID}_{|\ell+1}} := (R_\theta \cdot \overrightarrow{F}(*, \mathsf{ID}_{|\ell+1})^{\overrightarrow{\gamma}_\theta},\ g^{\overrightarrow{\gamma}_\theta}) \in \mathbb{G}^{\ell+3}$, where j is the level of θ in the tree $\mathtt{BT}_{\mathsf{ID}_{|\ell}}$.

Return $\mathsf{sk}_{\mathsf{ID}_{|\ell+1}} = \{\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell+1}} \in \mathbb{G}^{\ell+3}\}_{(i,j) \in [1,\ell+1] \times [1,n]}$.

$\mathsf{KeyUp}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{T}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{st}_{\mathsf{ID}_{|\ell}}, \mathsf{ku}_{\mathsf{ID}_{|\ell-1}, \mathsf{T}}, \mathsf{mpk})$: According to the value ℓ, this algorithm is differently defined.

$\ell = 0$: Note that $\mathsf{sk}_{\mathsf{ID}_{|0}} = \mathsf{msk}$, $\mathsf{RL}_{\mathsf{ID}_{|0}} = \mathsf{RL}_0$, $\mathsf{st}_{\mathsf{ID}_{|0}} = \mathsf{st}_0$, and $\mathsf{ku}_{\mathsf{ID}_{|-1}, \mathsf{T}} = \mathsf{msk}$. For all nodes $\theta \in \mathsf{KUNode}(\mathtt{BT}_0, \mathsf{RL}_0, \mathsf{T})$,

1. Recall R_θ from the node $\theta \in \mathtt{BT}_0$. Note that R_θ is already defined during the key generation process.
2. Choose $\delta_\theta \xleftarrow{\$} \mathbb{Z}_p$ and compute $\overrightarrow{f}_{0,\theta}$ by $((g_2^\alpha / R_\theta) F_0(\mathsf{T})^{\delta_\theta},\ g^{\delta_\theta},\ 1_{\mathbb{G}}) \in \mathbb{G}^3$.

Return $\mathsf{ku}_{0,\mathsf{T}} = \{\emptyset,\ \overrightarrow{f}_{0,\theta} \in \mathbb{G}^3\}_{\theta \in \mathsf{KUNode}(\mathtt{BT}_0, \mathsf{RL}_0, \mathsf{T})}$.

$\ell > 0$:

1. Parse $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ as $\{\{\mathsf{Lv}_i\}_{i\in[1,\ell-1]},\ \overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\theta} \in \mathbb{G}^{\ell+2}\}_{\theta\in\mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell-1}},\mathsf{RL}_{\mathsf{ID}_{|\ell-1}},\mathsf{T})}$.
   Note that if $\ell = 1$, then $\{\mathsf{Lv}_i\}_{i\in[1,0]}$ means $\emptyset$.
2. Identify one node $\tilde{\theta} \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T}) \cap \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$ and set Lv_ℓ to be the level of $\tilde{\theta}$.
3. For all $i \in [1,\ell]$, recall $\tilde{R}_{(i,\mathsf{Lv}_i)}$ from $\mathsf{st}_{\mathsf{ID}_{|\ell}}$.

For all nodes $\theta \in \mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell}}, \mathsf{RL}_{\mathsf{ID}_{|\ell}}, \mathsf{T})$,

1. Recall R_θ from $\theta \in \mathsf{BT}_{\mathsf{ID}_{|\ell}}$. Note that R_θ is already defined during the key generation process.
2. Choose $\overrightarrow{\delta}_\theta \xleftarrow{\$} \mathbb{Z}_p^{\ell+1}$ and compute $\overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta}$ by
   $(\overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\tilde{\theta}},\ 1_{\mathbb{G}}) \ \circ\ ((R_\theta \prod_{i=1}^{\ell} \tilde{R}_{(i,\mathsf{Lv}_i)})^{-1} \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{\overrightarrow{\delta}_\theta},\ g^{\overrightarrow{\delta}_\theta},\ 1_{\mathbb{G}})$.

Return $\mathsf{ku}_{\mathsf{ID}_{|\ell},\mathsf{T}} = \{\{\mathsf{Lv}_i\}_{i\in[1,\ell]},\ \overrightarrow{f}_{\mathsf{ID}_{|\ell},\theta} \in \mathbb{G}^{\ell+3}\}_{\theta\in\mathsf{KUNode}(\mathsf{BT}_{\mathsf{ID}_{|\ell}},\mathsf{RL}_{\mathsf{ID}_{|\ell}},\mathsf{T})}$.

$\mathsf{DKG}(\mathsf{sk}_{\mathsf{ID}_{|\ell}}, \mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}, \mathsf{T}, \mathsf{mpk})$:

1. Parse $\mathsf{sk}_{\mathsf{ID}_{|\ell}} = \{\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}} \in \mathbb{G}^{\ell+2}\}_{(i,j)\in[1,\ell]\times[1,n]}$ and
   $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}} = \{\{\mathsf{Lv}_i\}_{i\in[1,\ell-1]},\ \overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\theta} \in \mathbb{G}^{\ell+2}\}_{\theta\in\mathsf{J}}$.
2. If $\mathsf{J} \cap \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}}) = \emptyset$, then return $\perp$. Otherwise, choose a node $\theta \in \mathsf{J} \cap \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$ and let Lv_ℓ be the level of θ in $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}}) \subset \mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$.
3. Compute and output $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}} := \bigotimes_{i=1}^{\ell}(\overrightarrow{d}^{(i,\mathsf{Lv}_i)}_{\mathsf{ID}|\ell}) \ \circ\ \overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\theta} \in \mathbb{G}^{\ell+2}$.

$\mathsf{Enc}(\mathsf{ID}_{|\ell}, \mathsf{T}, \mathsf{M}, \mathsf{mpk})$: Choose a random value $t \xleftarrow{\$} \mathbb{Z}_p$ and return

$$\mathsf{CT} = (\mathsf{M} \cdot e(g_1, g_2)^t,\ \ g^t,\ \ F_0(\mathsf{T})^t,\ F_1(\mathsf{I}_1)^t, \ldots, F_\ell(\mathsf{I}_\ell)^t).$$

$\mathsf{Dec}(\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}, \mathsf{CT}, \mathsf{mpk})$: Parse $\mathsf{CT} = (A, B, C_0, C_1, \ldots, C_\ell)$ and $dk_{\mathsf{ID}_{|\ell},\mathsf{T}} = (D', D_0, \ldots, D_\ell)$ and return

$$A \cdot \frac{\prod_{i=0}^{\ell} e(C_i, D_i)}{e(B, D')} = \mathsf{M}.$$

$\mathsf{Revoke}(\mathsf{ID}_{|\ell}, \mathsf{T}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{st}_{\mathsf{ID}_{|\ell-1}})$: Let ζ be the leaf node in $\mathsf{BT}_{\mathsf{ID}_{|\ell-1}}$ associated with $\mathsf{ID}_{|\ell}$. Update the revocation list by $\mathsf{RL}_{\mathsf{ID}_{|\ell-1}} \leftarrow \mathsf{RL}_{\mathsf{ID}_{|\ell-1}} \cup \{(\zeta, \mathsf{T})\}$ and return the updated revocation list.

Efficiency. For encrypting to the ℓ-th level user, the ciphertext consists of $\ell + 2$ group elements in $\mathbb{G}$ and an element in $\mathbb{G}_t$. The decryption algorithm requires $\ell + 2$ pairings and $\ell + 3$ multiplications in $\mathbb{G}_t$. Each user in the ℓ-th level keeps $(\ell + 2)(\ell + 1) \log N$ group elements in $\mathbb{G}$ as its secret key.

5 Security Analysis

We provide a series of lemmas to thoroughly explain the forms of secret keys, key updates, and decryption keys well, and then give a theorem for the IND-sRID-CPA security of the proposed construction. The proofs of the lemmas and the theorem are given in the full version of this paper.

Lemma 1. *If a secret key* $\mathsf{sk}_{\mathsf{ID}_{|\ell}}$ *is normally generated, it has the following form:*

$$\mathsf{sk}_{\mathsf{ID}_{|\ell}} = \left\{ \overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}} \in \mathbb{G}^{\ell+2} \text{ for } (i,j) \in [1,\ell] \times [1,n] \right\},$$

where

1. $\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}} = \left(R'_{(i,j)} \overrightarrow{F}(*, \mathsf{ID}_{|\ell})^{\overrightarrow{r}_{(i,j)}}, g^{\overrightarrow{r}_{(i,j)}} \right)$ *for a uniformly distributed vector* $\overrightarrow{r}_{(i,j)} \in \{0\} \times \mathbb{Z}_p^{\ell}$ *and uniformly distributed value* $R'_{(i,j)} \in \mathbb{G}$. *(This implies that all randomness used in a secret key is independent from the parent's secret key.)*
2. *For* $i \in [1, \ell-1] \times [1,n]$, *all children of* $\mathsf{ID}_{|\ell-1}$ *have the same value* $R'_{(i,j)}$.
3. $R'_{(\ell,j)}$ *is an associated value with the* j*-th level node on* $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell-1}}$.

Lemma 2. *If a key update* $\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}}$ *is normally generated, it has the following form:*

$$\mathsf{ku}_{\mathsf{ID}_{|\ell-1},\mathsf{T}} = \left\{ \{\mathsf{Lv}_i\}_{i\in[1,\ell-1]},\ \overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\theta} \in \mathbb{G}^{\ell+2} \right\}_{\theta \in \mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T})},$$

where for $\overrightarrow{s}_{\theta} \in \mathbb{Z}_p^{\ell}$ *and* $\overrightarrow{f}_{\mathsf{ID}_{|\ell-1},\theta} = \Big(g_2^{\alpha} (R'_{\theta} \prod_{i=1}^{\ell-1} R'_{(i,\mathsf{Lv}_i)})^{-1} \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell-1})^{\overrightarrow{s}_{\theta}}, g^{\overrightarrow{s}_{\theta}}, 1_{\mathbb{G}} \Big)$, R'_{θ} *is an associated value with a node* θ *in* $\mathtt{BT}_{\mathsf{ID}_{|\ell-1}}$, *and* $R'_{(i,j)}$ *is a value defined in the secret key* $\overrightarrow{d}^{(i,j)}_{\mathsf{ID}_{|\ell}}$ *of Lemma 1, which is the same for any children* $\mathsf{ID}_{|\ell}$ *of* $\mathsf{ID}_{|\ell-1}$. *Moreover, for some* $\mathsf{ID}_{|\ell}$, *if* $\theta \in \mathsf{KUNode}(\mathtt{BT}_{\mathsf{ID}_{|\ell-1}}, \mathsf{RL}_{\mathsf{ID}_{|\ell-1}}, \mathsf{T}) \cap \mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}})$, *then* $R'_{\theta} = R'_{(\ell,\mathsf{Lv}_{\ell})}$, *where* Lv_{ℓ} *is the level of* θ *on* $\mathsf{Path}(\zeta_{\mathsf{ID}_{|\ell}}) \subset \mathtt{BT}_{\mathsf{ID}_{|\ell-1}}$, *and* $R'_{(\ell,\mathsf{Lv}_{\ell})}$ *is a value defined in the secret key* $\overrightarrow{d}^{(\ell,\mathsf{Lv}_{\ell})}_{\mathsf{ID}_{|\ell}}$ *of Lemma 1.*

Lemma 3. *If a decryption key* $\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}}$ *is normally generated, it has the following form:*

$$\mathsf{dk}_{\mathsf{ID}_{|\ell},\mathsf{T}} = (g_2^{\alpha} \overrightarrow{F}(\mathsf{T}, \mathsf{ID}_{|\ell})^{\overrightarrow{s}}, g^{\overrightarrow{s}}) \ \in \mathbb{G}^{\ell+2},$$

where $\overrightarrow{s}$ *is a vector in* $\mathbb{Z}_p^{\ell+1}$.

Even though our KeyGen algorithm (KeyUp algorithm, respectively) is recursively defined, the above lemmas dictate that the secret key (key update, respectively) in each level has the same format and the randomness used in each level is totally independent from those in the other levels. This fact gives us an essential advantage when we construct a simulator in the security proof; when the simulator generates a secret key (key update, respectively), it is not necessary to generate all ancestor's secret keys (key updates, respectively), though KeyGen (KeyUp, respectively) is recursively defined in the real scheme. Instead, in the proof, the simulator can directly simulate with fresh randomness.

Theorem 1. *Assume that the original BB-HIBE scheme is IND-sID-CPA secure. Then, the proposed RHIBE scheme is IND-sRID-CPA secure.*

Corollary 1. *The proposed RHIBE scheme is IND-sRID-CPA secure under the DBDH assumption.*

6 Summary and Open Problems

We proposed the first construction for efficient delegation of both key generating functionality and revocation functionality in the IBE system.

There are interesting open problems. Our construction is based on the BB-HIBE scheme and we proved only selective-security of our construction. Natural open problem is to construct RHIBE scheme based on more efficient (in the sense of the ciphertext size) and secure (in the sense of satisfying adaptive-security) HIBE scheme (e.g.,[15]). Another open problem is to combine HIBE scheme with so-called Subset Difference (SD) method [20] (instead of CS). It seems not easy to combine SD with (H)IBE scheme since the SD method requires more complicated key distributing method than CS method.

Acknolwdgement. We thank anonymous reviewers of CT-RSA 2013 and members of Shin-Akarui-Angou-Benkyou-Kai for their helpful comments.

References

1. Bellare, M., Palacio, A.: Protecting against key exposure: strongly key-insulated encryption with optimal threshold. In: IACR Cryptology ePrint Archive 2002:064 (2002)
2. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: ACM CCS 2008, pp. 417–426 (2008)
3. 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)
4. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
5. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. Boyen, X., Waters, B.: Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
7. Chen, J., Lim, H.W., Ling, S., Wang, H., Nguyen, K.: Revocable Identity-Based Encryption from Lattices. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 390–403. Springer, Heidelberg (2012)
8. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-Insulated Public Key Cryptosystems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 65–82. Springer, Heidelberg (2002)
9. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
10. Gentry, C., Halevi, S.: Hierarchical Identity Based Encryption with Polynomially Many Levels. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 437–456. Springer, Heidelberg (2009)

11. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
12. Hanaoka, G., Weng, J.: Generic Constructions of Parallel Key-Insulated Encryption. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 36–53. Springer, Heidelberg (2010)
13. Hanaoka, Y., Hanaoka, G., Shikata, J., Imai, H.: Identity-Based Hierarchical Strongly Key-Insulated Encryption and Its Application. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 495–514. Springer, Heidelberg (2005)
14. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
15. Lewko, A., Waters, B.: New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
16. Lewko, A., Waters, B.: Unbounded HIBE and Attribute-Based Encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 547–567. Springer, Heidelberg (2011)
17. Libert, B., Quisquater, J.-J., Yung, M.: Parallel Key-Insulated Public Key Encryption Without Random Oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 298–314. Springer, Heidelberg (2007)
18. Libert, B., Vergnaud, D.: Adaptive-ID Secure Revocable Identity-Based Encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 1–15. Springer, Heidelberg (2009)
19. Libert, B., Vergnaud, D.: Towards Black-Box Accountable Authority IBE with Short Ciphertexts and Private Keys. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 235–255. Springer, Heidelberg (2009)
20. Naor, D., Naor, M., Lotspiech, J.: Revocation and Tracing Schemes for Stateless Receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
21. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
22. Seo, J.H., Kobayashi, T., Ohkubo, M., Suzuki, K.: Anonymous Hierarchical Identity-Based Encryption with Constant Size Ciphertexts. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 215–234. Springer, Heidelberg (2009)
23. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
24. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
25. Weng, J., Liu, S., Chen, K., Zheng, D., Qiu, W.: Identity-Based Threshold Key-Insulated Encryption without Random Oracles. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 203–220. Springer, Heidelberg (2008)

The Low-Call Diet: Authenticated Encryption for Call Counting HSM Users

Mike Bond[1], George French[2], Nigel P. Smart[3], and Gaven J. Watson[3]

[1] Cryptomathic A/S, Cambridge, UK
[2] Barclays Bank Plc, London, UK
[3] University of Bristol, UK

Abstract. We present a new mode of operation for obtaining authenticated encryption suited for use in environments, e.g. banking and government, where cryptographic services are only available via a Hardware Security Module (HSM) which protects the keys but offers a limited API. The practical problem is that despite the existence of better modes of operation, modern HSMs still provide nothing but a basic (unauthenticated) CBC mode of encryption, and since they mediate all access to the key, solutions must work around this. Our mode of operation makes only a single call to the HSM, yet provides a secure authenticated encryption scheme; authentication is obtained by manipulation of the plaintext being passed to the HSM via a call to an unkeyed hash function. The scheme offers a considerable performance improvement compared to more traditional authenticated encryption techniques which must be implemented using multiple calls to the HSM. Our new mode of operation is provided with a proof of security, on the assumption that the underlying block cipher used in the CBC mode is a strong pseudorandom permutation, and that the hash function is modelled as a random oracle.

1 Introduction

Authenticated symmetric encryption, namely an encryption scheme which is both IND-CPA and INT-CTXT secure [3], is regarded as *the* goal for symmetric encryption. There is no shortage of constructions in the literature for such authenticated encryption (AE) schemes. The most famous of these is the Encrypt-Then-MAC construction, which first encrypts the message using an IND-CPA encryption scheme and then appends a secure MAC to the ciphertext. Over the last decade various special modes of operation have been defined which implement authenticated encryption such as OCB [17], CCM [19], EAX [5] and GCM [12]. However, while these modes have been optimised for modern use and parallel services, there are some situations in which generic constructions (like Encrypt-Then-MAC) or special modes cannot be used.

In the financial and government sectors, a common industrial deployment of cryptography is for keys to reside within a special piece of hardware known as a Hardware Security Module or HSM. Such HSMs store the keys and manipulate sensitive data on behalf of applications, and offer a measure of assurance that neither a lone corrupt insider manipulating his own systems nor an external hacker will be able to steal or abuse cryptographic keys. Another benefit of utilizing HSMs is that they define the point at which any abuse of the key material may occur; so whilst a given HSM may not be truly secure (e.g. it may be susceptible to API attacks [6,8]), nor may they require authentication of the caller, they do from a pragmatic point of view provide the location where any abuse by an attacker will occur.

In practice the dominant characteristic of HSMs is their compliance and certification against security standards such as NIST FIPS 140-2, PCI HSM, and in certain business areas common criteria evaluation (e.g. against the Secure Signature Creation Device (SSCD) profile). These standards have varying levels of practical applicability – for instance FIPS 140 has relatively little to say about software API security but is widely depended upon as a measure of all round security. The standards are completely entrenched and not just adherence but certification is required in order to do business. Certification against such standards is expensive and applies to each version of HSM firmware and hardware released. Consequently the focus on security compliance and certification has slowed the rate of software change on HSMs, and it is rarely cost effective for a supplier to speculatively add new modes of operation in the hope they will be adopted.

To compound the matter, in a chicken-and-egg situation, the banking industry has been reluctant to specify authenticated encryption as part of standards for core HSM activities such as manipulation of customer PINs in banking networks and EMV[1], since in making this demand they will be requiring implementers to change their HSM firmware or radically increase their HSM load by making multiple calls per encrypt/decrypt.

Finally, for software vendors implementing software which use HSMs for crypto, different HSMs are dominant and are demanded in different regional markets[2]. Supporting a variety of vendors is both required by the market and is wise to enable price competition, hence the algorithm chosen for crypto must be the lowest common denominator across all these platforms.

Thus creation of more complex cryptographic construction is usually done via multiple API calls to the HSM. For example an HSM may provide an API call to perform CBC Mode with AES. To call this operation the user passes the plaintext to the HSM and specifies the key to use. The HSM recovers the key (usually looking it up in an internal store by name, or possibly by unwrapping a supplied encrypted key via some other internal key), applies CBC Mode to the message and then passes the ciphertext back to the caller. If Encrypt-then-MAC is required the whole process is then repeated again to obtain the MAC on the ciphertext. Thus traditional ways of obtaining authenticated encryption are computationally expensive.

[1] EMV is the major international smartcard based payment scheme named after its founders Europay Mastercard and Visa.

[2] Since the USA reclassification of crypto hardware in 2000 as no longer a munition the market is consolidating, but this takes time.

The whole process is also very expensive in terms of latency; nowadays HSMs are network attached remote computers and thus are often orders of magnitude slower than using local crypto in the CPU of the calling machine. In addition HSMs are also rate-limited and price differentiated by supported number of transactions per second. Thus minimizing calls to an HSM is a major application requirement.

Note that considering the lowest common denominator across HSMs, CBC Mode may only be available with the all-zero IV, thus the HSM does not even provide an IND-CPA encryption scheme. This can be fixed by using the HSMs ability to generate random numbers, or using some other random number source, to produce a random IV and then xor-ing the first block of the plaintext with the IV before calling the HSM, but with a naive implementation this introduces an extra call to acquire randomness - yet again incurring a performance penalty.

Fundamentally HSM APIs were designed before the need for authenticated encryption was properly understood. Yet the cryptographic community has developed new modes for authenticated encryption which focus on squeezing all data fields including randomisation into a single blockcipher block; as well as obtaining authentication with only a single pass of the data. In particular, the specific new modes of operation for authenticated encryption (such as Galois Counter Mode) are not supported by such HSMs; and are unlikely to be so universally in the near future.

As the financial industry and local and national government come under greater pressure to protect data such as Personally Identifying Information (PII), there is a problem using existing HSM approaches, since this new data is much larger than the size of a single blockcipher block (unlike bank card PINs or cryptographic keys). The generic construction such as Encrypt-Then-MAC, utilizing two keys within the HSM (one for the encryption algorithm and one for the MAC), are not suitable for the reasons already described, so an efficient mode is required which operates on larger data, but within the constraints of lowest-common-denominator HSM support.

It is to solve this real world cryptographic issue that we turn our attention. The summary requirement is therefore to design a mode of operation which provides a service of authenticated encryption which has the following properties:

– All secret keys used by the mode should reside on the HSM.
– Only one call to the HSM is allowed, i.e. only one application of key material is allowed.
– The only modes of operation available are ECB or CBC-Encrypt with zero IV.

The mode of operation described in this paper has been deployed in several scenarios;

– By major European banks to protect sensitive customer data in transit and storage that does not fall under the existing frameworks for banking PINs or crypto keys.
– By vendors of customised HSM firmware that need a way of offloading cryptographic keys under a master key, held together with access control and

usage information. Code which runs internally within an HSM is often subject to the same limitations of crypto primitive availability as that which make external calls.

Yet this paper represents the first time it has been described publicly and has been provided with a full security analysis:

The main idea is to "authenticate" the message using a "MAC-like" function and then encrypt the result; i.e. to apply a MAC-Then-Encrypt methodology. The encryption is performed using CBC Mode with a zero IV, but a random first block (to achieve probabilistic encryption). This is equivalent to CBC Mode with a random IV given by the first block of ciphertext. However, our MAC-like function must be key-less (to avoid another call to the HSM). So we use a hash function on the message sent to the CBC Mode encryption, this is like applying the hash function as a "MAC" with key given by the initial random plaintext block we added before encryption. However, such a MAC on its own is not secure as it suffers from length extension attacks. Amazingly, despite the use of MAC-Then-Encrypt with an insecure keyless MAC and an insecure encryption scheme (zero-IV CBC mode is not even IND-CPA) the whole construction can be proved secure, assuming the hash function is modelled as a random oracle.

2 Preliminaries

Notation We let $x \| y$ denote the concatenation of two strings x and y. $|x|$ denotes the length of the string x. We use $x \overset{r}{\leftarrow} \mathcal{X}$ to denote the random selection of an element x from the set $\mathcal{X}$. We denote the addition of x to the set $\mathcal{X}$ by $\mathcal{X} \overset{\cup}{\leftarrow} x$.

Pseudorandom Functions and Permutations We define pseudorandom functions and permutations as follows. We also provide a definition for strong PRPs.

Definition 1. ***[Pseudorandom Function/Permutation (PRF/PRP)]*** *Let $F = \{F_K : K \in \{0,1\}^k\}$ where $F_K : \{0,1\}^l \to \{0,1\}^{l'}$ for each $K \in \{0,1\}^k$. Let $P = \{P_K : K \in \{0,1\}^k\}$ where for each $K \in \{0,1\}^k$, $P_K : \{0,1\}^l \to \{0,1\}^l$ is a permutation. Let* Rand *be the set of all functions mapping l-bit strings to l'-bit strings. Let* Perm *be the set of all permutations mapping l-bit strings to l-bit strings. The prf and prp advantage of an adversary $\mathcal{A}$ are defined as:*

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}) = \Pr[K \overset{r}{\leftarrow} \{0,1\}^k : \mathcal{A}^{F_K} \Rightarrow 1] - \Pr[f \overset{r}{\leftarrow} \mathrm{Rand} : \mathcal{A}^f \Rightarrow 1]$$

$$\mathbf{Adv}_P^{\mathrm{prp}}(\mathcal{A}) = \Pr[K \overset{r}{\leftarrow} \{0,1\}^k : \mathcal{A}^{P_K} \Rightarrow 1] - \Pr[\pi \overset{r}{\leftarrow} \mathrm{Perm} : \mathcal{A}^\pi \Rightarrow 1]$$

A function family F (permutation family P resp.) is said a prf (prp resp.) if $\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A})$ ($\mathbf{Adv}_F^{\mathrm{prp}}(\mathcal{A})$ resp.) is "small" for all adversaries $\mathcal{A}$ running in time t making at most q_f queries.

Definition 2. ***[Strong Pseudorandom Permutation (SPRP)]*** *Let $P = \{P_K : K \in \{0,1\}^k\}$ where for each $K \in \{0,1\}^k$, $P_K : \{0,1\}^l \to \{0,1\}^l$ is a permutation with corresponding inverse permutation $P_K^{-1} : \{0,1\}^l \to \{0,1\}^l$. Let*

Perm *be the set of all permutations mapping l-bit strings to l-bit strings. The sprp advantage of an adversary $\mathcal{A}$ is defined as:*

$$\mathbf{Adv}_P^{\text{sprp}}(\mathcal{A}) = \Pr[K \xleftarrow{r} \{0,1\}^k : \mathcal{A}^{P_K, P_K^{-1}} \Rightarrow 1] - \Pr[\pi \xleftarrow{r} \text{Perm} : \mathcal{A}^{\pi,\pi^{-1}} \Rightarrow 1]$$

A permutation family P is said to be an sprp if $\mathbf{Adv}_F^{\text{sprp}}(\mathcal{A})$ is "small" for all adversaries $\mathcal{A}$ running in time t making at most q_f queries to both oracles.

In our analysis we will use PRP/PRFs and SPRPs to represent the block cipher used in our scheme. We will also make use the following lemma to relate PRF to PRP.

Lemma 1. [PRF→ PRP, [2], Proposition 8] *For any permutation family $P = \{P_K : K \in \{0,1\}^k\}$ over l-bit strings.*

$$\mathbf{Adv}_P^{\text{prf}}(\mathcal{A}) \leq \mathbf{Adv}_P^{\text{prp}}(\mathcal{B}) + \frac{q_f^2}{2^{l+1}}$$

Hash function. Let $\mathsf{hash} : \{0,1\}^* \to \{0,1\}^l$ be a hash function with outputs truncated to l-bits (where l is the blocksize of the block cipher used in the scheme). When we denote the hash function with input as $\mathsf{hash}(X, Y, Z)$ we simply compute the hash on the concatenation of the input i.e. $\mathsf{hash}(X\|Y\|Z)$.

CBC mode. An HSM allows us an API call to CBC mode with a zero IV. We therefore define CBC mode accordingly. Let $\mathsf{E\text{-}CBC}^0[F](K, M)$ denote the CBC encryption of message M (with all zero IV) using the function family F under key K, i.e. $\mathsf{E\text{-}CBC}^0[F] : \{0,1\}^k \times \{0,1\}^{ln} \to \{0,1\}^{ln}$, $(n \in \mathbb{N})$, where for $M = M[1]M[2]...M[n]$ we have that $C[i] = F_K(M[i] \oplus C[i-1])$ and $C[0] = 0^l$. Let $\mathsf{D\text{-}CBC}^0[F](K, C)$ denote the CBC decryption of ciphertext C using the function family F under key K, i.e. $\mathsf{D\text{-}CBC}^0[F] : \{0,1\}^k \times \{0,1\}^{ln} \to \{0,1\}^{ln}$, $(n \in \mathbb{N})$, where for $C = C[1]C[2]...C[n]$ and $C[0] = 0^l$ we have $M[i] = F_K^{-1}(C[i]) \oplus C[i-1]$.

It is very important to note that CBC mode with a zero IV is *not* secure. To achieve even IND-CPA security we need to use a random initialisation vector (IV). CBC mode with random IVs was proven secure by Bellare et al. [2]. The scheme that we present will prepend a random block to the plaintext before the CBC call to achieve security. This random block will effectively replace the zero IV of the API encrypt call to make a random IV. The choice of this random block is internal to the scheme and so the adversary should/will not have control over it (this fact is crucial to the schemes security). The random block will either be generated by the HSM or will be prepended to (or xored with) the first plaintext block prior to being called to the HSM.

Padding scheme. When working with arbitrary length messages we need to pad the message before sending it to the blockcipher/CBC mode. Let $\mathsf{pad} : \{0,1\}^* \to \{0,1\}^{ln}$ be the padding function which pads the message to a multiple of the blocksize. Let $\mathsf{dpad} : \{0,1\}^{ln} \to \{0,1\}^{<ln} \cup \{\perp\}$ be the associated depadding function which depads the message, if the message is invalid it returns the symbol

$\perp$. The padding scheme used with the Managed Encryption Format is PKCS#7 padding [11] (add p bytes each of value p). Note that in our analysis we assume that uniform error reported is used so that no padding oracle attacks exist. This is the case in the implementation of the scheme in all known deployed instantiations.

2.1 Security Models

An authenticated encryption scheme is secure if it achieves both the IND-CPA and INT-CTXT notions of security [16,3]. As padding should only pad to the next block boundary and is not variable in length (i.e. we cannot have multiple blocks of padding) we do not consider length-hiding to be a security goal [14]. In implementations of the scheme we stress that uniform error reporting *must* be used. This will be vital for the scheme's security otherwise a padding oracle attack similar to that against SSL/TLS by Canvel et al. [7] may be possible. As a result our analysis we only considers one error type, $\perp$. Let $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be an encryption scheme and $\mathcal{A}$ be an adversary.

```
IND-CPA^A(Π)
    $K \leftarrow \mathsf{KeyGen}$
    $b \xleftarrow{r} \{0,1\}$
    $b' \leftarrow \mathcal{A}^{\mathbf{Enc}}$
    return $(b' = b)$

Enc(A, M_0, M_1)
    $C_0 \leftarrow \mathsf{Encrypt}(K, A, M_0)$
    $C_1 \leftarrow \mathsf{Encrypt}(K, A, M_1)$
    $\mathcal{C} \xleftarrow{\cup} C_b$
    return $C_b$
```

```
INT-CTXT^A(Π)
    $K \leftarrow \mathsf{KeyGen}$
    $\mathsf{win} \leftarrow \mathsf{false}$
    $(A^*, C^*) \leftarrow \mathcal{A}^{\mathbf{Enc},\mathbf{Test}}$
    return win

Enc(A, M)
    $C \leftarrow \mathsf{Encrypt}(K, A, M)$
    $\mathcal{C} \xleftarrow{\cup} (A, C)$
    return $C$

Test(A*, C*)
    $M^* \leftarrow \mathsf{Decrypt}(K, A^*, C^*)$
    if $M^* \neq \perp$ and $(A^*, C^*) \notin \mathcal{C}$ then
        $\mathsf{win} \leftarrow \mathsf{true}$
    return $(M^* \neq \perp)$
```

Fig. 1. The IND-CPA and INT-CTXT experiments

The security experiment for IND-CPA is found in Figure 1. Note, that the encryption scheme semantics are defined to encrypt messages M as well as (possibly public) associated data A. We define the *advantage* of an adversary $\mathcal{A}$ against the IND-CPA security of Π as:

$$\mathbf{Adv}_{\Pi}^{\mathrm{ind-cpa}}(\mathcal{A}) = 2\Pr[\mathbf{IND\text{-}CPA}^{\mathcal{A}}(\Pi) \Rightarrow \mathsf{true}] - 1,$$

We say that the scheme Π is IND-CPA secure if for all adversaries $\mathcal{A}$ the advantage $\mathbf{Adv}_{\Pi}^{\mathrm{ind-cpa}}(\mathcal{A})$ is "small", where the adversary $\mathcal{A}$ makes q_e queries to the encryption oracle $\mathbf{Enc}(A, M_0, M_1)$, totaling at most μ_e bits in each of the left M_0 and right M_1 inputs.

The security experiment for INT-CTXT, is found in Figure 1. We define the *advantage* of an adversary $\mathcal{A}$ against the INT-CTXT security of Π as:

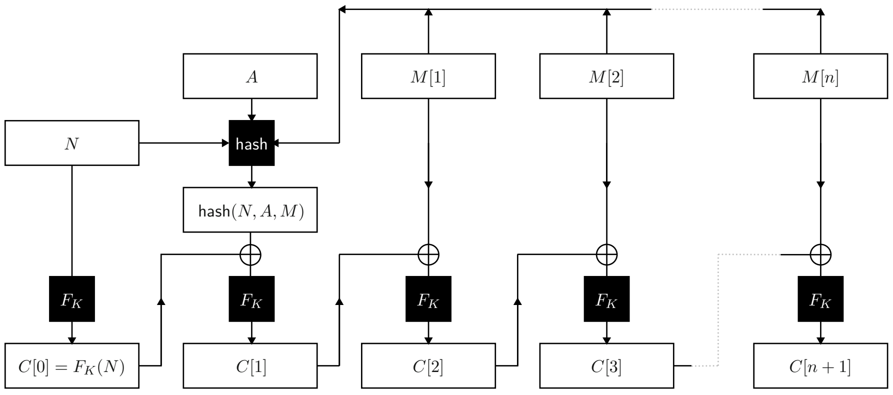
$$\mathbf{Adv}_{\Pi}^{\mathrm{int-ctxt}}(\mathcal{A}) = \Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi) \Rightarrow \mathsf{true}]$$

We say that Π is INT-CTXT secure if for all adversaries $\mathcal{A}$ the advantage $\mathbf{Adv}_{\Pi}^{\mathrm{int-ctxt}}(\mathcal{A})$ is "small", where $\mathcal{A}$ makes q_e queries to the encryption oracle $\mathbf{Enc}(A, M)$, totaling at most μ_e bits in the M input and q_t queries to the test oracle $\mathbf{Test}(A, C)$ totaling at most μ_t of ciphertext bits.

3 Description of the Scheme

The Managed Encryption Format combines CBC mode encryption and a hash function in a MAC-then-Encrypt style configuration. We shall denote the Manage Encryption Format scheme by $\Pi[F]$. The scheme $\Pi[F]$ consists of three algorithms: key-generation KeyGen, encryption $\mathsf{Encrypt}$ and decryption $\mathsf{Decrypt}$. The function F signifies the underlying blockcipher (i.e. the permutation family). We use the following notation: K is a key, N denotes a random IV, A is the header (associated data which is always the same fixed length), M is a (unpadded) message and C is a ciphertext.

To encrypt a message M with associated data A, a random N is first chosen. Then a hash is calculated over N, A, M. Following this CBC encryption (using the block cipher F_K and zero IV) is performed on the (padded) message M, prepended with N and the hash value. Effectively, N ensures that we will have a random IV despite our API crypto call being to CBC mode with a zero IV.



$\underline{\mathsf{KeyGen}(k)}$	$\underline{\mathsf{Encrypt}(K, A, M)}$	$\underline{\mathsf{Decrypt}(K, A, C)}$
$K \xleftarrow{r} \{0,1\}^k$	$N \xleftarrow{r} \{0,1\}^l$	$N \Vert h \Vert M' \leftarrow \mathsf{D\text{-}CBC}^0[F](K, C)$
return K	$h \leftarrow \mathsf{hash}(N, A, M)$	$M \leftarrow \mathsf{dpad}(M')$
	$C \leftarrow \mathsf{E\text{-}CBC}^0[F](K, N \Vert h \Vert \mathsf{pad}(M))$	**if** $M \neq \perp$ **then**
	return C	$\quad \overline{h} \leftarrow \mathsf{hash}(N, A, M)$
		if $\overline{h} \neq h$ **then** $M = \perp$
		return M

Fig. 2. Managed Encryption Format $\Pi[F] = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$

Decryption is the obvious inverse operation of encryption. The diagram and description in Figure 2 are given for aid of analysis.

4 Links with Prior Constructions

4.1 Analysis of the Underlying Message Authentication Code

If we look at the underlying MAC we see that it is of the Wegman-Carter style [18] and is particularly similar to VMAC [9,10]. The VMAC algorithm uses a keyed hash function hash and a prf F. Tags are constructed on a message M together with a nonce N and keys K_1 and K_2 as follows:

$$\tau = \mathsf{hash}_{K_1}(M) + F_{K_2}(N)$$

Notice that when $K_1 = N$ this is almost exactly the MAC we have in the above mode of operation, i.e.

$$\tau = \mathsf{hash}_N(M) \oplus F_K(N)$$

In VMAC we return the nonce N, message M and tag τ but in our MAC we cannot return N since this would break the schemes security instead we must return $F_K(N)$, message M and tag τ. Effectively this means we simply return the message M and $\mathsf{hash}_N(M)$. We now require unforgeability properties from $\mathsf{hash}_N(M)$ but since key N is simply a secret-prefix of the hash function input this falls to extension attacks when used with an iterated hash function [15]. The underlying MAC is therefore not secure on its own.

4.2 Encryption with Redundancy

Bellare and Rogaway [4] and An and Bellare [1] have previously study the problem of how to achieve a secure AE scheme by appending some redundancy to the data before encryption. We shall review the results of An and Bellare [1] here and discuss how to relate these to our work.

A redundancy function may simple take the form of a hash function, as in our scheme. An and Bellare consider two types of redundancy function; one with a secret key and the other where any keying material is public. Perhaps the most important result that they show from our perspective, is that an IND-CPA scheme when combined with either a secret or a public redundancy function is not secure in general. Further to this, they describe an attack on the generic construction which combines CBC mode with a public redundancy function.

Despite this, An and Bellare are able to provide a construction for an encryption scheme which when combined with a secret key redundancy function would achieve INT-CTXT [1, Theorem 6.5]. This construction is called Nested CBC or NCBC. The encryption procedure followed for this scheme is to proceed with CBC encryption as normal until the last block. When encrypting the last block a different key shall be used for the block cipher. This means that NCBC requires two keys and therefore, in the setting of HSMs, would require two key unwrap operations. As a result this construction would not meet the single key requirement of our setting.

The Managed Encryption Format can be viewed within the context of the Encryption with Redundancy paradigm with one main difference. Due to the encryption algorithm's selection of a new N upon each encryption call, this effectively means we are choosing a new random "key" for each hash function call. As a result we can view the Managed Encryption Format as an encryption scheme with secret redundancy, where the redundancy function is chosen anew upon each encryption call. Looking at a general construction of any IND-CPA secure encryption scheme and one time redundancy functions, we would still not necessarily achieve INT-CTXT security. An IND-CPA secure encryption scheme can be constructed, as in the attack presented by An and Bellare in the normal setting [1, Theorem 5.1], that when combined with one time secret redundancy functions, would not achieve INT-CTXT security. Despite this, in the next section we show that when we use a construction based on CBC mode (as in the Managed Encryption Format) we will obtain a secure AE scheme.

5 Security Analysis

In this section we shall prove that the Managed Encryption Format does achieve both IND-CPA security and INT-CTXT security and is therefore a secure authenticated encryption scheme. Note that to simplify our analysis we do not formally discuss the padding which is added to messages. This will not affect our security analysis since uniform-error reporting is used. We will therefore assume that all messages are already padded and omit the padding procedure from our proofs. In practice however the scheme could fall to a padding oracle attack of the style in [7] if uniform-error reporting is not present. Our proof is in the random oracle model (although this is only necessary for the proof of INT-CTXT). We denote by q_h the number of queries the adversary makes to the random oracle (not including those made through encryption, decryption or test queries).

Theorem 1. [IND-CPA] *Let $F = \{F_K : K \in \{0,1\}^k\}$ be a permutation family. Let $\Pi[F]$ be the encryption scheme for the Managed Encryption Format using the permutation family F. Let $\mathcal{A}$ be an adversary against the IND-CPA security which runs in time t; making q_e encryption queries totalling at most μ_e bits. Then there exists an adversary $\mathcal{B}$ such that:*

$$\mathbf{Adv}^{\mathrm{ind-cpa}}_{\Pi[F]}(\mathcal{A}) \leq 2\mathbf{Adv}^{\mathrm{prp}}_{F}(\mathcal{B}) + \frac{q_f^2}{2^l} + \frac{1}{2^l}\left(\left(\frac{\mu_e}{l} + 2q_e\right)^2 - \left(\frac{\mu_e}{l} + 2q_e\right)\right)$$

where $\mathcal{B}$ runs in time $t + O(\mu_e)$ asking at most $q_f = \frac{\mu_e}{l} + 2q_e$ queries.

```
Encrypt0$(K, A, M)$
$N \xleftarrow{r} \{0,1\}^l$
$h \leftarrow \mathsf{hash}(N, A, M)$
$C \leftarrow \mathsf{E\text{-}CBC}^0[F](K, N\|h\|M)$
return $C$
```

```
Encrypt1$(K, A, M)$
$N \xleftarrow{r} \{0,1\}^l$
$h \leftarrow \mathsf{hash}(N, A, M)$
$C \leftarrow \mathsf{E\text{-}CBC}^0[F](K, N\|h\|M)$
return $N\|C$
```

```
Encrypt1$^{\mathsf{rand}}(K, A, M)$
$N \xleftarrow{r} \{0,1\}^l$
$h \leftarrow \mathsf{hash}(N, A, M)$
$C \leftarrow \mathsf{E\text{-}CBC}^0[R](K, N\|h\|M)$
return $N\|C$
```

Fig. 3. Encryption Algorithm Hops in Proof of Lemma 1

Proof. This can be proven by extension to the existing proof of security for CBC mode by Bellare et al. [2]. Since N is chosen uniformly at random by the encryption algorithm this gives us the necessary randomness for the existing CBC mode proof to still hold. Our proof follows a series of game hops. The different encryption algorithms used in each hop are shown in Figure 3.

Let Game0 be the normal IND-CPA game where $\mathcal{A}$ has access to a left-or-right oracle that uses algorithm Encrypt0(K, A, M) to encrypt M_b.

Let Game1 be the same as Game0 but replace the encryption algorithm Encrypt0(K, A, M) used by the left-or-right oracle with Encrypt1(K, A, M). In Encrypt1(K, A, M) the value of N is now returned with the ciphertext. Knowledge of N allows the adversary to recalculate any hash values $\mathsf{hash}(N, A, M)$. This means that in the left-or-right indistinguishability game the adversary now has retrospective knowledge of his query, i.e. his left-or-right encryption is effectively $(0^l\|\mathsf{hash}(N, A, M_0)\|M_0, 0^l\|\mathsf{hash}(N, A, M_1)\|M_1)$, which will be encrypted by CBC mode with random IV N. Since N is chosen at random for each call, giving the adversary knowledge of a previously used N will not allow the adversary to predict any future N'. We therefore have that

$$\Pr[\mathsf{Game1} \Rightarrow \mathsf{true}] = \Pr[\mathsf{Game0} \Rightarrow \mathsf{true}].$$

Now we effectively have normal CBC encryption with a random IV given by N, plus an encryption query of the form $0^l\|h\|M$ (note that in Figure 3 we have already xored on the IV N to the first block, as the internal algorithm called is CBC mode with zero-IV).

It therefore remains to analyse $\mathcal{A}$'s success probability in Game1. This can be proven by extension to the existing proof of security for CBC mode by Bellare et al. [2]. Note that we cannot perform a direct reduction because prior knowledge of N is necessary to calculate the hash. We begin the proof of security by switching to consider F as a random function. Let Game1$^{\mathsf{rand}}$ be exactly the same as Game1 but we now replace F_K with a function drawn uniformly at random from the set of all functions mapping l-bit strings to $l' = l$-bit strings (i.e. $f \overset{r}{\leftarrow} \text{Rand} = R$). We can then construct a distinguisher $\mathcal{B}$ as in [2] such that:

$$\Pr[\mathsf{Game1} \Rightarrow \mathsf{true}] - \Pr[\mathsf{Game1}^{\mathsf{rand}} \Rightarrow \mathsf{true}] \leq 2\mathbf{Adv}_F^{\text{prf}}(\mathcal{B})$$

We shall later use Lemma 1 to consider the prp advantage.

We now examine $\Pr[\mathsf{Game1}^{\mathsf{rand}} \Rightarrow \mathsf{true}]$. $\mathsf{Game1}$ proceeds as in the original proof for CBC mode by Bellare et al. [2], the encryption algorithm randomly chooses the IV N, and then proceeds with CBC encryption. Bellare et al.'s proof shall therefore remain almost as is and we just give a brief summary here. Note now that we simply treat the hash h like an additional plaintext block.

We denote the left and right encryption queries as follows:

Let $\tilde{M}^L = 0^l \| \mathsf{hash}(N, A, M_0) \| M_0$ denote the left query and let $\tilde{M}^R = 0^l \| \mathsf{hash}(N, A, M_1) \| M_1$ denote the right query. We also let $C_j[k]$ denote the k-th block of the j-th ciphertext, $\tilde{M}_j^L[k]$ denote the k-th block of the j-th left query and $\tilde{M}_j^R[k]$ denote the k-th block of the j-th right query. For a fixed adversary $\mathcal{A}$ we define the event $\mathsf{D}_{i,u}$, where $i \in [q_e]$ and $u \in [n_i + 2]$, to be when the following two events occur:

$$C_j[k-1] \oplus \tilde{M}_j^L[k] \neq C_{j'}[k'-1] \oplus \tilde{M}_{j'}^L[k']$$

$$C_j[k-1] \oplus \tilde{M}_j^R[k] \neq C_{j'}[k'-1] \oplus \tilde{M}_{j'}^R[k']$$

for all $(j,k), (j',k') \in \{(j,k) : j \in [q_e] \text{ and } k \in [n_j + 2]\}$ satisfying $(j',k') \prec (j,k) \preceq (i,u)$, where n_j denotes the number of blocks in the j-th encryption query, note the addition of 2 here due to the extra all-zero block and hash block. Here $\prec$ denotes an ordering on the blocks queried to the encryption oracle. With $(j',k') \prec (j,k)$ implying that the k'-th block of the j'-th ciphertext was queried before the k-th block of the j-th ciphertext.

We now wish to study the probability of the event $\overline{\mathsf{D}} = \overline{\mathsf{D}_{q,n_q}}$, i.e. that a collision occurs at some point in the output ciphertexts. Bellare et al. prove that $\Pr[\overline{\mathsf{D}}|b=0] = \Pr[\overline{\mathsf{D}}|b=1]$, i.e. this probability is independent of whether we are observing left queries or right queries, the analysis is therefore the same for both $b = 0$ and $b = 1$. This probability is then denoted $p = \Pr[\overline{\mathsf{D}}|b=0] = \Pr[\overline{\mathsf{D}}|b=1]$ and was proved to be bounded as follows.

$$\begin{aligned} p &= \sum_{i=1}^{q_e} \sum_{j=1}^{n_i} \Pr[\overline{\mathsf{D}_{i,u}} | \mathsf{D}_{i,u-1}] \\ &\leq \frac{1}{2^l} \left(\left(\frac{\mu_e}{l} + 2q_e \right)^2 - \left(\frac{\mu_e}{l} + 2q_e \right) \right) \end{aligned}$$

Note that we adjust slightly from the original bound since we must account for the additional plaintext blocks caused by the hash and the initial all-zero block.

Combining the above and following similar arguments to the original CBC proof by Bellare et al. (along with Lemma 1) we obtain the following:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ind-cpa}}_{\Pi[F]}(\mathcal{A}) &= \Pr[\mathsf{Game0} \Rightarrow \mathsf{true}]\\ &\leq \Pr[\mathsf{Game1} \Rightarrow \mathsf{true}]\\ &\leq 2\mathbf{Adv}^{\mathrm{prf}}_{F}(\mathcal{B}) + \Pr[\mathsf{Game1}^{\mathsf{rand}} \Rightarrow \mathsf{true}]\\ &\leq 2\mathbf{Adv}^{\mathrm{prf}}_{F}(\mathcal{B}) + \frac{1}{2^l}\left(\left(\frac{\mu_e}{l}+2q_e\right)^2 - \left(\frac{\mu_e}{l}+2q_e\right)\right)\\ &\leq 2\mathbf{Adv}^{\mathrm{prp}}_{F}(\mathcal{B}) + \frac{q_f^2}{2^l} + \frac{1}{2^l}\left(\left(\frac{\mu_e}{l}+2q_e\right)^2 - \left(\frac{\mu_e}{l}+2q_e\right)\right).\end{aligned}$$

Theorem 2. [INT-CTXT] *Let $F = \{F_K : K \in \{0,1\}^k\}$ be a permutation family. Let $\Pi[F]$ be the encryption scheme for the Managed Encryption Format using the permutation family F. Let $\mathcal{A}$ be an adversary against the INT-CTXT security which runs in time t; making q_e encryption queries totalling at most μ_e bits, q_t test queries totalling at most μ_t bits and q_h random oracle queries. Then there exists an adversary $\mathcal{B}$ such that:*

$$\mathbf{Adv}^{\mathrm{int-ctxt}}_{\Pi[F]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{sprp}}_{F}(\mathcal{B}) + \frac{q_t}{2^l} + \frac{q_h \mu_e}{l 2^l}$$

where $\mathcal{B}$ makes $q_f = \frac{\mu_e}{l} + 2q_e + \frac{\mu_t}{l}$ queries and runs in time $t + O(\mu_e + \mu_t)$.

Proof. We assume we have an adversary $\mathcal{A}$ against INT-CTXT and we use it to construct an adversary $\mathcal{B}$ against SPRP. This is done as follows:

- When $\mathcal{A}$ makes an encryption query A, M, the algorithm $\mathcal{B}$ chooses N at random and calls the random oracle on N, A, M. Following this $\mathcal{B}$ makes calls to its π oracle for the appropriate CBC encryption (making a total of $\mu_e/l + 2q_e$ queries (where $2q_e$ accounts for the queries $\pi(N)$ and $\pi(h \oplus \pi(N))$). Finally $\mathcal{B}$ returns the ciphertext to $\mathcal{A}$.
- When $\mathcal{A}$ makes a test query, the algorithm $\mathcal{B}$ calls its π^{-1} oracle for the appropriate CBC decryption (μ_t/l queries). Then $\mathcal{B}$ verifies whether M is new or not. Next $\mathcal{B}$ calls the random oracle to verify the hash. The result of the whole verification is returned to $\mathcal{A}$.
- The random oracle maintains a list $\mathcal{H}$ of all queries.
- If $\mathcal{A}$ outputs a successfully forgery then $\mathcal{B}$ guesses it has access to the real permutation.
- If $\mathcal{A}$ is unsuccessful then $\mathcal{B}$ guesses it has access to the random permutation.

The following inequality then holds (where Perm is the set of all l-bit permutations):

$$\begin{aligned}\mathbf{Adv}^{\mathrm{sprp}}_{F}(\mathcal{B}) &\geq \Pr[\mathcal{B}^{\pi,\pi^{-1}} \Rightarrow 1 | \pi \xleftarrow{r} F] - \Pr[\mathcal{B}^{\pi,\pi^{-1}} \Rightarrow 1 | \pi \xleftarrow{r} \mathrm{Perm}].\\ &= \Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi[F]) \Rightarrow \mathsf{true}]\\ &\qquad - \Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi[\mathrm{Perm}]) \Rightarrow \mathsf{true}]\\ &= \mathbf{Adv}^{\mathrm{int-ctxt}}_{\Pi[F]}(\mathcal{A}) - \Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi[\mathrm{Perm}]) \Rightarrow \mathsf{true}].\end{aligned}$$

If $\mathsf{Decrypt}(K, A^*, C^*) = M^*$ then the probability that this message is a valid forgery is bounded by the probability that the hash verifies on M^*. The decrypted hash value to be verified will be given by $h^* = \pi^{-1}(C^*[1]) \oplus C^*[0] = \pi^{-1}(C^*[1]) \oplus \pi(N^*)$. We therefore obtain the following bound:

$$\Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi[\mathrm{Perm}]) \Rightarrow \mathsf{true}] \leq \Pr[\mathsf{hash}(N^*, A^*, M^*) = h^* | \pi \xleftarrow{r} \mathrm{Perm}]$$

We shall now consider this probability in two parts. First we study the case where N^*, A^*, M^* was never a random oracle query, i.e. $(N^*, A^*, M^*, h^*) \notin \mathcal{H}$. The actual hash value will now be chosen at random when the random oracle is called upon decryption. The probability that this hash collides with the decrypted value $h^* = \pi^{-1}(C^*[1]) \oplus C^*[0]$ is $\frac{1}{2^l}$ (for a single test query). We therefore obtain the following bound:

$$\Pr[(\mathsf{hash}(N^*, A^*, M^*) = h^*) \wedge ((N^*, A^*, M^*, h^*) \notin \mathcal{H}) | \pi \xleftarrow{r} \mathrm{Perm}] \leq \frac{q_t}{2^l}$$

Next consider the case when N^*, A^*, M^* has been previously called to the random oracle. We shall prove the following bound:

$$\Pr[(\mathsf{hash}(N^*, A^*, M^*) = h^*) \wedge ((N^*, A^*, M^*, h^*) \in \mathcal{H}) | \pi \xleftarrow{r} \mathrm{Perm}] \leq \frac{q_h \mu_e}{l 2^l}.$$

Since $(N^*, A^*, M^*, h^*) \in \mathcal{H}$, the query was already a call to hash but it cannot have been made by a previous encryption query. This is because a hash called previously by $\mathsf{Encrypt}$ on (N^*, A^*, M^*) would imply that C^* was already output by $\mathsf{Encrypt}$, breaking the restrictions of the INT-CTXT game. The query must have therefore been made by a separate call to hash.

Consider the query N^*, A^*, M^* which $\mathcal{A}$ makes to hash (receiving h^*). $\mathcal{A}$ can choose N^* such that it has seen its encryption $\pi(N^*)$ in a previous encryption query, i.e. $N^* = C_i[j] \oplus M_i[j]$ for some $i \in [q_e]$ and $j \in [n_i]$, where q_e is the total number of encryption queries and n_i is the number of blocks in query i. (Note here that it may look odd that $C_i[j]$ and $M_i[j]$ both have the same block index j but we stress this is still a normal CBC encryption step. The apparent difference is due to the hash h shifting the block indices of M_i; to see this more easily we refer the reader to Figure 2.)

To forge a valid ciphertext $\mathcal{A}$ must first ensure that the first ciphertext block is correct, i.e. $C^*[1] = \pi(h^* \oplus \pi(N^*))$. Since π is a random permutation $\mathcal{A}$ will choose $C^*[1]$ correctly only if it has seen $\pi(h^* \oplus \pi(N^*))$ before, i.e. there exists a call to π where $C_i[j] \oplus M_i[j] = h^* \oplus \pi(N^*)$ for some $i \in [q_e]$ and $j \in [n_i]$.

If $\mathcal{A}$ makes q_e encryption queries totaling μ_e bits, then the probability that h^* is generated by the random oracle such that $C_i[j] \oplus M_i[j]$ is queried to π for some $i \in [q_e]$ and $j \in [n_i]$, is $\frac{\mu_e / l}{2^l}$. The above probability bound then follows.

Combining all of the above we obtain our result.

$$\begin{aligned}\mathbf{Adv}^{\mathrm{int-ctxt}}_{\Pi[F]}(\mathcal{A}) &\leq \mathbf{Adv}^{\mathrm{sprp}}_{F}(\mathcal{B}) + \Pr[\mathbf{INT\text{-}CTXT}^{\mathcal{A}}(\Pi[\mathrm{Perm}) \Rightarrow \mathsf{true}] \\ &\leq \mathbf{Adv}^{\mathrm{sprp}}_{F}(\mathcal{B}) + \Pr[\mathsf{hash}(N^*, A^*, M^*) = h^* | \pi \xleftarrow{r} \mathrm{Perm}] \\ &\leq \mathbf{Adv}^{\mathrm{sprp}}_{F}(\mathcal{B}) + \frac{q_t}{2^l} + \frac{q_h \mu_e}{l \cdot 2^l}.\end{aligned}$$

```
$\mathsf{KeyGen}^c(k)$
$K \xleftarrow{r} \{0,1\}^k$
$ctr \xleftarrow{r} \{0,1\}^l$
return $K$
```

```
$\mathsf{Encrypt}^c(K, A, M)$
$h \leftarrow \mathsf{hash}(ctr, A, M)$
$C \leftarrow \text{E-CBC}^0[F](K, ctr \| h \| \mathsf{pad}(M))$
$ctr \leftarrow ctr + 1$
return $C$
```

```
$\mathsf{Decrypt}^c(K, A, C)$
$ctr \| h \| M' \leftarrow \text{D-CBC}^0[F](K, C)$
$M \leftarrow \mathsf{dpad}(M')$
if $M \neq \perp$ then
    $\overline{h} \leftarrow \mathsf{hash}(ctr, A, M)$
if $\overline{h} \neq h$ then $M = \perp$
return $M$
```

Fig. 4. Man Enc Format with Counter, $\Pi^c[F] = (\mathsf{KeyGen}^c, \mathsf{Encrypt}^c, \mathsf{Decrypt}^c)$

6 Further Discussions

6.1 Using a Counter

We also introduce a stateful version of the Managed Encryption Format as defined in Figure 4. Here we now replace the value N with a counter ctr. It is possible to prove CBC mode is IND-CPA secure with a ctr in this way [2], when a maximum of 2^l encryptions are permitted. Security is ensured by the fact that a collision $ctr^* = C_i[j-2] \oplus M_i[j]$ which reveals the value of $F_K(ctr^*)$ for some future ctr^*, occurs with small probability. Furthermore, if ctr is initialised as a random string then an adversary must first determine the current version of ctr in order to mount an attack based on the above collision. We omit further details of the proof but it can be seen that we will be able to extend this to prove that the stateful version of the Managed Encryption Format would be a secure AE scheme. The application of such a scheme to the setting of HSMs would of course depend on an HSM's ability to maintain state.

6.2 Parameter Choices

We note that the security bounds in our Theorems come with error terms of the order of

$$\frac{q_f^2}{2^l} \quad \text{and} \quad \frac{q_f \cdot q_h}{2^l}.$$

We recall that l is the block length of the underlying block cipher, q_f is the number of queries to the underlying PRP and q_h is the number of hash function queries. These bounds mean that if we wish to guarantee security against the probability of an adversary breaking the scheme not exceeding 2^{-40} (say), and we assume breaking the underlying PRP F is hard, then the number of queries made to the hash function and managed encryption scheme needs to be bounded.

If using DES as the underlying block cipher, where $l = 64$, this means that we need to ensure that $q_f \ll 2^{12} = 4096$. Thus use with DES can be deemed to be insecure, unless underlying block cipher keys are updated relatively quickly. When used with a block cipher such as AES, where $l = 128$, the number of queries to the underlying block cipher needs to be bounded by 2^{44} if one wishes to make the probability of breaking the scheme be bounded by 2^{-40}; whilst the product of the number of block cipher calls multiplied by the number of hash function calls needs to be bounded by 2^{88} to obtain a similar probability bound. Thus the scheme can be considered secure in practice when instantiated with AES, but needs to be used with care when instantiated with DES.

7 Conclusion

We have presented a new provably secure mode of operation for authenticated encryption. This mode has been designed for use in environments where keys are protect by an HSM but the API offers limited cryptographic functions. The scheme is built around an HSM which provides an API call to CBC mode with zero IV. To minimise expensive HSM calls the scheme uses only one key and hence makes a single call to the HSM.

Acknowledgements. The first author thanks his colleagues at Cryptomathic and the Computer Laboratory in Cambridge for useful conversations during the design process. The second author thanks his employers Barclays PLC for their support The third and fourth author were supported by Prof Smart's ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO. The third author was also partially supported by a Royal Society Wolfson Merit Award.

The work in this paper arose from a discussion held during eCrypt-2 sponsored workshop "Is Cryptographic Theory Practically Relevant?" held at the Newton Institute in January 2012. The authors thank eCrypt-2 and the Newton Institute for hosting this workshop. The authors also thank Kenny Paterson and Jean Paul Degabriele for helpful discussions.

References

1. An, J.H., Bellare, M.: Does Encryption with Redundancy Provide Authenticity? In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 512–528. Springer, Heidelberg (2001)
2. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS, pp. 394–403. IEEE Computer Society (1997)
3. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto [13], pp. 531–545
4. Bellare, M., Rogaway, P.: Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In: Okamoto [13], pp. 317–330
5. Bellare, M., Rogaway, P., Wagner, D.: The EAX Mode of Operation. In: Roy, B.K., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 389–407. Springer, Heidelberg (2004)
6. Bond, M.: Attacks on Cryptoprocessor Transaction Sets. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 220–234. Springer, Heidelberg (2001)
7. Canvel, B., Hiltgen, A.P., Vaudenay, S., Vuagnoux, M.: Password Interception in a SSL/TLS Channel. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 583–599. Springer, Heidelberg (2003)

8. Focardi, R., Luccio, F.L., Steel, G.: An Introduction to Security API Analysis. In: Aldini, A., Gorrieri, R. (eds.) FOSAD 2011. LNCS, vol. 6858, pp. 35–65. Springer, Heidelberg (2011)
9. Krovetz, T.: Message Authentication on 64-Bit Architectures. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 327–341. Springer, Heidelberg (2007)
10. Krovetz, T., Dai, W.: Vmac: Message authentication code using universal hashing. CFRG Working Group INTERNET-DRAFT (April 2007), http://www.fastcrypto.org/vmac/draft-krovetz-vmac-01.txt
11. RSA Laboratories. PKCS #7: Cryptographic message syntax standard, Version 1.5 (November 1993)
12. McGrew, D.A., Viega, J.: The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355. Springer, Heidelberg (2004)
13. Okamoto, T. (ed.): ASIACRYPT 2000. LNCS, vol. 1976. Springer, Heidelberg (2000)
14. Paterson, K.G., Ristenpart, T., Shrimpton, T.: Tag Size *Does* Matter: Attacks and Proofs for the TLS Record Protocol. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 372–389. Springer, Heidelberg (2011)
15. Preneel, B., van Oorschot, P.C.: MDx-MAC and Building Fast MACs from Hash Functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)
16. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri, V. (ed.) ACM Conference on Computer and Communications Security, pp. 98–107. ACM (2002)
17. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: Reiter, M.K., Samarati, P. (eds.) ACM Conference on Computer and Communications Security, pp. 196–205. ACM (2001)
18. Wegman, M.N., Carter, L.: New hash functions and their use in authentication and set equality. J. Comput. Syst. Sci. 22(3), 265–279 (1981)
19. Whiting, D., Housley, R., Ferguson, N.: Counter with CBC-MAC (CCM). RFC 3610 (Informational) (September 2003)

A Fully Homomorphic Cryptosystem with Approximate Perfect Secrecy

Michal Hojsík⋆ and Veronika Půlpánová

Department of Algebra, Faculty of Mathematics and Physics
Charles University in Prague, Czech Republic

Abstract. We propose a new fully homomorphic cryptosystem called *Symmetric Polly Cracker* (SymPC) and we prove its security in the information theoretical settings. Namely, we prove that SymPC approaches perfect secrecy in bounded CPA model as its security parameter grows (which we call approximate perfect secrecy).

In our construction, we use a Gröbner basis to generate a polynomial factor ring of ciphertexts and use the underlying field as the plaintext space. The Gröbner basis equips the ciphertext factor ring with a multiplicative structure that is easily algorithmized, thus providing an environment for a fully homomorphic cryptosystem.

Keywords: Polly Cracker, Fully homomorphic encryption, Gröbner bases.

1 Introduction

In 1994 Fellows and Koblitz presented a general outline for a construction of a public-key cryptosystems based on NP-hard problems in [1]. As an example, they described a cryptosystem based on the ideal membership problem and named it Polly Cracker. A whole family of cryptosystems based on this construction has been developed over the following years ([2],[3]). Polly Cracker has also played a critical role in the development of homomorphic encryption theory, mostly serving as a base stone on which more sophisticated systems were built. For instance, Craig Gentry's seminal work on fully homomorphic encryption system [4] was inspired by Polly Cracker. Ever since Gentry's paper has been published, there has been an extensive research in the area, e.g. [5], [6]. Majority of the schemes that followed the outbreak of fully homomorphic encryption have its security based on problems over lattices, such as Learning with Errors (LWE) [7] and most of the research focuses on the public key encryption.

In 2011, Albrecht et al. published a paper "Polly Cracker Revisited" [8]. It formally treats the security of certain classes of Polly Cracker-based cryptosystems and suggests particular transitions between public-key and symmetric versions of Polly Cracker-based systems. In the same paper, Albrecht et al. introduce the Polly Cracker with Noise (CPN) cryptosystem. Only recently, Herold has shown

⋆ The author was supported by grant VF20102015006.

in [9] that the CPN with zero-degree noise from [8] is either insecure or does not offer any security benefit compared to Regev's LWE-based scheme [7].

Our Contribution. In our work, we take a different approach. We propose a new fully homomorphic cryptosystem called *Symmetric Polly Cracker* (SymPC) and we prove its security in information theoretical settings - we prove that SymPC approaches perfect secrecy in bounded CPA model as its security parameter grows. More precisely, we define *approximate perfect secrecy* as the security of a cryptosystem $CS(t) = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ with security parameter t for which the probability $\Pr[P = p \mid C = c]$ approaches $\Pr[P = p]$ for all $p \in \mathcal{P}$, all probability distributions on $\mathcal{P}$ and almost all c as t grows to infinity. Then we prove that SymPC has approximate perfect secrecy in bounded CPA model.

In our construction, unlike in the previous classical Polly Cracker constructions, we use a Gröbner basis G to generate a zero-dimensional ideal $\langle G \rangle$ of a polynomial ring $\mathbb{F}[x_1, \ldots, x_n]$ over a finite field $\mathbb{F}$. Then we use the factor ring $\mathbb{F}[x_1, \ldots, x_n]/\langle G \rangle$ as the ciphertext space and the field $\mathbb{F}$ as the plaintext space. The Gröbner basis G equips the ciphertext factor ring with a multiplicative structure that is easily algorithmized, thus providing an environment for a fully homomorphic cryptosystem. The fully homomorphic property of our cryptosystem is achieved by a simple decryption operation - evaluation homomorphism.

This paper is organized as follows. In Sect. 2 we introduce our notation and state some known facts. In Sect. 3 we describe one instantiation of Polly Cracker cryptosystem. Then we describe our cryptosystem SymPC in Sect. 4 where we also prove that it is fully homomorphic. This is followed by the complexity analysis in Sect. 5. Finally, in Sect. 6, we define the approximate perfect secrecy, give the security proof of SymPC in bounded CPA model and briefly analyze SymPC in other attack scenarios.

2 Preliminaries and Notation

Let q be a prime power. By $\mathbb{F}$ we will denote the finite field $\mathrm{GF}(q)$. In this paper, we will work with the multivariate polynomial ring $\mathcal{R} = \mathbb{F}[x_1, \ldots, x_n]$, $n \in \mathbb{N}$ and operations $+, -$ and $\cdot$ on polynomials will always denote operations in $\mathcal{R}$. Later on, we will define a factor ring $\mathcal{C} = \mathcal{R}/I$ for an ideal I. We will denote the operations in this factor ring as $+_{\mathcal{C}}, -_{\mathcal{C}}$ and $\cdot_{\mathcal{C}}$. Furthermore, we endow $\mathcal{R}$ with an admissible monomial ordering $<$. For $f \in \mathcal{R}$, $\deg(f)$ will denote the total degree of f, i.e. degree of the leading term of f with respect to $<$. The maximum degree of variable x_i in any term of f will be denoted $\deg_{x_i}(f)$.

Let G be a basis of an ideal I in $\mathcal{R}$, i.e. $\langle G \rangle = I$. Recall that G is a Gröbner basis, iff for all $f \in \mathcal{R}$, the remainder on division of f by G is unique. For $f, g \in \mathcal{R}$ define the s-polynomial as $\mathrm{spol}(f, g) = \mathrm{lcm}(\mathrm{lt}(f), \mathrm{lt}(g)) \cdot f/\mathrm{lm}(f) - \mathrm{lcm}(\mathrm{lt}(f), \mathrm{lt}(g)) \cdot g/\mathrm{lm}(g)$, where $\mathrm{lt}(f)$ denotes the leading term of f and $\mathrm{lm}(f)$ the leading monomial of f with respect to $<$. The following theorem is employed in Buchberger's algorithm and we will use it to prove that a given set is a Gröbner basis. The proof can be found in [10].

Theorem 1. *$G \subset \mathcal{R}$ is a Gröbner basis of an ideal $I = \langle G \rangle$, iff the remainder on division of* $\operatorname{spol}(f, g)$ *by G equals zero for all $f, g \in G,\ f \neq g$.*

A Gröbner basis G is called reduced, iff for all $g \in G$ it holds $g \bmod G \setminus \{g\} = g$ and it is called normed, iff all $g \in G$ are monic. A well known theorem states, that for every ideal I in $\mathcal{R}$, there exists a unique normed reduced Gröbner basis G of I.

In descriptions of algorithms, we will use $x \overset{R}{\longleftarrow} X$ to denote that x is chosen uniformly at random from a finite set X.

In this paper, we propose a fully homomorphic probabilistic cryptosystem. By fully homomorphic we mean the usual concept where both the plaintext and the ciphertext sets are equipped with addition and multiplication, they both form rings and the decryption operation is a ring homomorphism (i.e. $d_k(f(c_1, \ldots, c_l)) = f(d_k(c_1), \ldots, d_k(c_l))$ for any polynomial f):

Definition 2. *Let $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \{e_k\}, \{d_k\})$ be a probabilistic cryptosystem, where $\mathcal{P}(+, -, \cdot, 0, 1)$ is the plaintext ring and $\mathcal{C}(+, -, \cdot, 0, 1)$ is the ciphertext ring. We call the cryptosystem fully homomorphic, iff for all $k \in \mathcal{K}$, the decryption operation $d_k : \mathcal{C} \to \mathcal{P}$ is a ring homomorphism.*

3 Polly Cracker

In this section, we will describe one instantiation of Polly Cracker. This scheme has inspired our cryptosystem, which we present in the next section. We denote $\mathcal{S} = \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle$.

Algorithms 1, 2 and 3 describe the Polly Cracker cryptosystem. The set of messages is $\mathcal{P} = \mathbb{F}$, the set of ciphertexts is $\mathcal{C} = \mathcal{R}/\mathcal{S}$ and the keys $K \in \mathcal{K}$ are pairs $(\boldsymbol{s}, PK)$, where the secret key $\boldsymbol{s}$ is a vector in $\mathbb{F}^n$ and the public key $PK = \{f_1, \ldots, f_k\}$ is a set of polynomials in $\mathcal{R}/\mathcal{S}$ of degree at most ν, such that in $\boldsymbol{s}$ they all evaluate to zero, as described in the `SETUP` by Algorithm 1.

Algorithm 1. Polly Cracker: SETUP

```
   Input: $n, k, q, \nu \in \mathbb{N},\ q$ prime power, $\nu < q - 1$
   Output: $(\boldsymbol{s}, PK),\ \boldsymbol{s} \in \mathbb{F}^n, PK \subset \mathcal{R}$
1  set $\mathbb{F} := \mathbb{F}_q$
2  set $\mathcal{R} := \mathbb{F}[x_1, \ldots, x_n]$
3  set the secret key $\boldsymbol{s} = (s_1, ..., s_n) \overset{R}{\longleftarrow} \mathbb{F}^n$
4  for $j = 1$ to $k$ do
5      $f_j \overset{R}{\longleftarrow} \mathcal{R}$ s.t. $\forall i\ \deg_{x_i}(f_j) \leq \nu$ and $f_j(\boldsymbol{s}) = 0$
6  set the public key $PK := \{f_1, ..., f_k\}$
7  set $\mathcal{C} := \mathcal{R}/\mathcal{S}$
8  return $(\boldsymbol{s}, PK)$
```

Algorithm 2 describes encryption. A random subset of the polynomials from PK is added to a message $m \in \mathbb{F}$ to get a ciphertext polynomial $c \in \mathcal{R}/\mathcal{S}$.

Algorithm 2. Polly Cracker: ENCRYPT

```
  Input: message $m \in \mathbb{F}$, public key $PK = \{f_1 \ldots, f_k\} \subset \mathcal{R}$
  Output: ciphertext $c \in \mathcal{R}$
1 select $I \subseteq \{1, .., k\}$ uniformly at random
2 set the ciphertext $c := m + \sum_{j \in I} f_j \in \mathcal{R}$
3 return $c$
```

The decryption is given by Alg. 3. It evaluates the ciphertext polynomial c in the secret key $\boldsymbol{s}$. It is easy to see, that if $c = e_{PK}(m)$, then $c(\boldsymbol{s}) = m$, as $f(\boldsymbol{s}) = 0$ for all $f \in PK$.

Algorithm 3. Polly Cracker: DECRYPT

```
  Input: ciphertext $c \in \mathcal{R}$, secret key $\boldsymbol{s} \in \mathbb{F}^n$
  Output: message $m \in \mathbb{F}$
1 set $m := c(\boldsymbol{s})$
2 return $m$
```

Using the Fundamental theorem on homomorphism and the fact that evaluation of polynomials is a ring homomorphism, one can show that decryption operation is a ring homomorphism on $\mathcal{R}/\mathcal{S}$. Hence Polly Cracker is a fully homomorphic cryptosystem. A disadvantage is, that the size of a ciphertext grows rapidly with the number of multiplications, which is not practical. However, we work with the ring $\mathcal{R}/\mathcal{S}$, which is finite, so after about $\frac{q}{\nu}$ multiplications the resulting ciphertexts stop growing.

Nevertheless, the size of a random polynomial in $\mathcal{R}/\mathcal{S}$ is $O(q^n)$ bits. (The size of a random polynomial in $\mathcal{R}/\mathcal{S}$ is $\log_2(|\mathcal{R}/\mathcal{S}|) = \log_2\left(q^{q^n}\right)$.) Hence one needs to keep the number of variables very low in order to get a reasonable ciphertext size.

Unfortunately, the Polly Cracker cryptosystem can be attacked by calculating the Gröbner basis of the ideal generated by PK. If an adversary has a set $\{g_1, \ldots, g_l\}$, the Gröbner basis of $\langle PK \rangle$, then for any $c \in \mathcal{C}$ he can calculate $c \bmod \{g_1, \ldots, g_l\}$ and as a result he will get $d_{\boldsymbol{s}}(c) = c(\boldsymbol{s})$, i.e. the plaintext.

4 Symmetric Polly Cracker (SymPC)

In this section, we propose a new fully homomorphic probabilistic symmetric cryptosystem called *Symmetric Polly Cracker - SymPC*.

The cryptosystem SymPC is described by Algorithms 4, 5, 6, 7 and 8. Algorithm 4 describes `SETUP`, which takes security parameters n, q, ν and returns a pair $(\boldsymbol{s}, G)$, where $\boldsymbol{s} \in \mathbb{F}^n$ is the secret key and $G = \{g_1, \ldots, g_n\} \subset \mathcal{R}$ is the multiplication key. This is a special kind of key, that is only used in the multiplication of ciphertexts. It provides information about the ring of ciphertexts $\mathcal{R}/\langle G \rangle$ and we will assume that it is public. Furthermore `SETUP` defines the set (field) of plaintexts as $\mathbb{F}$ and the set (ring) of ciphertexts $\mathcal{C}$ as the factor ring $\mathcal{R}/\langle G \rangle$. The choice of polynomials g_i in Step 8 has some important consequences. First, as we shall see in Sect. 6, this choice maximizes the size of $V(G)$, the algebraic set of G. This leads to optimal security for a given security parameter. Second, G is the reduced normed Gröbner basis of $\langle G \rangle$ (the proof can be found in Appendix A):

Theorem 3. *Let $\mathcal{R}$ and G be defined by Alg. 4. Then G is the reduced normed Gröbner basis of the ideal $\langle G \rangle$.*

Algorithm 4. SETUP

```
   Input: $n, \nu, q \in \mathbb{N}, \nu < q-1,\ q$ a prime power
   Output: $\boldsymbol{s} \in \mathbb{F}^n, G \subset \mathcal{R}$
 1 set $\mathbb{F} := \mathbb{F}_q$
 2 set $\mathcal{R} := \mathbb{F}[x_1, \ldots, x_n]$
 3 set the secret key $\boldsymbol{s} := (s_1, ..., s_n) \xleftarrow{R} \mathbb{F}^n$
 4 for $i = 1$ to $n$ do
 5     for $l = 1$ to $\nu$ do
 6         $t_l^{(i)} \xleftarrow{R} \mathbb{F} \setminus \left\{t_1^{(i)}, \ldots, t_{l-1}^{(i)}\right\}$
 7 for $i = 1$ to $n$ do
 8     set $g_i := (x_i - s_i) \cdot \prod_{l=1}^{\nu} \left(x_i - t_l^{(i)}\right)$
 9 set the multiplication key $G := \{g_1, \ldots, g_n\}$
10 set $\mathcal{C} := \mathcal{R}/\langle G \rangle$
11 return $(\boldsymbol{s}, G)$
```

Finally, the special choice of polynomials g_i allows us to use the set $\{f \in \mathcal{R} \mid \deg_{x_i}(f) \leq \nu,\ i = 1, \ldots, n\}$ as the support set of $\mathcal{C}$.

Algorithm 5 describes the encryption procedure. In Step 1, we choose a polynomial $f \in \mathcal{R}$ uniformly at random, s.t. $\deg_{x_i}(f) \leq \nu$ for all i, hence $f \in \mathcal{C}$ and also $c \in \mathcal{C}$. Note, that according the our notation the operations used in Step 2 are the operations in $\mathcal{R}$ and not in $\mathcal{C}$. We will comment on this later on.

Decryption is described by Alg. 6. Let $\boldsymbol{s}$ be a secret key and $m \in \mathbb{F}$ a message. Then, by Step 2 of Alg. 5, $e_{\boldsymbol{s}}(m) = c = f - f(\boldsymbol{s}) + m$ for some random f and $d_{\boldsymbol{s}}(e_{\boldsymbol{s}}(m)) = d_{\boldsymbol{s}}(c) = c(\boldsymbol{s}) = f(\boldsymbol{s}) - f(\boldsymbol{s}) + m = m$.

Algorithm 7 describes the addition operation $+_{\mathcal{C}}$ in $\mathcal{C}$. From the definition of polynomials g_i in Alg. 4 it follows, that the addition in the factor ring $\mathcal{C} = \mathcal{R}/\langle G \rangle$ is the same as the addition in the polynomial ring $\mathcal{R}$ used in Step 1 of Alg. 7. This also clarifies the operations used in Step 2 of Alg. 5.

Algorithm 5. ENCRYPT

```
    Input: message $m \in \mathbb{F}$, secret key $\boldsymbol{s} \in \mathbb{F}^n$
    Output: ciphertext $c \in \mathcal{C}$
1 $f \overset{R}{\leftarrow} \{h \in \mathcal{R} \mid \deg_{x_i}(h) \leq \nu,\ \forall i = 1, \ldots, n\}$
2 set $c := f - f(\boldsymbol{s}) + m \in \mathcal{C}$
3 return $c$
```

Algorithm 6. DECRYPT

```
    Input: ciphertext $c \in \mathcal{C}$, secret key $\boldsymbol{s} \in \mathbb{F}^n$
    Output: message $m \in \mathbb{F}$
1 set $m := c(\boldsymbol{s})$
2 return $m$
```

Finally, Alg. 8 describes multiplication in $\mathcal{C}$. It follows from Theorem 3, that for $c_1, c_2 \in \mathcal{C}$, $c_1 \cdot c_2 \bmod G$ is uniquely determined.

Algorithm 7. ADD, $+_{\mathcal{C}}$

```
    Input: ciphertexts $c_1, c_2 \in \mathcal{C}$
    Output: ciphertext $c \in \mathcal{C}$
1 set $c := c_1 + c_2$
2 return $c$
```

From the random choice of f in Step 1 of Alg. 5 it follows, that SymPC is a probabilistic cryptosystem. Now we prove that it is fully homomorphic.

Theorem 4. *The cryptosystem SymPC is fully homomorphic.*

Proof. Let $\boldsymbol{s}$ be a secret key. Let $\varphi : \mathcal{R} \to \mathbb{F}$ be the evaluation homomorphism defined as $\varphi(f) = f(\boldsymbol{s})$. By definition, $\mathrm{Ker}(\varphi) = \{f \mid f(\boldsymbol{s}) = 0\}$. Since $g_i(\boldsymbol{s}) = 0$ for all $i = 1, \ldots, n$, we get that $\langle G \rangle \subseteq \mathrm{Ker}(\varphi)$. By the Fundamental theorem on homomorphisms, $d_{\boldsymbol{s}} : \mathcal{C} \to \mathbb{F},\ d_{\boldsymbol{s}}(c) = c(\boldsymbol{s})$ is a ring homomorphism. □

From now on, we will use SymPC(n, ν, q) to denote the cryptosystem SymPC with security parameters n, ν, q.

Algorithm 8. MULTIPLY, $\cdot_{\mathcal{C}}$

```
    Input: ciphertexts $c_1, c_2 \in \mathcal{C}$, multiplication key $G$
    Output: ciphertext $c \in \mathcal{C}$
1 set $c := c_1 \cdot c_2 \bmod G$
2 return $c$
```

5 Complexity

We evaluate the complexity of each function of $\mathrm{SymPC}(n, \nu, q)$. We denote α the number of terms in $\mathcal{C}$, $\alpha = (\nu + 1)^n$.

- `SETUP` The most complex operation is generation of the polynomials g_i. For each g_i we need to perform ν multiplications of a polynomial of degree one with a polynomial of degree at most ν in $\mathbb{F}[x_i]$. The complexity is $O(n \cdot \nu^2)$ operations in $\mathbb{F}$.
- `ENCRYPT` The most complex operation is the evaluation of f in $\boldsymbol{s} \in \mathbb{F}^n$. The algorithm performs $\log_2 q \cdot \alpha$ assignments of random bits to coefficients of f and α evaluations of monomials in $\mathcal{C}$. Each of these evaluations consists of at most $\deg(f) \leq \nu \cdot n$ multiplications in $\mathbb{F}$. Then it adds evaluations in the monomials. The overall complexity is $O\left(n \cdot (\nu + 1)^{n+1}\right)$ operations in $\mathbb{F}$. We calculated the complexity of a naive evaluation algorithm. We can see, that the complexity of this algorithm could be optimized by the use of sparse polynomials. We will comment on that later.
- `DECRYPT` The complexity is the same as the complexity of `ENCRYPT`, that is $O\left(n \cdot (\nu + 1)^{n+1}\right)$ operations in $\mathbb{F}$.
- `ADD` The function performs α additions in $\mathbb{F}$, so the complexity is $O\left((\nu + 1)^n\right)$ operations in $\mathbb{F}$.
- `MULTIPLY` The function consists of two parts: multiplication and reduction. The first part is more complex and involves α^2 multiplications in $\mathbb{F}$. The overall complexity is $O\left((\nu + 1)^{2n}\right)$ operations in $\mathbb{F}$.

6 Security

We start the section with a few simple observations.

Proposition 5. *The cryptosystem* $\mathrm{SymPC}(n, \nu, q)$ *is not CCA secure.*

Proof. If an attacker can use the SymPC decryption oracle, he can ask for the decryption of the ciphertexts $c_1 = x_1, c_2 = x_2, \ldots, c_n = x_n$ and he will obtain the points of the secret key $s_1, s_2, \ldots, s_n$. $\square$

Proposition 6. *For the* $\mathrm{SymPC}(n, \nu, q)$ *cryptosystem, the CPA security is equivalent to the KPA security.*

Proof. CPA-security implies KPA-security in general. To prove the other implication we need to realize, that if an attacker has a known plaintext-ciphertext pair (m, c), he can get a valid plaintext-ciphertext pair (m', c') for any m' by setting $c' = c - m + m'$, as $d_{\boldsymbol{s}}(c') = c'(\boldsymbol{s}) = c(\boldsymbol{s}) - m + m' = m'$. Hence, from any known plaintext-ciphertext pair, he can devise a chosen plaintext-ciphertext pair, so SymPC needs to be CPA-secure to achieve the KPA-security. $\square$

6.1 Approximate Perfect Secrecy in Bounded CPA Model

In this section, we will prove that SymPC has approximate perfect secrecy (Definition 7) in the so-called k-bounded chosen plaintext attack (**k-bounded CPA**) model. In k-bounded CPA, an attacker can obtain at most k plaintext-ciphertext pairs for some given $k \in \mathbb{N}$. This can be ensured by allowing at most k plaintexts to be encrypted with a single key. As we will see, this limitation corresponds to the limitation that the size of the keyspace has to be larger or equal to the size of the plaintext space in order to reach perfect secrecy.

Similarly to perfect secrecy, we also assume that the attacker has unbounded computational power.

Definition 7. *Let $CS(t) = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \{e_k\}_k, \{d_k\}_k)$ be a cryptosystem with a security parameter t and P, C random variables on $\mathcal{P}, \mathcal{C}$. Let $\mathcal{F} = \{f : \mathcal{C} \to \mathbb{R}\}$ be the set of all functions from $\mathcal{C}$ to $\mathbb{R}$ and let $\delta : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ be a metric (distance) on $\mathcal{F}$. For $m \in \mathcal{P}$, $\Pr[P = m|C = c]$, $\Pr[P = m] \in \mathcal{F}$ (the later one is a constant function in c). We say, that $CS(t)$ has* ***approximate perfect secrecy****, iff for all probability distributions on $\mathcal{P}$ and all $m \in \mathcal{P}$*

$$\lim_{t\to\infty} \delta(\Pr[P = m \mid C = c], \Pr[P = m]) = 0 \ . \tag{1}$$

In other words, $CS(t)$ has approximate perfect secrecy, iff for all $m \in \mathcal{P}$ the probability $\Pr[\mathcal{P} = m \mid C = c]$ approaches $\Pr[\mathcal{P} = m]$ for almost all $c \in \mathcal{C}$ as t grows.

We will prove the approximate perfect secrecy of our cryptosystem with respect to the following simple metric.

Definition 8. *Let $f, g \in \mathcal{F} = \{f : \mathcal{C} \to \mathbb{R}\}$. Define a metric $\delta : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ as*

$$\delta(f, g) = \frac{1}{|\mathcal{C}|} \cdot \sum_{c\in\mathcal{C}} (f(c) - g(c))^2 \ .$$

Theorem 9. *Let $1 < \nu < q - 1$, $a = q/(\nu + 1)$ and $l > \log_a(q)/(\log_a(q) - 1)$. Then the cryptosystem $\mathrm{SymPC}(n, \nu, q)$ achieves approximate perfect secrecy in k-bounded CPA model for $k = n/l - 1$.*

First, let us comment on the choice of parameters. ν is the restriction on degrees of polynomials in $\mathcal{C}$ so naturally $\nu < q - 1$ otherwise there is no need for ν. We assume that a is fixed. The number of plaintext-ciphertext pairs is limited by $k = n/l - 1$. Clearly, l goes to one as q grows. Hence k goes to $n - 1$ as q grows.

Note 10. Assume that we allow (at most) k plaintexts to be encrypted with a single key. Then we can define our plaintext space as $\mathcal{P}' = \mathbb{F}^k$. Our keyspace equals to $\mathcal{K} = \mathbb{F}^n$. We see that our asymptotical bound $k \leq n - 1$ is similar to the condition $|\mathcal{K}| \geq |\mathcal{P}|$ on perfect secrecy.

Consider Alg. 4. The choice of g_i's in Step 8 implies that for the algebraic set $V(G)$ of the ideal $\langle G \rangle$ it holds

$$V(G) = \{(a_1, \ldots, a_n) \mid a_i \in \{s_i, t_1^{(i)}, \ldots, t_\nu^{(i)}\} \ \forall i = 1, \ldots, n\} \ , \tag{2}$$

and thus $|V(G)| = (\nu + 1)^n$. Set $t = |V(G)|$ and denote the elements of $V(G)$ as $V(G) = \{\boldsymbol{r^{(1)}}, \ldots, \boldsymbol{r^{(t)}}\}$.

Note 11. Although the multiplication key G is to be known only to the owners of the secret key and to a computational party, we assume that G is also known to the attacker. Since $g_i \in \mathbb{F}[x_i]$, he can successively find all the roots of g_i, i.e. $\{s_i, t_1^{(i)}, \ldots, t_\nu^{(i)}\}$, $i = 1, \ldots, n$ and he can compute $V(G)$. Indeed, the knowledge of the multiplicative key G is equivalent to the knowledge of $V(G)$.

Note 12. As we will see later on, we would like to maximize the size of the algebraic set $V(G)$. In the proof of Theorem 13 in Appendix A we show, that $\dim_{\mathbb{F}}(\mathcal{R}/\langle G \rangle) = (\nu+1)^n$. For any ideal I in $\mathcal{R}$ and its algebraic set $V(I)$ it holds

$$|V(I)| \leq \dim_{\mathbb{F}}(\mathcal{R}/I) \ . \tag{3}$$

(Proof of a version for polynomial rings over complex numbers can be found in [10] and it will hold for rings over finite fields as well.) We see that $|V(G)| = (\nu + 1)^n$ is the best we can do, hence our choice of g_is in Alg. 4 is optimal.

The proof of the following theorem can be found in Appendix A. It implies that choosing a ciphertext $c \in \mathcal{C}$ is equivalent to choosing a vector $\boldsymbol{u} \in \mathbb{F}^t$.

Theorem 13. *The mapping*

$$\begin{aligned} \varphi : \mathcal{R}/\langle G \rangle &\longrightarrow \mathbb{F}^t \\ f &\longmapsto \Big(f(\boldsymbol{r^{(1)}}), \ldots, f(\boldsymbol{r^{(t)}}) \Big) \end{aligned} \tag{4}$$

is a ring isomorphism.

Corollary 14. *Choose $c \in C$ uniformly at random. Then for all $\boldsymbol{u} \in \mathbb{F}^t$ it holds*

$$\Pr[\varphi(c) = \boldsymbol{u}] = \frac{1}{|\mathbb{F}|^t} = \frac{1}{q^t} \ , \tag{5}$$

i.e. $\varphi(c)$ is distributed uniformly over $\mathbb{F}^t$. In particular, for $\boldsymbol{r} \in V(G)$ and any $a \in \mathbb{F}$, it holds

$$\Pr[c(\boldsymbol{r}) = a] = \frac{1}{|\mathbb{F}|} = q^{-1} \ . \tag{6}$$

Consider the probability distribution $\Pr[C = c]$ on $\mathcal{C}$ given by Alg. 5. We see, that the polynomial f in Step 1 of Alg. 5 is chosen uniformly at random from $\mathcal{C}$ and then in Step 2 it is "shifted" by a scalar value $m - f(\boldsymbol{s})$. Hence $\Pr[C = c]$ depends on the probability distribution on $\mathcal{P}$ and it is not necessarily the uniform distribution. However, if we define equivalence $\sim$ on $\mathcal{C}$ by $c_1 \sim c_2$ iff $c_1 - c_2 \in \mathbb{F}$

and denote the equivalence class of c by $[c]_\sim$ (i.e. $[c]_\sim = \{c + a \mid a \in \mathbb{F}\}$), then $\Pr[C \in [c]_\sim] = 1/q^{t-1}$ is the uniform distribution on $\mathcal{C}/\sim$.

In the following, we will assume that $\Pr[C = c]$ is the uniform distribution on $\mathcal{C}$. This assumption will make it easier for us to evaluate the security of our cryptosystem in the general case. We claim that the security of SymPC can be proved also without this simplification, but we leave it for the extended version of the paper.

Proof of Theorem 9

Proof. (Theorem 9) We want to prove security of SymPC(n, ν, q) in k-bounded-CPA model. Let us assume that the attacker knows the multiplicative key G (or equivalently its algebraic set $V(G)$) and polynomials $c_1, \ldots, c_k \in E_{\boldsymbol{s}}(0) = \{c \in \mathcal{C} \mid c(\boldsymbol{s}) = 0\}$, i.e. k encryptions of zero for some unknown $\boldsymbol{s}$. Recall that by Proposition 6 for SymPC CPA equals KPA.

We start with evaluation of the conditional probability $\Pr[P = m \mid C = c, c_1, \ldots, c_k]$ for some fixed $k \in \mathbb{N}, c, c_1, \ldots, c_k \in \mathcal{C}$, i.e. the probability that the plaintext equals m if we know that the ciphertext equals c and $c_1, \ldots, c_k$ are encryptions of zero.

By definition of conditional probability

$$\Pr[P = m \mid C = c, c_1, \ldots, c_k] = \frac{\Pr[P = m, C = c, c_1, \ldots, c_k]}{\Pr[C = c, c_1, \ldots, c_k]} \ , \tag{7}$$

which we can rewrite as

$$\frac{1}{\Pr[C = c, c_1, \ldots, c_k]} \sum_{\boldsymbol{r} \in V(G)} \Pr[P = m, C = c, c_1, \ldots, c_k \mid r] \cdot \Pr[r] \ . \tag{8}$$

First note, that $\Pr[C = c, c_1, \ldots, c_k] = \Pr[C = c] \cdot \prod_{i=1}^{k} \Pr[c_i]$ as these are independent events and $\Pr[C = c] = 1/q^t = \Pr[c_i]$ for all i by our assumption. The secret key is chosen uniformly at random and we know it belongs to $V(G)$, so $\Pr[\boldsymbol{r}] = 1/|V(G)| = 1/t$. Clearly if $c(\boldsymbol{r}) \neq m$ or $c_i(\boldsymbol{r}) \neq 0$ for some i, then $\Pr[P = m, C = c, c_1, \ldots, c_k \mid r] = 0$. Hence we can sum in (8) only over vectors $\boldsymbol{r}$ from the set $A = \{\boldsymbol{r} \in V(G) \mid c(\boldsymbol{r}) = m, c_1(\boldsymbol{r}) = 0, \ldots, c_k(\boldsymbol{r}) = 0\}$. Now for a fixed $\boldsymbol{r} \in A$ the choice of c is equivalent to choice of the equivalence class $[c]_\sim$ which is independent of m and uniformly distributed with probability $1/q^{t-1}$. The same is valid for all c_i. Finally, we will use the fact that the message and the secret key are independent to obtain

$$\begin{aligned} \Pr[P = m \mid C = c, c_1, \ldots, c_k] = \\ &= \frac{q^{t(k+1)}}{t} \cdot \sum_{\boldsymbol{r} \in A} \Pr[M = m] \cdot \Pr[C = c \mid r] \cdot \prod_{i=1}^{k} \Pr[c_i \mid r] \\ &= \frac{q^{k+1}}{t} \cdot \sum_{\boldsymbol{r} \in A} \Pr[M = m] \ . \end{aligned} \tag{9}$$

Set $\gamma(m, c, c_1, \dots, c_k) = |\{\boldsymbol{r} \in V(G) \mid c(\boldsymbol{r}) = m, c_1(\boldsymbol{r}) = 0, \dots, c_k(\boldsymbol{r}) = 0\}| = |A|$. Then

$$\Pr[P = m \mid C = c, c_1, \dots, c_k] = \frac{q^{k+1}}{t} \cdot \Pr[P = m] \cdot \gamma(m, c, c_1, \dots, c_k) \ . \tag{10}$$

For $\boldsymbol{r} \in V(G)$ consider a random variable $Y_{\boldsymbol{r}}$ which equals 1 iff $r \in A$ and equals 0 otherwise. We get $\sum_{\boldsymbol{r} \in V(G)} Y_{\boldsymbol{r}} = \gamma(m, c, c_1, \dots, c_k)$ and Corollary 14 implies, that it has the binomial distribution with parameters t and $1/q^{k+1}$, which is denoted by $B(t, 1/q^{k+1})$. It is well known, that $B(t, 1/q^{k+1})$ has the expected value $\mathrm{E}[\gamma] = t/q^{k+1}$ and variance $\mathrm{Var}[\gamma] = t/q^{k+1} \cdot (q^{k+1} - 1)/q^{k+1}$.

Now we can calculate the distance between $\Pr[P = m \mid C = c, c_1, \dots, c_k]$ and $\Pr[P = m]$ with respect to δ from Definition 8 to show that $\mathrm{SymPC}(n, \nu, q)$ achieves approximate perfect secrecy in k-bounded CPA model.

$$\begin{aligned}
&\delta(\Pr[P = m \mid C = c, c_1, \dots, c_k], \Pr[P = m]) = \\
&\quad = \frac{1}{|C|} \cdot \sum_{c \in \mathcal{C}} (\Pr[P = m | C = c, , c_1, \dots, c_k] - \Pr[P = m])^2 \\
&\quad = \frac{1}{|C|} \cdot \sum_{c \in \mathcal{C}} \left(\frac{q^{k+1}}{t} \cdot \Pr[P = m] \cdot \gamma(m, c, c_1, \dots, c_k) - \Pr[P = m] \right)^2 \\
&\quad = \Pr[P = m]^2 \cdot \frac{q^{2(k+1)}}{t^2} \cdot \sum_{c \in \mathcal{C}} \frac{1}{|C|} \cdot \left(\gamma(m, c, c_1, \dots, c_k) - t/q^{k+1} \right)^2 \\
&\quad = \Pr[P = m]^2 \cdot \frac{q^{2(k+1)}}{t^2} \cdot \underbrace{\sum_{c \in \mathcal{C}} \frac{1}{|C|} \cdot (\gamma(m, c, c_1, \dots, c_k) - \mathrm{E}_c(\gamma(m, c, c_1, \dots, c_k)))^2}_{\mathrm{Var}(\gamma(m,c,c_1,\dots,c_k)) = t(q^{k+1}-1)/q^{2(k+1)}} \\
&\quad = \Pr[P = m]^2 \cdot \frac{q^{k+1} - 1}{t} \ .
\end{aligned}$$

We assumed $1 < \nu < q - 1$, $a = q/(\nu + 1)$ and $l > \log_a(q)/(\log_a(q) - 1)$. To finish the proof, we need to show that for $k = n/l - 1$ and a fixed a we have $\lim_{n \to \infty} (q^{k+1} - 1)/t = 0$. We have

$$\begin{aligned}
\frac{q^{k+1} - 1}{t} &= \frac{q^{\frac{n}{l}} - 1}{(\nu + 1)^n} = \frac{q^n \cdot q^{\frac{n}{l} - n} - 1}{(\nu + 1)^n} = \left(\frac{q}{\nu + 1} \right)^n \cdot q^{n \cdot (\frac{1}{l} - 1)} - \frac{1}{(\nu + 1)^n} = \\
&= \left(\frac{a}{q^{1 - \frac{1}{l}}} \right)^n - \left(\frac{1}{\nu + 1} \right)^n \ .
\end{aligned}$$

Clearly, $(1/(\nu + 1))^n$ goes to zero as n grows (recall that $a = q/(\nu + 1)$ is fixed). We assumed $l > \log_a(q)/(\log_a(q) - 1)$, which implies $1 < (1 - 1/l) \cdot \log_a(q)$. Hence $a < q^{1 - 1/l}$ and $(a/q^{1-1/l})^n$ goes to zero as n grows. The speed of convergence is linear with a rate of convergence $\mu = \frac{a}{q^{1 - \frac{1}{l}}}$. Altogether, we have shown that

$$\delta(\Pr[P = m \mid C = c, c_1, \dots, c_k], \Pr[P = m]) \xrightarrow{n \longrightarrow \infty} 0 \ ,$$

i.e. SymPC(n, ν, q) achieves approximate perfect secrecy in k-bounded CPA model. □

6.2 Security in KPA Model

In this section, we analyze the security of SymPC in the (unbounded) KPA model. Namely, we present a known plaintext attack that computes the secret key $\boldsymbol{s}$ and we estimate the required number of plaintext-ciphertext pairs.

Let $\boldsymbol{s} \in \mathbb{F}^n$ be a secret key and assume that the attacker knows k plaintext-ciphertext pairs. By Prop. 6, we can assume that he knows $c_1, c_2, \ldots, c_k \in \mathcal{C}$ such that $c_i(\boldsymbol{s}) = 0,\ i = 1, \ldots, k$. Following Note 11, we also assume that he knows the algebraic set $V(G)$ (recall that $|V(G)| = (\nu+1)^n$), the set of key candidates.

Now for each $\boldsymbol{r} \in V(G)$, the attacker can test whether $c_i(\boldsymbol{r}) = 0$ for all $i = 1, \ldots, k$ with complexity $O((\nu+1)^n)$. Set

$$V = \{\boldsymbol{r} \in V(G) \mid c_i(\boldsymbol{r}) = 0,\ i = 1, \ldots, k\} \ .$$

We will calculate the expected size of V, i.e. the expected number of secret key candidates given k know plaintext-ciphertext pairs.

Let φ be the mapping from Theorem 13 and assume that $\boldsymbol{s} = \boldsymbol{r^{(1)}}$. Then by Corrolary 14, the vectors $\varphi(c_1), \ldots, \varphi(c_k)$ are independent and uniformly distributed over $\{0\} \times \mathbb{F}^{t-1}$. In particular, the j-th coordinates of vectors $\varphi(c_i)$ are independent uniformly distributed elements of $\mathbb{F}$. Hence for a random $\boldsymbol{r} \in V(G),\ \boldsymbol{r} \neq \boldsymbol{s}$ we have $\Pr[c_i(\boldsymbol{r}) = 0,\ i = 1, \ldots, k] = 1/q^k$ and the random variable $|V| - 1$ has the binomial distribution $B((\nu+1)^n - 1, 1/q^k)$. We get that

$$\mathrm{E}[|V|] = 1 + \frac{(\nu+1)^n - 1}{q^k} \ .$$

Set (as in Theorem 9) $a = q/(\nu+1)$. Then $\mathrm{E}[|V|] < 1 + (\nu+1)^n/q^k = q^{n-k}a^{-n}$ and so if $q^{n-k}a^{-n} \leq 1$ then $\mathrm{E}[|V|] \leq 2$. This implies $k \geq n \cdot (\log_a(q) - 1)/\log_a(q)$ and we see that the bound for k in Theorem 9 is tight.

6.3 Security in COA Model

Here we briefly analyze the security of SymPC in COA model. Again, let $\boldsymbol{s} \in \mathbb{F}^n$ be a secret key and assume that the attacker knows ciphertexts $c_1, \ldots, c_k \in \mathcal{C}$ but not the corresponding plaintexts $m_i = c(\boldsymbol{s}),\ i = 1, \ldots, k$. Furthermore we assume that he knows $V(G) = \{\boldsymbol{r^{(1)}}, \ldots, \boldsymbol{r^{(t)}}\},\ t = (\nu+1)^n$ and also $\Pr[P = m],\ m \in \mathbb{F}$ the probability distribution on the plaintext space.

By Corrolary 14, the values of $c_i(\boldsymbol{r})$ for $\boldsymbol{r} \neq \boldsymbol{s}$ are uniformly distributed over $\mathbb{F}$. So the goal of the attacker is to distinguish the plaintext distribution $\Pr[P = m]$ from $(\nu+1)^n - 1$ independent uniform distributions given a sample of size k of each. Clearly, if $\Pr[P = m]$ is also uniform, then the attacker cannot determine $\boldsymbol{s}$ regardless of k. The other extreme distribution on P is the distribution with $\Pr[P = m] = 1$ for some $m \in \mathbb{F}$. In this case we get the known plaintext attack.

7 Sparse Version of SymPC

Back in Sect. 5 we noted, that the complexity of `ENCRYPT` is $O(n \cdot (\nu+1)^{n+1})$. If we modify Alg. 5 in such way, that in Step 2 it will choose a sparse polynomial (say number of non-zero coefficients will be bounded by some fixed $\xi \in \mathbb{N}$), the complexity of `ENCRYPT` will go down to $O(\xi \cdot n \cdot \nu)$. We believe, that the distribution of evaluations of these polynomials in $V(G)$ will stay close to the uniform distribution on $\mathbb{F}^t$ and the proof of Theorem 9 will go through even with this modification.

References

1. Fellows, M., Koblitz, N.: Combinatorial cryptosystems galore! In: Mullen, G.L., Shiue, P.J.-S. (eds.) Finite Fields: Theory, Applications, and Algorithms. Contemporary Mathematics, vol. 168, pp. 51–61. AMS (1994)
2. Steinwandt, R., Geiselmann, W., Endsuleit, R.: Attacking a polynomial-based cryptosystem: Polly cracker. International Journal of Information Security 1(3), 143–148 (2002)
3. Caboara, M., Caruso, F., Traverso, C.: Lattice polly cracker cryptosystems. J. Symb. Comput., 534–549 (2011)
4. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) STOC, pp. 169–178. ACM (2009)
5. Gentry, C., Halevi, S., Smart, N.P.: Fully Homomorphic Encryption with Polylog Overhead. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 465–482. Springer, Heidelberg (2012)
6. Brakerski, Z., Gentry, C., Vaikuntanathan, V. (leveled) fully homomorphic encryption without bootstrapping. In: Goldwasser, S. (ed.) ITCS, pp. 309–325. ACM (2012)
7. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM 56(6) (2009)
8. Albrecht, M.R., Farshim, P., Faugère, J.-C., Perret, L.: Polly Cracker, Revisited. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 179–196. Springer, Heidelberg (2011)
9. Herold, G.: Polly Cracker, Revisited, Revisited. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 17–33. Springer, Heidelberg (2012)
10. Cox, D.A., Little, J., O'Shea, D.: Ideals, varieties, and algorithms - an introduction to computational algebraic geometry and commutative algebra, 2nd edn. Undergraduate texts in mathematics, pp. 1–536. Springer (1997)

A Proofs

Theorem 3. *Let $\mathcal{R}$ and G be defined by Alg. 4. Then G is the reduced normed Gröbner basis of the ideal $\langle G \rangle$.*

Proof. By Alg. 4, $G = \{g_1, \dots, g_n\}$, $g_i := (x_i - s_i) \cdot \prod_{l=1}^{\nu} (x_i - t_l^{(i)}) \in \mathbb{F}[x_i]$. According to Theorem 1, we need to show that for all $g_i, g_j \in G$, $i \neq j$ the remainder on division of $\mathrm{spol}(f, g)$ by G equals zero. Clearly, $\mathrm{lt}(g_i) = x_i^{\nu+1}$ and

$\mathrm{lt}(g_j) = x_j^{\nu+1}$, hence $\mathrm{lcm}(\mathrm{lt}(g_i), \mathrm{lt}(g_j)) = x_i^{\nu+1} x_j^{\nu+1}$. Define $h_i = g_i - x_i^{\nu+1}$ and $h_j = g_j - x_j^{\nu+1}$. We obtain

$$\begin{aligned}\mathrm{spol}(g_i, g_j) &= \mathrm{lcm}(\mathrm{lt}(g_i), \mathrm{lt}(g_j)) \cdot g_i/\mathrm{lm}(g_i) - \mathrm{lcm}(\mathrm{lt}(g_i), \mathrm{lt}(g_j)) \cdot g_j/\mathrm{lm}(g_j) \\ &= x_j^{\nu+1}(x_i^{\nu+1} + h_i) - x_i^{\nu+1}(x_j^{\nu+1} + h_j) \\ &= x_j^{\nu+1} h_i - x_i^{\nu+1} h_j \ .\end{aligned}$$

Now, if we reduce $\mathrm{spol}(g_i, g_j)$ by g_i and g_j we get

$$\begin{aligned}\mathrm{spol}(g_i, g_j) \bmod g_i &= x_j^{\nu+1} h_i + h_i h_j \\ (\mathrm{spol}(g_i, g_j) \bmod g_i) \bmod g_j &= -h_i h_j + h_i h_j = 0 \ .\end{aligned}$$

□

Theorem 13. *The mapping*

$$\begin{aligned}\varphi : \mathcal{R}/\langle G\rangle &\longrightarrow \mathbb{F}^t \\ f &\longmapsto \left(f(\boldsymbol{r}^{(1)}), \ldots, f(\boldsymbol{r}^{(t)})\right)\end{aligned}$$

is a ring isomorphism.

Proof. As for all $i = 1, \ldots, t$, $\boldsymbol{r}^{(i)} \in V(G)$, each of the mappings $f \mapsto f(\boldsymbol{r}^{(i)})$ is a ring homomorphism. Hence φ is also a ring homomorphism.

Let $i \in \{1, \ldots, n\}$ and $u_i = (0, \ldots, 1, \ldots, 0) \in \mathbb{F}^t$ be a vector with a 1 at the i-th position. We show that we can find $f \in \mathcal{R}/\langle G\rangle$, such that $\varphi(f) = u_i$. As i has been chosen arbitrarily and φ is linear, the surjectivity of φ will follow.

The desired f needs to satisfy $f(\boldsymbol{r}^{(i)}) = 1$ and $f(\boldsymbol{r}^{(j)}) = 0$ for all $j \neq i$. For $j \neq i$ it holds $\boldsymbol{r}^{(j)} \neq \boldsymbol{r}^{(i)}$, therefore we can find an $l = l(j) \in \{1, \ldots, n\}$, such that $r_{l(j)}^{(j)} \neq r_{l(j)}^{(i)}$. For $j = 1 \ldots, i-1, i+1, \ldots, t$ we set $b_j := r_{l(j)}^{(j)}$ and $h_j := (x_{l(j)} - b_j)/(r_{l(j)}^{(i)} - b_j)$. We have $h_j(\boldsymbol{r}^{(i)}) = 1$ and $h_j(\boldsymbol{r}^{(j)}) = 0$. Set $\tilde{f} := \prod_{j=1,\ j\neq i}^{t} h_j \in \mathcal{R}$. We obtain

$$\begin{aligned}\tilde{f}(\boldsymbol{r}^{(i)}) &= \prod_{j=1,\ j\neq i}^{t} h_j(\boldsymbol{r}^{(i)}) = \prod_{j=1,\ j\neq i}^{t} 1 = 1 \ , \\ \tilde{f}(\boldsymbol{r}^{(j)}) &= h_j(\boldsymbol{r}^{(j)}) \cdot \prod_{k=1,\ k\neq i,j}^{t} h_k(\boldsymbol{r}^{(j)}) = 0,\ j = 1 \ldots, i-1, i+1, \ldots, t \ .\end{aligned}$$

Set $f := \tilde{f} + G \in \mathcal{R}/\langle G\rangle$. As $\boldsymbol{r}^{(j)} \in V(G)$ for $j = 1, \ldots, t$, we get that $f(\boldsymbol{r}^{(i)}) = \tilde{f}(\boldsymbol{r}^{(i)}) = 1$ and for all $j \neq i$, $f(\boldsymbol{r}^{(j)}) = \tilde{f}(\boldsymbol{r}^{(j)}) = 0$. So we have found $f \in \mathcal{R}/\langle G\rangle$, such that $\varphi(f) = u_i$.

In order to finish the proof, it is sufficient to show that $\dim_{\mathbb{F}}(\mathcal{R}/\langle G\rangle) = \dim_{\mathbb{F}}(\mathbb{F}^t)$ as both rings are finite. Clearly, $\dim_{\mathbb{F}}(\mathbb{F}^t) = t = (\nu+1)^n$. As G is a Gröbner basis, $\mathcal{R}/\langle G\rangle$ is as a vector space generated by all the terms in $\mathcal{R}$ irreducible by $\langle G\rangle$. By our choice of g_i in Alg. 4, these are $x_1^{j_1} \cdots x_n^{j_n}$, $j_1, \ldots, j_n \in \{0, \ldots, \nu\}$ and there are $(\nu+1)^n$ such terms. □

Weak Keys of the Full MISTY1 Block Cipher for Related-Key Differential Cryptanalysis*

Jiqiang Lu[1], Wun-She Yap[1,2], and Yongzhuang Wei[3,4]

[1] Institute for Infocomm Research,
Agency for Science, Technology and Research
1 Fusionopolis Way, Singapore 138632
lvjiqiang@hotmail.com, {jlu,wsyap}@i2r.a-star.edu.sg
[2] Faculty of Information Science and Technology, Multimedia University,
Melaka 75450, Malaysia
[3] Guilin University of Electronic Technology,
Guilin City, Guangxi Province 541004, P.R. China
[4] State Key Laboratory of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing 100190, P.R. China
walker_wei@msn.com

Abstract. The MISTY1 block cipher has a 64-bit block length, a 128-bit user key and a recommended number of 8 rounds. It is a Japanese CRYPTREC-recommended e-government cipher, a European NESSIE selected cipher, and an ISO international standard. Despite of considerable cryptanalytic efforts during the past fifteen years, there has been no published cryptanalytic attack on the full MISTY1 cipher algorithm. In this paper, we present a related-key differential attack on the full MISTY1 under certain weak key assumptions: We describe $2^{103.57}$ weak keys and a related-key differential attack on the full MISTY1 with a data complexity of 2^{61} chosen ciphertexts and a time complexity of $2^{90.93}$ encryptions. For the first time, our result exhibits a cryptographic weakness in the full MISTY1 cipher (when used with the recommended 8 rounds), and shows that the MISTY1 cipher is distinguishable from an ideal cipher and thus cannot be regarded to be an ideal cipher.

Keywords: Block cipher, MISTY1, Differential cryptanalysis, Related-key cryptanalysis, Weak key.

1 Introduction

The MISTY1 block cipher was designed by Matsui [26] and published in 1997. It has a 64-bit block length, a 128-bit user key, and a variable number of rounds;

* An earlier version of this work appeared in 2012 as part of Cryptology ePrint Archive Report 2012/066 [25]. This work was partially supported by the Natural Science Foundation of China (No. 61100185), Guangxi Natural Science Foundation (No. 2011GXNSFB018071), the Foundation of Guangxi Key Lab of Wireless Wideband Communication and Signal Processing (No. 11101), and China Postdoctoral Science Foundation Funded Project.

the officially recommended number of rounds is 8. We consider the version of MISTY1 that uses the recommended 8 rounds in this paper, which is also the most widely discussed version so far. MISTY1 has a Feistel structure with a total of ten key-dependent logical functions **FL** — two **FL** functions at the beginning plus two inserted after every two rounds. It became a CRYPTREC [7] e-government recommended cipher in 2002, and a NESSIE [27] selected block cipher in 2003, and was adopted as an ISO [11] international standard in 2005 and 2010.

MISTY1 has attracted extensive attention since its publication, and its security has been analysed against a wide range of cryptanalytic techniques [1,6,9,10,18,19,22,24,29–31]. In summary, the main previously published cryptanalytic results on MISTY1 are as follows. In 2008, Dunkelman and Keller [10] described impossible differential attacks [3,16] on 6-round MISTY1 with FL functions and 7-round MISTY1 without FL functions. In the same year, Lee et al. [22] gave a related-key amplified boomerang attack [13] on 7-round MISTY1 with FL functions under a class of 2^{73} weak keys[1], and Tsunoo et al. [30] presented a higher-order differential attack [15,20] on 6 and 7-round MISTY1 with FL functions (without making a weak key assumption). In 2009, Sun and Lai [29] presented an integral attack on 6-round MISTY1 with FL functions, building on Knudsen and Wagner's integral attack [17] on 5-round MISTY1. Following Lee et al.'s work, in 2011 Chen and Dai [6] presented a 7-round related-key amplified boomerang distinguisher with probability 2^{-118} under a class of 2^{90} weak keys and gave a related-key amplified boomerang attack on the 8-round MISTY1 with only the first 8 FL functions; and subsequently Dai and Chen [8,9] described a 7-round related-key differential characteristic with probability 2^{-60} under a class of 2^{105} weak keys and finally presented a related-key differential attack on the 8-round MISTY1 with only the last 8 FL functions.[2] By now, there has been no published (non-generic) cryptanalytic attack on the full 8 rounds of MISTY1 yet.

Related-key cryptanalysis [2,14] assumes that the attacker knows the relationship between one or more pairs of unknown keys; certain current real-world applications may allow for practical related-key attacks, for example, key-exchange protocols [12]. Related-key differential cryptanalysis [12] is a combination of differential cryptanalysis [4] and related-key cryptanalysis; it takes advantage of how a specific difference in a pair of inputs of a cipher or function can affect a difference in the pair of outputs of the cipher or function, where the pair of outputs are obtained by encrypting the pair of inputs using two different keys with a specific difference. Remarkably, under certain weak key assumptions the related-key differential cryptanalysis technique was used in 2009 by Biryukov et al. [5] to yield

[1] A class of weak keys is defined as a class of keys under which the concerned cipher is more vulnerable to be attacked.

[2] Our work is based on the version of Dai and Chen's paper that we requested from Dai in February 2012 [8]. However, we note that the post-proceedings version [9] of their paper appeared in the LNCS website a few days ago, acknowledging us, where the results were modified as given in Table 1.

Table 1. Main cryptanalytic results on MISTY1 with FL functions

#Rounds	#Keys	Attack Type	Data	Memory	Time	Source
6 $(1-6)$	2^{128}	Impossible differential	2^{51}CP	not specified	$2^{123.4}$Enc.	[10]
6 $(1-6)$	2^{128}	Higher-order differential	$2^{53.7}$CP	not specified	$2^{64.4}$Enc.	[30]
6 $(3-8)$	2^{128}	Integral	2^{32}CC	not specified	$2^{126.1}$Enc.	[29]
7 $(1-7)$	2^{128}	Higher-order differential	$2^{54.1}$CP	not specified	$2^{120.7}$Enc.	[30,31]
$7^{\dagger}$ $(2-8)$	2^{73}	Related-key amplified boo.	2^{54}CP	2^{59}Bytes	$2^{55.3}$Enc.	[22]
$8^{\dagger}$ $(1-8)$	2^{90}	Related-key amplified boo.	2^{63}CP	2^{65}Bytes	2^{70}Enc.	[6]
$8^{\dagger}$ $(1-8)$	$2^{105\ddagger}$	Related-key differential	2^{63}CC	2^{37}Bytes	$2^{86.6}$Enc.	[8]
	$2^{102.57}$	Related-key differential	2^{61}CC	2^{35}Bytes	$2^{84.6}$Enc.	[9]
full	$2^{103.57}$	Related-key differential	2^{61}CC	$2^{99.2}$Bytes	$2^{90.93}$Enc.	Sect. $4^{\S}$

†: Exclude the first/last two FL functions; ‡: There is a flaw, see Section 3 for detail;
§: Complexity is only for one class of weak keys.

the first cryptanalytic attack on the full version of the AES [28] block cipher with 256 key bits.

In this paper, we show for the very first time that the full MISTY1 cipher can be distinguished from an ideal cipher (in the related-key model), mainly from a theoretical perspective: Building on Dai and Chen's work described in [8,9], we present a related-key differential attack on the full MISTY1 cipher under certain weak key assumptions. First, we spot a flaw in Dai and Chen's differential cryptanalysis results from [8], and find that there are only about $2^{102.57}$ weak keys in their weak key class such that their 7-round related-key differential holds, but with probability 2^{-58}. Then, we use the 7-round related-key differential with probability 2^{-58} to break the full MISTY1 under the class of $2^{102.57}$ weak keys. Finally, we observe that there also exists a different class of $2^{102.57}$ weak keys under which similar results hold. Table 1 summarises our and previously published main cryptanalytic results on MISTY1, where CP and CC refer respectively to the numbers of chosen plaintexts and chosen ciphertexts, and Enc. refers to the required number of encryption operations of the relevant version of MISTY1.

We would like to mention that the original version of this paper, entitled "weak keys of the full MISTY1 block cipher for related-key cryptanalysis", contained a set of 2^{92} weak keys of the full MISTY1 for a related-key amplified boomerang attack [25], but we remove it from this proceedings version, because of page constraints.

The remainder of the paper is organised as follows. In the next section, we give the notation and describe the MISTY1 cipher. In Section 3 we review Dai and Chen's class of weak keys and their 7-round related-key differential characteristic, and give our corrected class of weak keys and 7-round related-key differential. We present our attack on MISTY1 in Section 4. In Section 5 we describe another class of weak keys. Section 6 concludes this paper.

2 Preliminaries

In this section we give the notation and briefly describe the MISTY1 cipher.

2.1 Notation

The bits of a value are numbered from left to right, starting with 1. We use the following notation throughout this paper.

$\oplus$: bitwise logical exclusive OR (XOR) of two bit strings of the same length
$\cap$: bitwise logical AND of two bit strings of the same length
$\cup$: bitwise logical OR of two bit strings of the same length
$||$: bit string concatenation

2.2 The MISTY1 Block Cipher

MISTY1 [26] employs a complex Feistel structure with a 64-bit block length and a 128-bit user key. It uses the following three functions **FL**, **FI**, **FO**, which are respectively depicted in Fig. 1-(a), Fig. 1-(b) and Fig. 1-(c) with their respective subkeys to be described below.

– **FL** : $\{0,1\}^{32} \times \{0,1\}^{32} \rightarrow \{0,1\}^{32}$ is a key-dependent linear function. If $X = (X_L||X_R)$ is a 32-bit block of two 16-bit words X_L, X_R, and $Y = (Y_1||Y_2)$ is a 32-bit block of two 16-bit words Y_1, Y_2, then

$$\mathbf{FL}(X,Y) = (X_L \oplus ((X_R \oplus (X_L \cap Y_1)) \cup Y_2), X_R \oplus (X_L \cap Y_1)).$$

– **FI** : $\{0,1\}^{16} \times \{0,1\}^{16} \rightarrow \{0,1\}^{16}$ is a non-linear function. If $X = (X_L||X_R)$ and $Y = (Y_1||Y_2)$ are 16-bit blocks, here X_L, Y_2 are 9 bits long and X_R, Y_1 are 7 bits long, then $\mathbf{FI}(X,Y)$ is computed as follows, where $XL_0, XR_0, \cdots,$ XL_3, XR_3 are 9 or 7-bit variables, S_9 is a 9×9-bit bijective S-box, S_7 is a 7×7-bit bijective S-box, the function Extnd extends from 7 bits to 9 bits by concatenating two zeros on the left side, and the function Trunc truncates two bits from the left side.
  1. $XL_0 = X_L, XR_0 = X_R$;
  2. $XL_1 = XR_0$, $XR_1 = \mathrm{S}_9(XL_0) \oplus \mathrm{Extnd}(XR_0)$;
  3. $XL_2 = XR_1 \oplus Y_2$, $XR_2 = \mathrm{S}_7(XL_1) \oplus \mathrm{Trunc}(XR_1) \oplus Y_1$;
  4. $XL_3 = XR_2$, $XR_3 = \mathrm{S}_9(XL_2) \oplus \mathrm{Extnd}(XR_2)$;
  5. $\mathbf{FI}(X,Y) = (XL_3||XR_3)$.

– **FO** : $\{0,1\}^{32} \times \{0,1\}^{64} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32}$ is a non-linear function. If $X = (X_L||X_R)$ is a 32-bit block of two 16-bit words X_L, X_R, $Y = (Y_1||Y_2||Y_3||Y_4)$ is a 64-bit block of four 16-bit words Y_1, Y_2, Y_3, Y_4, and $Z = (Z_1||Z_2||Z_3)$ is a 48-bit block of three 16-bit words Z_1, Z_2, Z_3, then $\mathbf{FO}(X,Y,Z)$ is defined as follows, where $XL_0, XR_0, \cdots, XL_3$, XR_3 are 16-bit variables.
  1. $XL_0 = X_L, XR_0 = X_R$;
  2. For $j = 1,2,3$:
     $XL_j = XR_{j-1}$, $XR_j = \mathbf{FI}(XL_{j-1} \oplus Y_j, Z_j) \oplus XR_{j-1}$;
  3. $\mathbf{FO}(X,Y,Z) = (XL_3 \oplus Y_4)||XR_3$.

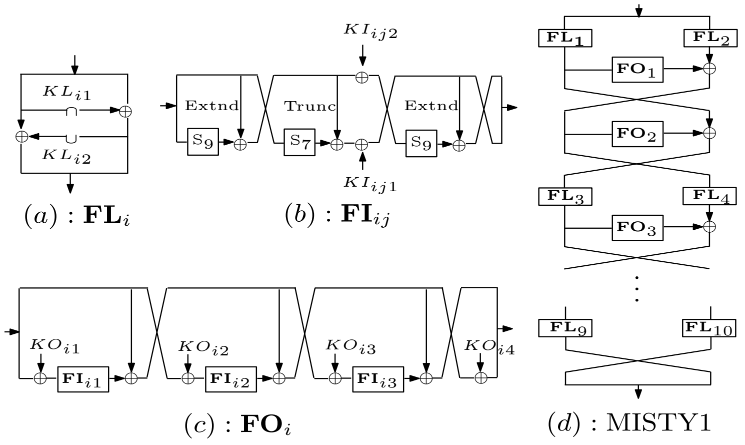


Fig. 1. MISTY1 and its components

MISTY1 uses a total of ten 32-bit subkeys $KL_1, KL_2, \cdots, KL_{10}$ for the **FL** functions, twenty-four 16-bit subkeys KI_{ij} for the **FI** functions, and thirty-two 16-bit subkeys KO_{il} for the **FO** functions, $(1 \leqslant i \leqslant 8, 1 \leqslant j \leqslant 3, 1 \leqslant l \leqslant 4)$, all derived from a 128-bit user key K. The key schedule is as follows.

1. Represent K as eight 16-bit words $K = (K_1, K_2, \cdots, K_8)$.
2. Generate a different set of eight 16-bit words $K'_1, K'_2, \cdots, K'_8$ by

$$K'_i = \mathbf{FI}(K_i, K_{i+1}), \text{ for } i = 1, 2, \cdots, 8,$$

   where the subscript $i+1$ is reduced by 8 when it is larger than 8, (similar for some subkeys in the following step).
3. The subkeys are as follows.

$$KO_{i1} = K_i, KO_{i2} = K_{i+2}, KO_{i3} = K_{i+7}, KO_{i4} = K_{i+4};$$
$$KI_{i1} = K'_{i+5}, KI_{i2} = K'_{i+1}, KI_{i3} = K'_{i+3};$$
$$KL_i = K_{\frac{i+1}{2}} || K'_{\frac{i+1}{2}+6}, \text{ for } i = 1, 3, 5, 7, 9; \text{otherwise}, KL_i = K'_{\frac{i}{2}+2} || K_{\frac{i}{2}+4}.$$

MISTY1 takes a 64-bit plaintext P as input, and has a variable number of rounds; the officially recommended number of rounds is 8. Its encryption procedure is as follows, where $L_0, R_0, \cdots, L_i, R_i$ are 32-bit variables, $KO_j = (KO_{j1}||KO_{j2}||KO_{j3}||KO_{j4})$, and $KI_j = (KI_{j1}||KI_{j2}||KI_{j3})$, $(j = 1, 2, \cdots, 8)$; see Fig. 1-(d).

1. $(L_0||R_0) = (P_L||P_R)$.
2. For $i = 1, 3, 5, 7$:

$R_i = \mathbf{FL}(L_{i-1}, KL_i)$, $L_i = \mathbf{FL}(R_{i-1}, KL_{i+1}) \oplus \mathbf{FO}(R_i, KO_i, KI_i)$;
$R_{i+1} = L_i$, $L_{i+1} = R_i \oplus \mathbf{FO}(L_i, KO_{i+1}, KI_{i+1})$.

3. Ciphertext $C = \mathbf{FL}(R_8, KL_{10}) || \mathbf{FL}(L_8, KL_9)$.

We refer to the 8 rounds in the above description as Rounds $1, 2, \cdots, 8$, respectively.

3 A Related-Key Differential for 7-Round MISTY1 under a Class of $2^{102.57}$ Weak Keys

In this section, we first review Dai and Chen's class of 2^{105} weak keys and their 7-round related-key differential characteristic with probability 2^{-60} under the class of weak keys. Then, we show that there are actually only $2^{102.57}$ weak keys such that the 7-round related-key differential characteristic holds, and it has a probability of 2^{-58}.

3.1 A Class of 2^{105} Weak Keys Owing to Dai and Chen

First define three constants which will be used subsequently: A 7-bit constant $a = 0010000$, a 16-bit constant $b = 0010000000010000$, and another 16-bit constant $c = 0010000000000000$, all in binary notation. Observe that $b = (a||0^2||a)$ and $c = (a||0^9)$, where 0^2 represents a binary string of 2 zeros, and so on.

Let K_A, K_B be two 128-bit user keys defined as follows:

$$K_A = (K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8),$$
$$K_B = (K_1, K_2, K_3, K_4, K_5, K_6^*, K_7, K_8).$$

By the key schedule of MISTY1 we can get the corresponding eight 16-bit words for K_A, K_B, which are denoted as follows.

$$K'_A = (K'_1, K'_2, K'_3, K'_4, K'_5, K'_6, K'_7, K'_8),$$
$$K'_B = (K'_1, K'_2, K'_3, K'_4, K'^*_5, K'^*_6, K'_7, K'_8).$$

Then, the class of weak keys is defined to be the set of all possible values for (K_A, K_B) that satisfy the following 10 conditions, where $K_{6,12}$ denotes the 12-th bit of K_6, and similar for $K_{7,3}, K_{7,12}, K_{8,3}, K'_{4,3}, K'_{4,12}$, $K'_{7,3}$.

$$K_6 \oplus K_6^* = c; \quad (1)$$
$$K'_5 \oplus K'^*_5 = b; \quad (2)$$
$$K'_6 \oplus K'^*_6 = c; \quad (3)$$
$$K_{6,12} = 0; \quad (4)$$
$$K_{7,3} = 1; \quad (5)$$
$$K_{7,12} = 0; \quad (6)$$
$$K_{8,3} = 1; \quad (7)$$
$$K'_{4,3} = 1; \quad (8)$$
$$K'_{4,12} = 1; \quad (9)$$
$$K'_{7,3} = 0. \quad (10)$$

Now let us analyse the number of the weak keys. First observe that when Condition (1) holds, then Condition (2) holds with certainty.

Note that $K'_4 = \mathbf{FI}(K_4, K_5), K'_6 = \mathbf{FI}(K_6, K_7),\ K'^*_6 = \mathbf{FI}(K^*_6, K_7), K'_7 = \mathbf{FI}(K_7, K_8)$. By performing a computer search, we get

$$|\{(K_4, K_5)|\text{Conditions (8) and (9)}\}| = 2^{30};$$
$$|\{(K_6, K_7, K_8)|\text{Conditions (1), (3), (4), (5), (6), (7) and (10)}\}| = 2^{27}.$$

Therefore, Dai and Chen [8] concluded that there are a total of 2^{105} possible values for K_A satisfying the above 10 conditions, and thus there are 2^{105} weak keys.

3.2 Dai and Chen's 7-Round Related-Key Differential Characteristic

Under the class of 2^{105} weak keys (K_A, K_B) described in Section 3.1, Dai and Chen described the following 7-round related-key differential characteristic $\Delta\alpha \to \Delta\beta$: $(b||0^{32}||c) \to (0^{32}||c||0^{16})$ with probability 2^{-60} for Rounds 2–8. In Fig. 3 in the Appendix we illustrate the related-key differential characteristic in detail, where $R_{4,3}$ denotes the 3-rd bit of R_4 (the right half of the output of Round 4), and $R_{4,12}$ denotes the 12-th bit of R_4.

As a result, Dai and Chen presented a related-key differential attack on 8-round MISTY1 without the first two FL functions, by conducting a key recovery on $\mathbf{FO}_1$ (in a way similar to the early abort technique for impossible differential cryptanalysis introduced in [24] as well as in Chapter 4.2 of [23]).

3.3 A Corrected Class of Weak Keys and Improved 7-Round Related-Key Differential

We first focus on the $\mathbf{FI}_{73}$ function in Dai and Chen's 7-round related-key differential characteristic, where the probability is 2^{-16}. Observe that $KI_{73} = K'_2$. Dai and Chen assumed a random distribution when calculating the probability of the differential $\Delta c \to \Delta c$ for $\mathbf{FI}_{73}$, and thus obtained a probability value of 2^{-16}, (An alternative explanation is to consider the two S_9 S-boxes, each having a probability value of 2^{-8}). However, intuitively we should make sure that a weak key (K_A, K_B) should also satisfy the condition that the differential $\Delta c \to \Delta c$ is a possible differential for $\mathbf{FI}_{73}$; otherwise, the differential $\Delta c \to \Delta c$ would have a zero probability, and the 7-round differential characteristic would be flawed. Thus, we should put the following additional condition when defining a set of weak keys:

$$\Pr_{\mathbf{FI}(\cdot, K'_2)}(\Delta c \to \Delta c) > 0. \tag{11}$$

Motivated by this, we perform a computer program to test the number of K'_2 satisfying Condition (11), and we find that the number of K'_2 satisfying Condition (11) is equal to 2^{15}. As a consequence, we know that the number of (K_2, K_3) satisfying Condition (11) is 2^{31}, thus not all 2^{32} possible values for (K_2, K_3) meet

Condition (11), so this is really a flaw in Dai and Chen's results.[3] Furthermore, we find that for each satisfying K'_2, there are exactly two pairs of inputs to $\mathbf{FI}_{73}$ which follow the differential $\varDelta c \to \varDelta c$, that is to say, the probability $\Pr_{\mathbf{FI}(\cdot,K'_2)}(\varDelta c \to \varDelta c) = 2^{-15}$, twice as large as the probability value 2^{-16} used by Dai and Chen.

Next we focus on the $\mathbf{FI}_{21}$ function in Dai and Chen's 7-round related-key differential characteristic, where the probability is 2^{-16}, and $KI_{21} = K'_7$. Likewise, we should make sure that a weak key (K_A, K_B) should also satisfy the condition that the differential $\varDelta b \to \varDelta c$ is a possible differential for $\mathbf{FI}_{21}$; otherwise, the differential $\varDelta b \to \varDelta c$ would have a zero probability, and the 7-round differential characteristic would be flawed. Similarly, we should put another condition when defining a set of weak keys:

$$\Pr_{\mathbf{FI}(\cdot,K'_7)}(\varDelta b \to \varDelta c) > 0. \tag{12}$$

By performing a computer program we find that the number of K'_7 satisfying Condition (12) is $24320 \approx 2^{14.57}$; on the other hand, the number of K'_7 satisfying Conditions (1), (3), (4), (5), (6), (7) and (10) is 2^{15} (and for each satisfying K'_7 there are 2^{12} possible values for (K'_6, K_8)), so not all the possible values of K'_7 satisfying Conditions (1), (3), (4), (5), (6), (7) and (10) satisfy Condition (12). After a further test, we get that the number of K'_7 satisfying Conditions (1), (3), (4), (5), (6), (7), (10) and (12) is $12160 \approx 2^{13.57}$. As a result, we know that the number of (K_6, K_7, K_8) satisfying Conditions (1), (3), (4), (5), (6), (7), (10) and (12) is $2^{13.57} \times 2^{12} = 2^{25.57}$, so this is another flaw in Dai and Chen's results. Furthermore, we have that $\Pr_{\mathbf{FI}(\cdot,K'_7)}(\varDelta b \to \varDelta c)$ is 2^{-15} for each of 9600 satisfying values for K'_7, 2^{-14} for each of 2432 satisfying values for K'_7, and $\frac{6}{2^{16}} \approx 2^{-13.42}$ for each of 128 satisfying values for K'_7.

In summary, there are approximately $2^{102.57}$ weak keys satisfying Conditions (1)–(12), and the 7-round related-key differential $\varDelta\alpha \to \varDelta\beta$ has a minimum probability of 2^{-58} under a weak key (K_A, K_B). In particular, we have the following result.

Proposition 1. *In the class of* $2^{102.57}$ *weak keys satisfying Conditions (1)–(12),*

1. *there are* 2^{16} *possible values for* K_1, 2^{16} *possible values for* K_3, *and* 2^{16} *possible values for* K_5*;*
2. *there are* $2^{25.57}$ *possible values for* (K_6, K_7, K_8)*; in particular there are a total of* $2^{13.57}$ *possible values for* K'_7, *and for every possible value of* K'_7 *there are* 2^{12} *possible values for* (K'_6, K_8)*;*
3. *there are a total of* 2^8 *possible values for* $K'_{2,8-16}$, 2^{16} *possible values for* K'_3, *and* 2^8 *possible values for* $K'_{4,8-16}$, *where* $K'_{2,8-16}$ *denotes bits* $(8, \cdots, 16)$ *of* K'_2 *and* $K'_{4,8-16}$ *denotes bits* $(8, \cdots, 16)$ *of* K'_4*;*
4. $\Pr_{\mathbf{FI}(\cdot,\forall K'_7)}(\varDelta b \to \varDelta c) \geq 2^{-15}, \Pr_{\mathbf{FI}(\cdot,\forall K'_2)}(\varDelta c \to \varDelta c) = 2^{-15}$.

[3] Note that this is not a mistake under the stochastic equivalence hypothesis for differential cryptanalysis given in [21], although it contradicts the fact.

4 Related-Key Differential Attack on the Full MISTY1 under the Class of $2^{102.57}$ Weak Keys

In this section, we devise a related-key differential attack on the full MISTY1 under a weak key from the class of $2^{102.57}$ weak keys, basing it on the 7-round related-key differential with probability 2^{-58}.

4.1 Preliminary Results

We first concentrate on the propagation of the input difference $\alpha(= b||0^{32}||c)$ of the 7-round differential through the preceding Round 1, including the $\mathbf{FL}_1$ and $\mathbf{FL}_2$ functions, under (K_A, K_B); see Fig. 2.

Under (K_A, K_B), by the key schedule of MISTY1 we have

$$\begin{aligned}
&\varDelta KO_{11} = \varDelta K_1 = 0, \varDelta KO_{12} = \varDelta K_3 = 0,\\
&\varDelta KO_{13} = \varDelta K_8 = 0, \varDelta KO_{14} = \varDelta K_5 = 0,\\
&\varDelta KI_{11} = \varDelta K'_6 = c, \varDelta KI_{12} = \varDelta K'_2 = 0, \varDelta KI_{13} = \varDelta K'_4 = 0,\\
&\varDelta KL_1 = \varDelta(K_1||K'_7) = 0, \varDelta KL_2 = \varDelta(K'_3||K_5) = 0.
\end{aligned}$$

As depicted in Fig. 2, the right half of α is $(0^{16}||c)$, so the $\mathbf{FI}_{11}$ function has a zero input difference; however since $\varDelta KO_{11} = 0$ and $\varDelta KI_{11} = c$, the output difference of $\mathbf{FI}_{11}$ is b with probability 1. The input difference of the $\mathbf{FI}_{12}$ function is c, thus the first S_9 function in $\mathbf{FI}_{12}$ has an input difference $a||0^2$, and we assume its output difference is $A \in \{0,1\}^9$; the S_7 function in $\mathbf{FI}_{12}$ has a zero input and output difference. The second S_9 function in $\mathbf{FI}_{12}$ has an input difference A, and we assume its output difference is $B \in \{0,1\}^9$. As a result, the $\mathbf{FI}_{12}$ function has an output difference $X = (\mathrm{Trunc}(A)||(B \oplus (0^2||\mathrm{Trunc}(A))))$. A simple computer program reveals that $\mathrm{Trunc}(A)$ can take all 2^7 possible values, and thus we assume that X can take all values in $\{0,1\}^{16}$.

Since the input difference of the $\mathbf{FI}_{13}$ function is $0^9||a$, the first S_9 function in $\mathbf{FI}_{13}$ has a zero input difference. The S_7 function in $\mathbf{FI}_{13}$ has an input difference a, and we assume its output difference is $D \in \{0,1\}^7$, which can take only 2^6 possible values. The second S_9 function in $\mathbf{FI}_{13}$ has an input difference $0^2||a$, and we assume its output difference is $E \in \{0,1\}^9$. Consequently, the $\mathbf{FI}_{13}$ function has an output difference $Y = ((a \oplus D)||(E \oplus (0^2||(a \oplus D))))$, and it can take about 2^{15} values in $\{0,1\}^{16}$; we denote the set of 2^{15} values by $\mathcal{S}_d$.

The $\mathbf{FL}_1$ function has an output difference $(0^{16}||c)$, so its input difference can only be of the form $\overbrace{00?0000000000000||00?0000000000000}^{32\ bits}$, which will be denoted by $\eta = (\eta_L, \eta_R)$ in the following descriptions, where the question marker "?" represents an indeterminate bit; and when the first question marker takes a zero value, the second question marker can take only 1, that is η has only three possible values, (The specific form depends on the values of the two subkey bits $K_{1,3}$ and $K'_{7,3}$). The $\mathbf{FL}_2$ function has an output difference $(X \oplus c)||(X \oplus Y \oplus (0^9||a))$, so its input difference is indeterminate, denoted by "?" in Fig. 2.

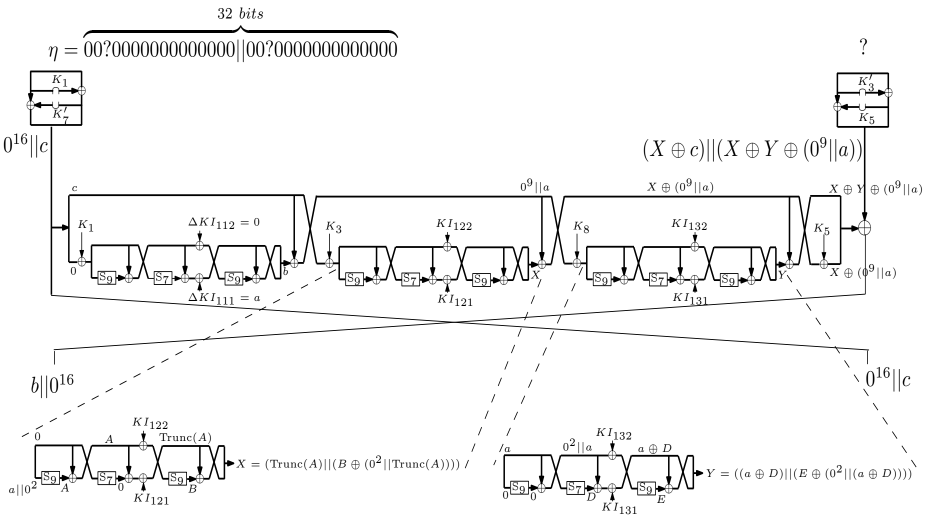


Fig. 2. Propagation of α through the inverse of Round 1 with $\mathbf{FL}_1$ and $\mathbf{FL}_2$

From the above analysis we can see that the subkeys KI_{121} and KI_{131} do not affect the values of X and Y, and thus they are not required when checking whether a candidate plaintext pair generates the input difference $\alpha = (b||0^{32}||c)$ of the 7-round related-key differential. Further, as $K'_3 = \mathbf{FI}(K_3, K_4), K'_4 = \mathbf{FI}(K_4, K_5)$, $K'_6 = \mathbf{FI}(K_6, K_7)$ and $K'_7 = \mathbf{FI}(K_7, K_8)$, we obtain the following result.

Proposition 2. *Only the subkeys* $(K_1, K'_{2,8-16}, K_3, K_4, K_5, K_6, K_7, K_8)$ *are required when checking whether a candidate plaintext pair produces the input difference* $\alpha = (b||0^{32}||c)$ *of the 7-round related-key differential.*

4.2 Attack Procedure

We first precompute two hash tables $\mathcal{T}_1$ and $\mathcal{T}_2$. Observe that from the left halves of a pair of plaintexts we only need $(K_1, K_3, K'_{2,8-16})$ when computing the output difference X of the $\mathbf{FI}_{12}$ function and only need $(K_1, K'_6, K'_7, K_8, K'_{4,8-16})$ when computing the output difference Y of the $\mathbf{FI}_{13}$ function. To generate $\mathcal{T}_1$ and $\mathcal{T}_2$, we do the following procedure under every 32-bit value $x = (x_L||x_R)$.

1. For every possible K_1:
   (a) Compute $Z = (x_L \cap K_1) \oplus ((x_L \oplus \eta_L) \cap K_1) \oplus \eta_R$, and proceed to the following steps only when $Z = c$.
   (b) For every possible $(K_3, K'_{2,8-16})$, compute the output difference of $\mathbf{FI}_{12}$ as X.

2. Store all satisfying $(K_1, K_3, K'_{2,8-16})$ into Table $\mathcal{T}_1$ indexed by (x, η, X).
3. For every possible K'_7:
   (a) Compute $W = \eta_L \oplus (((x_L \cap K_1) \oplus x_R) \cup K'_7) \oplus (((x_L \cap K_1) \oplus x_R \oplus c) \cup K'_7)$, and proceed to the following steps only when $W = 0$.
   (b) For every possible $(K'_6, K_8, K'_{4,8-16})$, compute the output difference of $\mathbf{FI}_{13}$ as Y.
4. Store the values of (K_6, K_7, K_8) corresponding to all satisfying (K'_6, K'_7, K_8) into Table $\mathcal{T}_2$ indexed by $(x, \eta, Y, K_1, K'_{4,8-16})$.

There are 2^{16} possible values for K_1, 2^{16} possible values for K_3, 2^8 possible values for $K'_{2,8-16}$, and 3 possible values for η. For a fixed (x, η, X), on average there are $2^{16} \times 2^{-1} \times 2^{16} \times 2^8 \times 2^{-16} = 2^{23}$ satisfying values for $(K_1, K_3, K'_{2,8-16})$ in $\mathcal{T}_1$. The precomputation for $\mathcal{T}_1$ takes about $2^{32} \times 3 \times 2^{16} \times 2^{16} \times 2^8 \approx 2^{73.59}$ **FI** computations, and $\mathcal{T}_1$ requires a memory of about $2^{24} \times 2^{32} \times 3 \times 2^{16} \times \frac{16+16+8}{8} \approx 2^{75.91}$ bytes. There are $2^{13.57}$ possible values for K'_7, 2^{12} possible values for (K'_6, K_8), 2^8 possible values for $K'_{4,8-16}$, and 2^{15} possible values for Y. For a fixed $(x, \eta, Y, K_1, K'_{4,8-16})$, on average there are $2^{13.57} \times 2^{-1} \times 2^{12} \times 2^{-15} = 2^{9.57}$ satisfying values for (K'_6, K'_7, K_8) in $\mathcal{T}_2$. The precomputation for $\mathcal{T}_2$ takes about $2^{32} \times 3 \times 2^{16} \times 2^{13.57} \times 2^{12} \times 2^8 \times 2 \approx 2^{84.16}$ **FI** computations, and $\mathcal{T}_2$ requires a memory of about $2^{9.57} \times 2^{32} \times 3 \times 2^{15} \times 2^{16} \times 2^8 \times 6 \approx 2^{84.74}$ bytes. Note that we can use several tricks to optimise the procedure to reduce the computational complexity for generating the two tables, but anyway it is negligible compared with the computational complexity of the following online attack procedure.

We devise the following attack procedure to break the full MISTY1 when a weak key is used.

1. Initialize zero to an array of $2^{95.57}$ counters corresponding to all the $2^{95.57}$ possible values for $(K_1, K'_{2,8-16}, K_3, K_4, K_5, K_6, K_7, K_8)$.
2. Choose 2^{60} ciphertext pairs $(C, C^* = C \oplus (0^{32}||c||0^{16}))$. In a chosen-ciphertext attack scenario, obtain the plaintexts for the ciphertexts C, C^* under K_A, K_B, respectively, and we denote the plaintext for ciphertext C encrypted under K_A by $P = (PL_L||PL_R, PR_L||PR_R)$, and the plaintext for ciphertext C^* encrypted under K_B by $P^* = (PL^*_L||PL^*_R, PR^*_L||PR^*_R)$.
3. Check whether a plaintext pair (P, P^*) meets the condition $(PL_L||PL_R) \oplus (PL^*_L||PL^*_R) = \eta$ by first checking the 30 bit positions with a zero difference and then checking the remaining two bit positions. Keep only the satisfying plaintext pairs.
4. For every remaining plaintext pair (P, P^*), do the following sub-steps.
   (a) Guess a possible value for (K'_3, K_5), and compute (X, Y) such that

$$(X \oplus c)||(X \oplus Y \oplus (0^9||a)) = \mathbf{FL}(PR_L||PR_R, K'_3||K_5) \oplus \mathbf{FL}(PR^*_L||PR^*_R, K'_3||K_5).$$

   Execute the next steps only if $Y \in \mathcal{S}_d$; otherwise, repeat this step with another subkey guess.
   (b) Access Table $\mathcal{T}_1$ at entry $(PL_L||PL_R, \eta, X)$ to get the satisfying values for $(K_1, K_3, K'_{2,8-16})$.

(c) For each satisfying value for $(K_1, K_3, K'_{2,8-16})$, retrieve K_4 from the equation $K'_3 = \mathbf{FI}(K_3, K_4)$, compute $K'_4 = \mathbf{FI}(K_4, K_5)$, and access Table $\mathcal{T}_2$ at entry $(PL_L||PL_R, \eta, Y, K_1, K'_{4,8-16})$ to get the satisfying values for (K_6, K_7, K_8).

(d) Increase 1 to each of the counters corresponding to the obtained values for $(K_1,\ K'_{2,8-16}, K_3, K_4, K_5,\ K_6, K_7, K_8)$.

5. For a value of $(K_1, K'_{2,8-16}, K_3, K_4, K_5, K_6, K_7, K_8)$ whose counter number is equal to or larger than 3, exhaustively search the remaining 7 key bits with two known plaintext-ciphertext pairs. If a value of $(K_1, K_2, \cdots, K_8)$ is suggested, output it as the user key of the full MISTY1.

4.3 Attack Complexity

The attack requires $2^{60} \times 2 = 2^{61}$ chosen ciphertexts. In Step 3, only $2^{60} \times 2^{-30} \times \frac{3}{4} \approx 2^{29.58}$ plaintext pairs are expected to satisfy the condition, and it takes about 2^{60} memory accesses to obtain the satisfying plaintext pairs. Step 4(a) has a time complexity of about $2^{29.58} \times 2^{16} \times 2^{16} \times 2 = 2^{62.58}$ **FL** computations. In Step 4(b), for a plaintext pair and a possible value for (K'_3, K_5), on average we obtain 2^{23} possible values for $(K_1, K_3, K'_{2,8-16})$, as discussed in the precomputation phase; owing to the filtering condition in Step 4(a), Step 4(b) has a time complexity of about $2^{29.58} \times \frac{2^{15}}{2^{16}} \times 2^{32} \times 2^{23} = 2^{83.58}$ memory accesses (if conducted on a 64-bit computer). In Step 4(c), for a plaintext pair and a possible value for $(K_1, K_3, K_5, K'_{2,8-16}, K'_3)$, on average we obtain $2^{9.57}$ possible values for (K_6, K_7, K_8), (as discussed in the precomputation phase), thus Step 4(c) has a time complexity of about $2^{28.58} \times 2^{32} \times 2^{23} \times 2^{9.57} = 2^{93.15}$ memory accesses. Step 4(d) has a time complexity of about $2^{93.15} \times 2 = 2^{94.15}$ memory accesses, where the factor "2" represents that it requires two memory accesses for a single access to an entry whose length is between 65 and 128 bits when conducted on a 64-bit computer.

The probability that the counter for a wrong $(K_1, K'_{2,8-16}, K_3, K_4, K_5, K_6, K_7, K_8)$ has a number equal to or larger than 3 is approximately $\sum_{i=3}^{2^{60}}[\binom{2^{60}}{i} \cdot (2^{-64})^i \cdot (1-2^{-64})^{2^{60}-i}] \approx 2^{-14.67}$. Thus, it is expected that there are a total of $2^{95.57} \times 2^{-14.67} = 2^{80.9}$ wrong values of $(K_1, K'_{2,8-16}, K_3, K_4, K_5, K_6, K_7, K_8)$ whose counters have a number equal to or larger than 3. Thus it requires $2^{80.9} \times 2^7 + 2^{80.9} \times 2^7 \times 2^{-64} \approx 2^{87.9}$ trial encryptions to check them in Step 5. In Step 5, a wrong value of $(K_1, K_2, \cdots, K_8)$ is suggested with probability $2^{-64\times 2} = 2^{-128}$, so the number of suggested values for $(K_1, K_2, \cdots, K_8)$ is expected to be $2^{87.9} \times 2^{-128} = 2^{-40.1}$, which is rather low. Thus, the time complexity of the attack is dominated by Steps 4(c), 4(d) and 5.

The question that how many memory accesses (table lookups) are equivalent to one MISTY1 encryption in terms of time depends closely on the used platform and MISTY1 implementation as well as the storage location of the hash table. In theoretical block cipher cryptanalysis, it is usually assumed by default that a hash table is stored in an ideal place, RAM say, like an S-box table; and it takes an almost constant time to access an entry in a hash table, independently of the number of entries. Thus, an extremely conservative estimate is: 16 memory accesses equal a full MISTY1 encryption in terms of time, assuming that in every round, Round i say, the $\mathbf{FI}_{i1}$ and $\mathbf{FI}_{i2}$ functions are implemented in parallel, equivalent to one memory access, and the subsequent $\mathbf{FI}_{i3}$ function is equivalent to one memory access, (neglecting the computational complexity for other operations and the key schedule); that is, one round is equivalent to 2 memory accesses. Therefore, the attack has a total time complexity of about $\frac{2^{93.15}+2^{94.15}}{16} + 2^{87.9} \approx 2^{90.93}$ MISTY1 encryptions.

The counter for the correct key has an expected number of $2^{60} \times 2^{-58} = 4$, and the probability that the counter for the correct key has a number equal to or larger than 3 is approximately $\sum_{i=3}^{2^{60}}[\binom{2^{60}}{i} \cdot (2^{-58})^i \cdot (1-2^{-58})^{2^{60}-i}] \approx 0.76$. Therefore, the related-key differential attack has a success probability of 76%.

The memory complexity of the attack is dominated by the space for the array of $2^{95.57}$ counters, which is $2^{95.57} \times \frac{95.57}{8} \approx 2^{99.2}$ bytes.

It is worthy to note that there exist time–memory tradeoff versions to the above attack.

5 Another Class of $2^{102.57}$ Weak Keys

We have described a class of $2^{102.57}$ weak keys and a related-key differential attack on the full MISTY1 under a weak key. However, we observe that there exists another class of $2^{102.57}$ weak keys under which similar results hold. The new weak key class is obtained by setting $K'_{7,3} = 1$, which is further classified into two sub-classes by the possible values of the subkey bit $K_{1,3}$. This will affect only the $\mathbf{FL}_{10}$ function in the 7-round related-key differential, but the output difference of $\mathbf{FL}_{10}$ will be fixed once $K_{1,3}$ is given, that is, the right half of the output difference of the resulting 7-round related-key differential will be $c||c$ when $K_{1,3} = 1$, and $0^{16}||c$ when $K_{1,3} = 0$. Thus, by choosing a number of ciphertext pairs with a corresponding difference we can conduct a similar attack on the full MISTY1 under every sub-class of weak keys.

In total, we have $2^{103.57}$ weak keys under which a related-key differential attack can break the full MISTY1 cipher algorithm.

6 Conclusions

The MISTY1 block cipher has received considerable attention and its security has been thoroughly analysed since its publication, particularly the European NESSIE project announced that "no weaknesses were found in the selected designs" when making the portfolio of selected cryptographic algorithms including MISTY1. In this paper, we have described $2^{103.57}$ weak keys for a related-key differential attack on the full MISTY1 cipher algorithm.

For the very first time, our result exhibits a cryptographic weakness in the full MISTY1 cipher algorithm, mainly from an academic point of view: The cipher does not behave like an ideal cipher (in the related-key model); thus it cannot be regarded to be an ideal cipher. From a practical point of view, our attack does not pose a significant threat to the security of MISTY1, for it works under the assumptions of weak-key and related-key scenarios and its complexity is beyond the power of a general computer of today. But nevertheless our result means that a large fraction of all possible 2^{128} keys in the whole key space of MISTY1 is weak in the sense of related-key differential cryptanalysis, roughly, one of every twenty-two million keys, and thus the chance of picking such a weak key at random is not trivial; in this sense, the presence of these weak keys has an impact on the security of the full MISTY1 cipher.

Acknowledgments. The authors thank Prof. Wenling Wu for her help, Yibin Dai for providing the final version of their paper at INSCRYPT 2011, and several anonymous referees for their comments on earlier versions of the paper.

References

1. Babbage, S., Frisch, L.: On MISTY1 Higher Order Differential Cryptanalysis. In: Won, D. (ed.) ICISC 2000. LNCS, vol. 2015, pp. 22–36. Springer, Heidelberg (2001)
2. Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 398–409. Springer, Heidelberg (1994)
3. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999)
4. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology 4(1), 3–72 (1991)
5. Biryukov, A., Khovratovich, D., Nikolić, I.: Distinguisher and Related-Key Attack on the Full AES-256. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
6. Chen, S., Dai, Y.: Related-key amplified boomerang attack on 8-round MISTY1. In: Li, C., Wang, H. (eds.) CHINACRYPT 2011, pp. 7–14. Science Press USA Inc. (2011)
7. CRYPTREC — Cryptography Research and Evaluatin Committees, report 2002 (2003)
8. Dai, Y.: Personal communications (February 2012)
9. Dai, Y.-b., Chen, S.-z.: Weak-Key Class of MISTY1 for Related-Key Differential Attack. In: Wu, C.-K., Yung, M., Lin, D. (eds.) Inscrypt 2011. LNCS, vol. 7537, pp. 227–236. Springer, Heidelberg (2012)
10. Dunkelman, O., Keller, N.: An Improved Impossible Differential Attack on MISTY1. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 441–454. Springer, Heidelberg (2008)
11. International Standardization of Organization (ISO), International Standard – ISO/IEC 18033-3, Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers (2005/2010)
12. Kelsey, J., Schneier, B., Wagner, D.: Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 237–251. Springer, Heidelberg (1996)

13. Kim, J., Hong, S., Preneel, B., Biham, E., Dunkelman, O., Keller, N.: Related-key boomerang and rectangle attacks: theory and experimental analysis. IEEE Transactions on Information Theory 58(7), 4948–4966 (2012)
14. Knudsen, L.R.: Cryptanalysis of LOKI91. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 196–208. Springer, Heidelberg (1993)
15. Knudsen, L.R.: Truncated and Higher Order Differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)
16. Knudsen, L.R.: DEAL — a 128-bit block cipher. Technical report, Department of Informatics, University of Bergen, Norway (1998)
17. Knudsen, L.R., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
18. Kühn, U.: Cryptanalysis of Reduced-Round MISTY. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 325–339. Springer, Heidelberg (2001)
19. Kühn, U.: Improved Cryptanalysis of MISTY1. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 61–75. Springer, Heidelberg (2002)
20. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Communications and Cryptography, pp. 227–233. Academic Publishers (1994)
21. Lai, X., Massey, J.L., Murphy, S.: Markov Ciphers and Differential Cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 17–38. Springer, Heidelberg (1991)
22. Lee, S., Kim, J., Hong, D., Lee, C., Sung, J., Hong, S., Lim, J.: Weak key classes of 7-round MISTY 1 and 2 for related-key amplied boomerang attacks. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 91-A(2), 642–649 (2008)
23. Lu, J.: Cryptanalysis of block ciphers. PhD thesis, University of London, UK (2008)
24. Lu, J., Kim, J., Keller, N., Dunkelman, O.: Improving the Efficiency of Impossible Differential Cryptanalysis of Reduced Camellia and MISTY1. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 370–386. Springer, Heidelberg (2008)
25. Lu, J., Yap, W.S., Wei, Y.: Weak keys of the full MISTY1 block cipher for related-key cryptanalysis. Cryptology ePrint Archive, Report 2012/066 (2012)
26. Matsui, M.: New Block Encryption Algorithm MISTY. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 54–68. Springer, Heidelberg (1997)
27. NESSIE — New European Schemes for Signatures, Integrity, and Encryption, final report of European project IST-1999-12324 (2004)
28. National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES), FIPS-197 (2001)
29. Sun, X., Lai, X.: Improved Integral Attacks on MISTY1. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 266–280. Springer, Heidelberg (2009)
30. Tsunoo, Y., Saito, T., Shigeri, M., Kawabata, T.: Higher Order Differential Attacks on Reduced-Round MISTY1. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 415–431. Springer, Heidelberg (2009)
31. Tsunoo, Y., Saito, T., Shigeri, M., Kawabata, T.: Security analysis of 7-round MISTY1 against higher order differential attacks. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 93-A(1), 144–152 (2010)

Appendix: Dai and Chen's 7-Round Related-Key Differential Characteristic

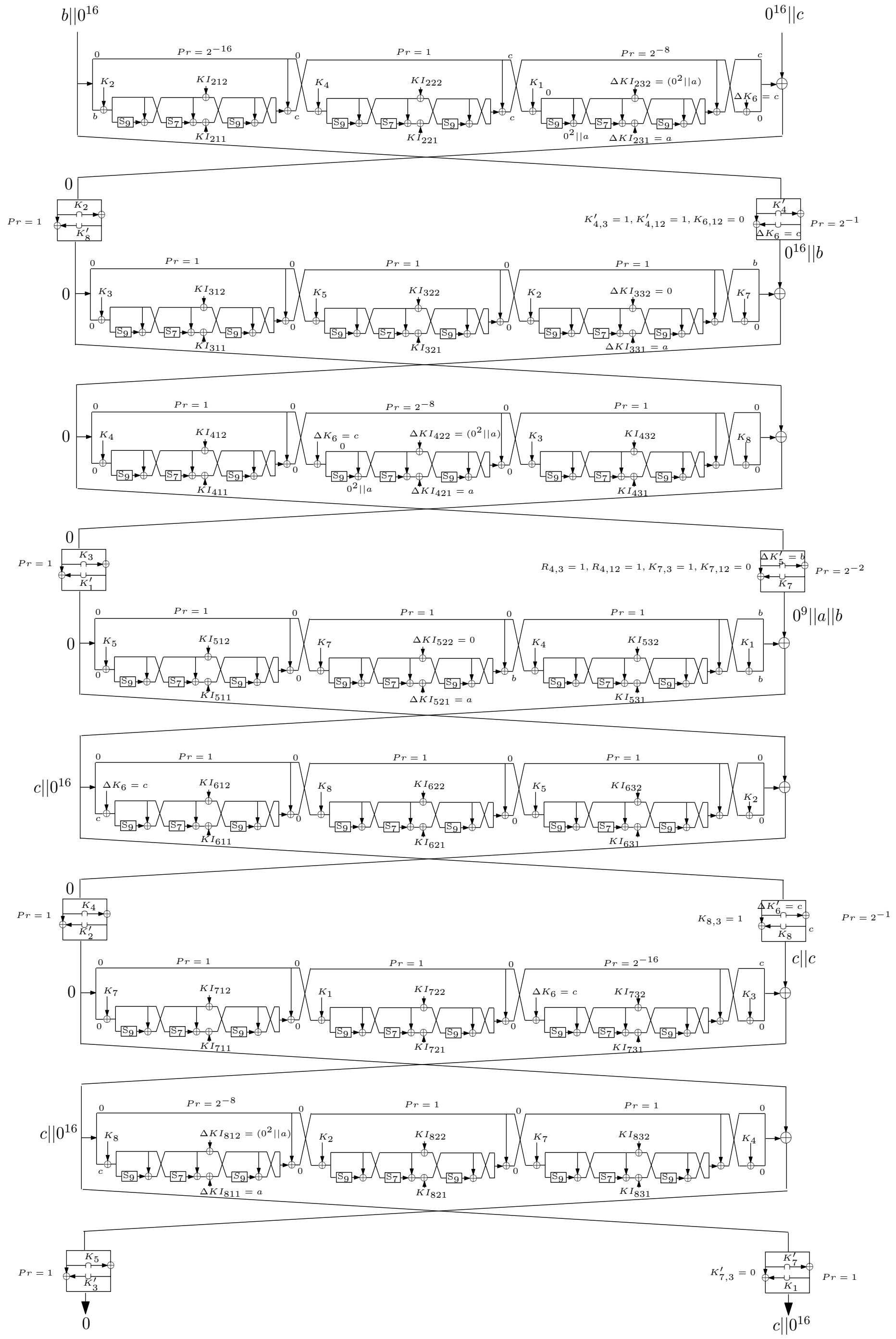


Fig. 3. Chen and Dai's related-key differential characteristic for Rounds 2–8

Author Index